Carlos Quesada Granja

Real-time simulation of surgery by Proper Generalized Decomposition techniques

Departamento

Ingeniería Mecánica

Director/es

Alfaro Ruiz, Icíar
González Ibáñez, David

ISSN 2254-7606

© Universidad de Zaragoza
Servicio de Publicaciones

ISSN 2254-7606

Tesis Doctoral

REAL-TIME SIMULATION OF SURGERY BY PROPER GENERALIZED DECOMPOSITION TECHNIQUES

Autor

Carlos Quesada Granja

Director/es

Alfaro Ruiz, Icíar
González Ibáñez, David

UNIVERSIDAD DE ZARAGOZA
Ingeniería Mecánica

2017

Real-time simulation of surgery by Proper Generalized Decomposition techniques

PhD thesis by

Carlos Quesada Granja

Doctoral advisors

Dr. Icíar Alfaro Ruiz
Dr. David González Ibáñez

PhD Programme in **Mechanical Engineering**
November 2016

Universidad
Zaragoza

1542

Real-time simulation of surgery by Proper Generalized Decomposition techniques

PhD thesis by
Carlos Quesada Granja

Doctoral advisors
Dr. Icíar Alfaro Ruiz
Dr. David González Ibáñez

PhD Programme in **Mechanical Engineering**
November 2016

Universidad
Zaragoza
1542

Agradecimientos

En primer lugar, quiero dar las gracias a mis directores de tesis, el Dr. David González y la Dra. Icíar Alfaro, por su inestimable ayuda, sus provechosos consejos y su gran empatía durante mi estancia en la Universidad de Zaragoza. En todo momento me han recibido con las puertas abiertas para resolver cualquier duda, ya fuera teórica o práctica, que me ha surgido a lo largo de estos últimos cuatro años. Incluyo también en este agradecimiento al Prof. Elías Cueto, investigador principal del proyecto asociado a mi beca FPI (BES-2012-060744), por su atenta supervisión tanto del proceso investigador que ha permitido la elaboración de este trabajo como de la formación que he adquirido. A los tres les agradezco mucho que, sin conocerme de nada y a pesar de provenir de una rama académica *ligeramente* diferente, depositasen su confianza en mí desde el primer momento.

En segundo lugar, quiero agradecer al Prof. Francisco Chinesta y a su equipo (en particular al Dr. José Vicente Aguado y al Dr. Domenico Borzacchiello) por su valiosa orientación y excelente trato durante mi estancia en la École Centrale de Nantes.

A continuación, quiero dar las gracias a aquellos compañeros con quienes he podido intercambiar ideas y opiniones sobre la PGD, en especial a Diego Canales, Elena López, Santiago Montagud, Ángel León, Adrián Berges y Alberto Badías. A éste último, amigo además de compañero, le deseo un doctorado lleno de éxitos. También quiero agradecer a todos con quienes he compartido las salas de becarios y doctores, por haberme hecho el trabajo más ameno; y *a fortiori* a mi amigo Frederico Ribeiro, por sus intensos debates sobre la vida en general.

A Rosa Monge, por todo lo que hemos vivido y viviremos juntos.

A mis hermanos, Alejandro e Irene, porque con ellos vuelvo a estar en casa cada vez que regreso a Alicante. Y a mis padres, incansables luchadores, que tan duramente han trabajado siempre por procurarnos una buena educación y un buen futuro. Os estoy muy agradecido.

Abstract

The real-time computer-based simulation of surgery has proven to be an appealing alternative to traditional surgical simulators. Amongst other advantages, computer-based simulators provide considerable savings on time and maintenance costs, and allow trainees to practice their surgical skills in a safe environment as often as necessary. However, in spite of the current computer capabilities, computational surgery continues to be a challenging field of research. One of its major issues is the high speed at which complex problems in continuum mechanics have to be solved so that haptic interfaces can render a realistic sense of touch (generally, feedback rates of 500–1 000 Hz are required).

This thesis introduces some novel numerical methods for the interactive simulation of two usual surgical procedures: cutting and tearing of soft tissues. The common framework of the presented methods is the use of the Proper Generalised Decomposition (PGD) for the generation of computational vademecums, i. e. general meta-solutions of parametric high-dimensional problems that can be evaluated at feedback rates compatible with haptic environments.

In the case of cutting, computational vademecums are used jointly with XFEM-based techniques, and the computing workload is distributed into an off-line and an on-line stage. During the off-line stage, both a computational vademecum for any position of a load and the displacements produced by a set of cuts are pre-computed for the organ under consideration. Thus, during the on-line stage, the pre-computed results are properly combined together to obtain in real-time the response to the actions driven by the user. Concerning tearing, a computational vademecum is obtained from a parametric equation based on continuum damage mechanics. The complexity of the model is reduced by Proper Orthogonal Decomposition (POD) techniques, and the vademecum is incorporated into an explicit incremental formulation that can be viewed as a sort of time integrator.

By way of example, the cutting method is applied to the simulation of a corneal refractive surgical procedure known as radial keratotomy, whereas the tearing method focuses on the simulation of laparoscopic cholecystectomy (i. e. the removal of the gallbladder). In both cases, the implemented methods offer excellent performances in terms of feedback rates, and produce very realistic simulations from the visual and haptic point of view.

Resumen

La simulación quirúrgica por ordenador en tiempo real se ha revelado como una alternativa muy atractiva a los simuladores quirúrgicos tradicionales. Entre otras ventajas, los simuladores por ordenador consiguen ahorros importantes de tiempo y de costes de mantenimiento, y permiten que los estudiantes practiquen sus habilidades quirúrgicas en un entorno seguro tantas veces como sea necesario. Sin embargo, a pesar de las capacidades de los ordenadores actuales, la cirugía computacional sigue siendo un campo de investigación exigente. Uno de sus mayores retos es la alta velocidad a la que se tienen que resolver complejos problemas de mecánica de medios continuos para que los interfaces hápticos puedan proporcionar un sentido del tacto realista (en general, se necesitan velocidades de respuesta de 500–1 000 Hz).

Esta tesis presenta algunos métodos numéricos novedosos para la simulación interactiva de dos procedimientos quirúrgicos habituales: el corte y el rasgado (o desgarro) de tejidos blandos. El marco común de los métodos presentados es el uso de la Descomposición Propia Generalizada (PGD en inglés) para la generación de vademécums computacionales, esto es, metasoluciones generales de problemas paramétricos de altas dimensiones que se pueden evaluar a velocidades de respuesta compatibles con entornos hápticos.

En el caso del corte, los vademécums computacionales se utilizan de forma conjunta con técnicas basadas en XFEM, mientras que la carga de cálculo se distribuye entre una etapa *off-line* (previa a la ejecución interactiva) y otra *on-line* (en tiempo de ejecución). Durante la fase *off-line*, para el órgano en cuestión se precalculan tanto un vademécum computacional para cualquier posición de una carga, como los desplazamientos producidos por un conjunto de cortes. Así, durante la etapa *on-line*, los resultados precalculados se combinan de la forma más adecuada para obtener en tiempo real la respuesta a las acciones dirigidas por el usuario. En cuanto al rasgado, a partir de una ecuación paramétrica basada en mecánica del daño continuo se obtiene un vademécum computacional. La complejidad del modelo se reduce mediante técnicas de Descomposición Ortogonal Apropiada (POD en inglés), y el vademécum se incorpora a una formulación

incremental explícita que se puede interpretar como una especie de integrador temporal.

A modo de ejemplo, el método para el corte se aplica a la simulación de un procedimiento quirúrgico refractivo de la córnea conocido como queratotomía radial, mientras que el método para el rasgado se centra en la simulación de la colecistectomía laparoscópica (la extirpación de la vesícula biliar mediante laparoscopia). En ambos casos, los métodos implementados ofrecen excelentes resultados en términos de velocidades de respuesta y producen simulaciones muy realistas desde los puntos de vista visual y háptico.

Contents

Contents

List of Figures

List of Algorithms

Thesis

Chapter 1

Introduction

1.1 General overview

This thesis presents some efficient real-time approaches for the computer-based simulation of surgical procedures, involving deformation, cutting and tearing of soft tissues. The aim of this chapter is to describe, in a general manner, the usefulness of simulators for the training of future surgeons, in particular those that make use of computer-based implementations, and their underlying technologies.

1.1.1 Surgical training

Surgical training consists in acquiring a series of skills, mainly cognitive, clinical, and technical, which allow medical students to develop expertise in surgical interventions [1–4]. Traditionally, surgical training relies primarily on attending instructional lectures and seminars, followed by the guided performance of the procedures on real patients. The latter is achieved progressively, first by observing in the operating theatre, then by assisting mentors in simple procedures, and finally by operating under supervision (see Fig. 1.1). Eventually, trainees gain sufficient experience to work unsupervised. This traditional learning model, known as Halstedian training model, is essentially based on the concept of "*learning-by-doing*" [3–5]. However, gaining surgical expertise only by dedicating a large amount of working hours in the operating theatre presents a number of drawbacks.

First of all, gaining experience in the operating theatre depends largely on the random flow of patients and their unpredictable clinical cases. This causes a significant variability in apprenticeship and makes an equitable learning of

Figure 1.1. The painting *The Agnew Clinic* (detail) by Thomas Eakins (1889) shows very accurately how traditional surgical training was conducted in an anatomical theatre of the nineteenth century.

all trainees difficult [1, 4, 6]. Secondly, students are increasingly required to know a greater number of technical and operational skills, not only related to the most recent surgical techniques, but also to their former approaches; e. g. both modern minimally-invasive procedures and traditional open surgery [2, 7]. Time, though, is the major limitation in this regard [4]. This situation, combined with the increasing costs of health care, suggests that efficiency is becoming an essential requirement of training [3, 7–9].

Another significant concern about surgical training relates to patient safety. A balance between instruction and patient care should exist when performing a guided surgical intervention, since one of those responsibilities is fulfilled at the expense of the other. In this sense, several studies have revealed that the rate of complications at operative performance is higher amongst junior surgeons than their senior counterparts [7]. Consequently, skill practice sessions with actual patients have raised ethical, moral, and medicolegal concerns, and have even been considered as no longer acceptable [4, 6–8].

One of the areas that can face these issues and improve the quality of surgical education is simulation. As described in the next section, surgical simulators provide a low-risk environment to train surgical procedures continually, as well as a reduction of the instructional time, away from the operating theatre [2, 10].

1.1.2 Surgical simulation

Simulation is a technique by which the essential features of a real-life process or system are imitated by another process or system [4, 5]. For years, the use of simulators has been extensive in air and space flight, and significant in other fields such as defence, mining, urban planning, nuclear energy generation, system maintenance, and also medicine [7, 11, 12]. Within medicine, several methods of simulation have been used for different purposes, including education, research, health system integration, pre-operative planning and assessment, and skills training [5, 12] —the latter being the most relevant one for the scope of this thesis. Thus, given their ability to reproduce situations that resemble reality, simulation can be successfully used in medical environments as an instructional strategy to teach technical skills and procedures. Such is the particular case of **surgical simulation**, which allows students of surgery to practise their skills before applying them to real patients [2].

There are two key features that make surgical simulation more efficient than traditional training methods. On the one hand, simulation provides trainees a low-risk environment that does not compromise the care and safety of patients. On the other hand, simulation eliminates most external factors that generate variability in learning. This last point is particularly significant, since reducing interference to a minimum brings several advantages. For example, the training limitations produced by both the availability of patients and the rarity of their clinical cases can be prevented by the ability of simulators to recreate several different surgical scenarios. All students can train the same cases, which ensures equal learning opportunities, and the level of difficulty can be readjusted as proficiency increases. In addition, training schedules can be set in advance to ensure compliance with the period of instruction. Thus, surgical simulation facilitates a more productive and standardised training [6, 8].

1.1.2.1 General classification of surgical simulators

There is a wide range of surgical simulation devices, although they differ from each other by their degree of fidelity or realism with respect to real patients. Surgical simulators extend from boxes to practise simple processes using common inexpensive materials, e. g. cutting circles out of a piece of gauze or dropping beans into a small pot; to complex high-technology equipment, e. g. physiologic human mannequins or haptic computer-based simulators [5, 9, 13]. A classification of the most common simulation methods is presented below.

Physical models. This kind of simulators includes models made of rubber or plastic which imitate the human anatomy and (depending on their purpose) also its physiology. These models represent different body parts and pathologies, allowing trainees to manipulate their synthetic tissues by means of real surgical instruments. Although physical models help develop the skills necessary for particular tasks, they present certain issues. For instance, they must be re-equipped or replaced after several uses, which implies a commitment of time and money; and in some cases, their realism is rather limited [2].

Box trainers. Also known as **endotrainers**, these simulators allow to train basic tasks of laparoscopy. They consist of a cavity with access points through which real laparoscopic instruments can be used. Different items, ranging from simple beans to synthetic models of human organs, can be placed inside to perform tasks such as grasping and moving small objects, tying knots, or cutting patterns. More advanced systems can incorporate a video camera and a monitor to track the processes —in that case, they are called **video-trainers** [7, 14] (Fig. 1.2). In general, box (or video) trainers are sufficient for acquiring the most essential laparoscopic skills, but fail to simulate the realism and complexity of an actual surgical intervention [15].

Live animals models. Anaesthetised animals have been extensively used before the development of other simulators, particularly for training both open and laparoscopic surgery. Live animals provide a realistic working environment. Mostly pigs, but also dogs and sheep, are used because of the similarity of their anatomical structures with the human ones. Live animal model training, however, requires access to animal facilities and continuous monitoring, which have a direct impact on costs. In addition, the use of animals for this purpose has been limited in several countries, and may create legal, ethical and cultural issues [2, 4, 7].

Ex vivo animal tissue models. This type of training models uses tissues or anatomical sections from dead animals to train a broad range of surgical skills, such as making incisions or suturing. These tissues can be attached to a frame arranged to imitate the human anatomy, and may provide a similar tactile sensation. These simulators are relatively inexpensive, easily available, disposable, and can be frozen for later use. However, their reuse is limited to a certain number of times, and training requires dedicated installations [1, 2, 13].

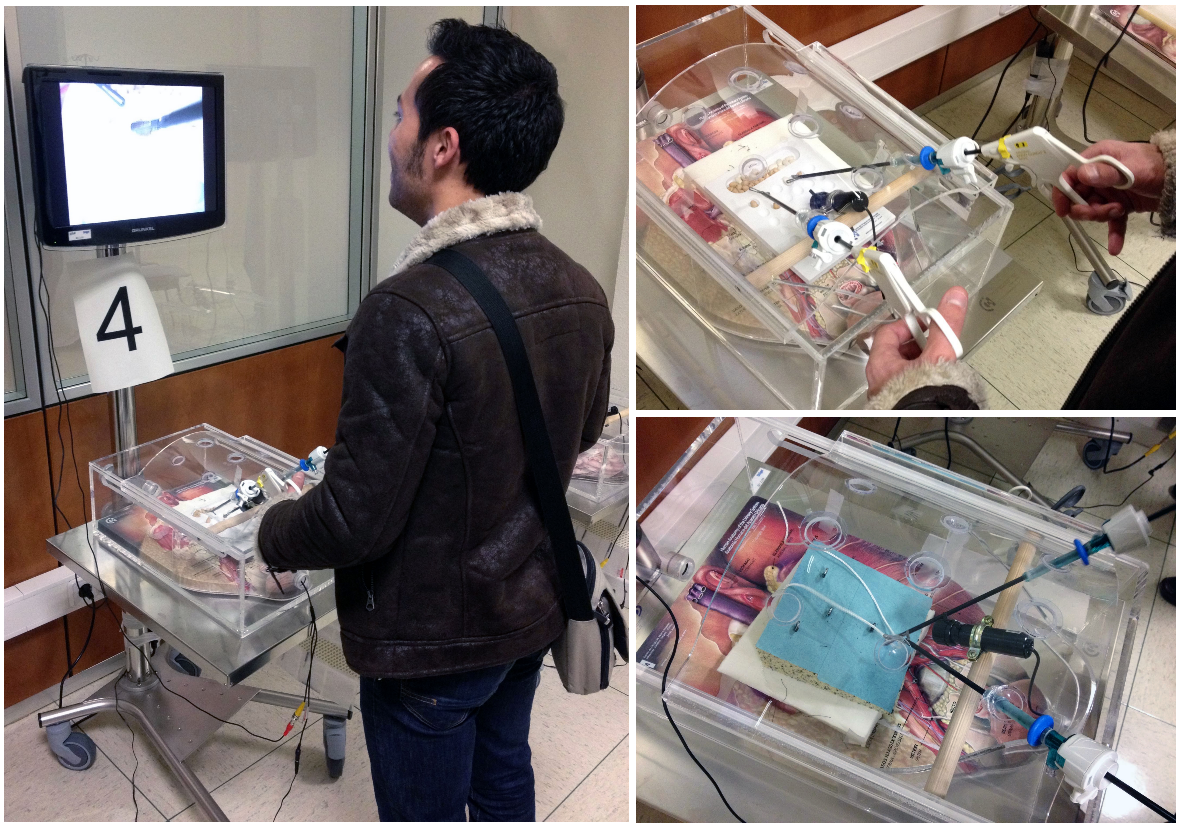


Figure 1.2. Actual hand-made video-trainers developed and used at Hospital Royo Villanova, in Zaragoza. These simulators allow to train basic skills by performing simple tasks with laparoscopic instruments, such as grasping, moving, and dropping chickpeas in an orderly manner (left and top right), or passing a string through a set of eyebolts attached to a pad (bottom right).

Cadaveric models. The use of human cadavers for a detailed understanding of anatomy and interaction amongst body parts dates back to the sixteenth century. In addition to anatomical dissection, cadavers can be used for training several procedures, including endoscopy and laparoscopy. Nevertheless, a number of reasons suggests the reduction or elimination of cadaver use in medical education: there may be unwanted changes in anatomy (e. g. loss of tissue elasticity) that alter their fidelity as a simulator, maintenance is expensive and time-consuming, and a potential health hazard exists. Additionally, cadavers are single-use, non-portable, and may give rise to ethical or cultural concerns [2, 4, 13].

Computer-based models. These simulators, which have attracted great interest in recent years, allow a real-time interaction between surgical trainees and computer generated images of organs. Section 1.1.3 is devoted to them.

Hybrid simulators. The combination of physical and computer-based models leads to hybrid simulators. They consist of mannequins (or other physical models) connected to a computer that simulates the patient responses to a serie s of procedures. They are not intended to replace basic skills training. Instead, they provide a realistic teamwork environment to practise different clinical scenarios (e. g. crisis management, team response, communication). They are expensive and require long setup times [2, 9, 16, 17].

1.1.2.2 Skills transfer

There is a rich debate about whether the skills acquired in simulated environments are directly transferable to the clinical setting. One of the main causes of discussion lies in the difficulty to establish unified objective criteria for assessing the transfer outcomes of all the analysed scenarios, which are very varied [2, 3, 18]. However, in spite of these issues, the vast majority of studies that compare simulation-trained groups versus untrained groups conclude that the former achieve significant improvements in almost all the measured parameters, regardless of the training methods and training duration [2, 9]. A complete systematic review on skills transfer can be found in Sturm et al. (2007) [2], which points in this direction. Other analyses with similar conclusions are also conducted in Aucar et al. (2005) [1], Sutherland et al. (2006) [3], and Dawe et al. (2014) [19, 20], to mention a few.

1.1.3 Computer-based simulation

Computer-based simulation can be generally defined as the reproduction of some aspects of reality by means of a computerised environment, in which the user can interact through a human-computer interface [4, 10]. The expressions "computer-based simulation" and "**virtual reality**" are often used interchangeably by the consulted literature to refer to the same concept [1, 10–12, 21]. While it is true that the early stages of virtual reality bring back to mind head-mounted displays, wired gloves, and other wearable devices, recent definitions of the term include any interactive three-dimensional environment that makes use of visual and haptic (touch and force) technologies [10].

In contrast with virtual reality, whose environment is completely computerised, **augmented reality** integrates some virtual objects into a real-world environment. Although augmented reality has been successfully incorporated into several fields of medicine, including simulation, it does not fall within the scope of this thesis. A good review about the medical applications of augmented reality can be found in [12].

Very recently, some similar terms have been coined to define related concepts. For example, "**computational surgery**" describes the combination of sciences and technologies (such as mathematics, algorithm design, imaging, robotics, and computing, amongst others) which incorporate biological and physical principles to improve surgical processes [12, 22]. In this case, computational surgery is a general term which encompasses computer-based simulation of surgery. Another example is the concept of "**virtual surgery**", which particularises the computer-based simulation of surgery to the planning of procedures based on patient-specific models [12]. As it may be inferred, in addition to the latter, there are several areas of medicine in which computer-based simulation can be applied; the most relevant ones are mentioned below.

1.1.3.1 Areas of application

In the field of medicine, there are three broad areas in which the computer-based simulation can be applied: **education and teaching**, for the learning of anatomy and surgical procedures; **medical planning and training**, for the development of psycho-motor skills, the diagnosis of patients based on their clinical data, and the pre-operative planning of surgery; and **virtual prototyping**, for the development and testing of medical equipment and instruments [12, 21].

Amongst these areas of application, medical training is the most relevant one for the scope of this thesis. When computer-based simulation is applied to the training of surgical skills, the virtual environment consists of realistic three-dimensional computer-generated models of human organs that replicate the biomechanics of living soft tissues. The interactive manipulation of the models is achieved through a physical interface that emulates the instruments used in clinical practice, and provides the user with haptic feedback [4, 10, 12].

As pointed out above, medical planning is closely linked to medical training through **computer-integrated surgery** (CIS) systems, i. e. those that use the computer technology for surgical planning, and for guiding or performing surgical interventions [23, 24]. Indeed, recent fields of study in this area seek methods to generate computational biomechanics models for patient-specific applications, which allow to plan a surgical intervention on a *patient avatar* instead of on a generic model [25]. A review about patient-specific medical planning can be found in [26].

1.1.3.2 Classification by generation

In the early 1990s, the surgeon Richard Satava conducted a series of clinical trials for the U.S. Department of Defense in which compared traditional versus virtual reality surgical training [8]. Based on his findings, Satava suggested a **taxonomy** to classify computer-based surgical simulators depending on their realism and complexity [21]. He established five groups or **generations**, each representing a technological leap forward.

The **first** generation includes the simulators that can display accurate representations of the organs at a macroscopic level, only focusing on the three-dimensional geometry of the anatomic structures. These were the only simulators to be developed at the early stages of computer-based surgical simulation. Because the user interaction is restricted to a mere virtual navigation around the organs, simulators belonging to the first generation are simply used as complementary educational tools.

The **second** generation also incorporates the simulation of physical properties of the organs. This requires the addition into the models of certain responses to external interactions, such as the deformation of the organs when performing palpation or the modification of their topology when performing cutting. Most of the real-time physics-based simulators that are currently under development belong to this generation.

The **third** level in this taxonomy adds physiological features to the models, such as the release of body fluids or the bleeding of wounds. Due to the complexity of coupling physics and physiology, only a limited number of prototypes incorporates some of these features. The **fourth** generation includes the representation of the anatomy at a microscopic level (e. g. microglandular, neurovascular structures); whereas the **fifth** generation also models the biochemical systems (e. g. immunologic, endocrine systems) [15, 27].

1.1.3.3 Architecture of simulators

The architecture of a second generation computer-based surgical simulator was described a decade ago by Delingette and Ayache [15]. Although the description was particularised for minimally-invasive procedures, it is also valid for any surgical skills training simulator. Broadly speaking, a simulator can be divided into three main components: the **input devices**, the **output devices**, and the **core** of the simulator.

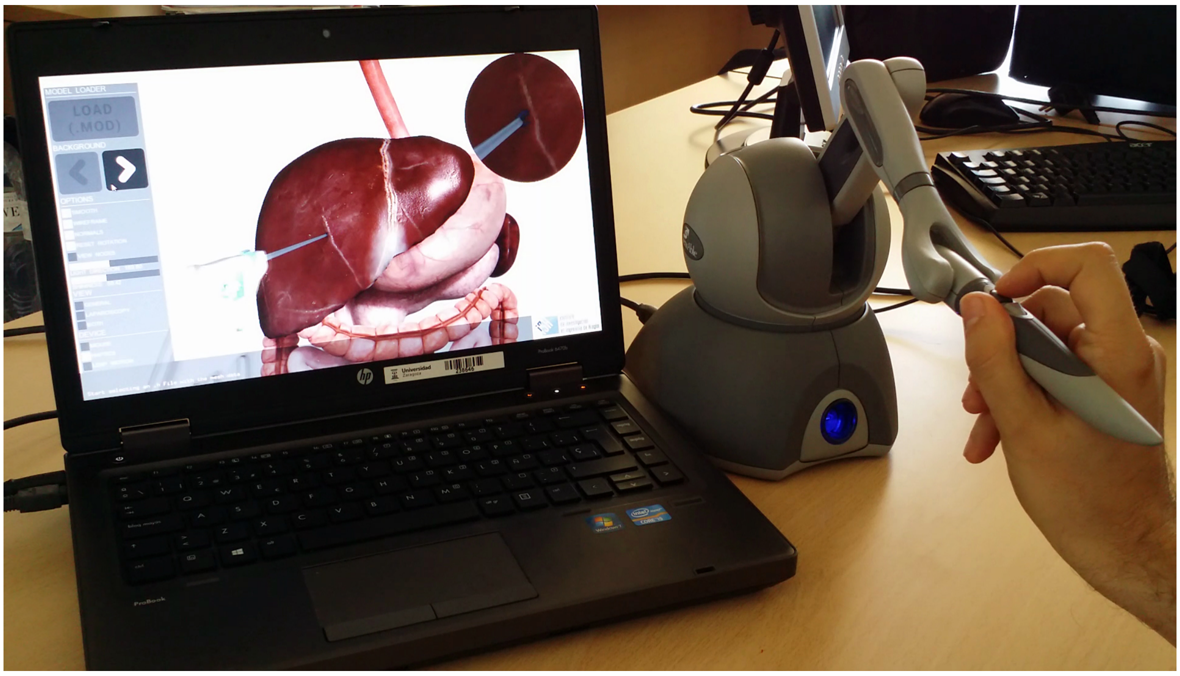


Figure 1.3. Appearance of the computer-based surgical simulator developed at Universidad de Zaragoza, consisting of a laptop (left), and the stylus-based haptic interface *Phantom Omni* by *GeoMagic SensAble* (right).

Input devices. Input devices refer to the mechanical systems (hardware) that allow to move and operate the virtual surgical instruments within the simulated environments. Keyboards and mice can be used as input devices to interact roughly with the scenario. However, advanced input/output peripherals (i. e. those that perform input and output functionality simultaneously) are more frequent in this kind of simulators.

Such is the case of haptic interfaces, which incorporate both sensors to measure the inputs and actuators (motors) to generate the outputs. The inputs may include the position and orientation of the device, and the forces exerted by the user; whereas the outputs recreate the sense of touch by applying vibrations, forces, or motions to the user. There exist several classes of haptic interfaces, ranging from simple commercial desktop devices to complex prototype exoskeletons. Commonly, haptic interfaces used in surgical simulation are either devices which allow to attach specific tools (e. g. laparoscopic instruments) or stylus-based devices. The latter consist of an articulated arm, with a pen-shaped pointing tool attached on its end, which can be grabbed and moved around as if it were the handle of any surgical instrument (see Fig. 1.3) [15, 28, 29].

Output devices. A surgical simulator must provide visual and haptic feedback through their output interfaces. **Visual interfaces**, ranging from basic screens to glasses for stereoscopic vision, are key components in computer-based surgical simulators, since they allow trainees to maintain eye contact with the virtual environment. In order to achieve a realistic visual feedback, response frequency must be in the range of 20–60 Hz [15]. With regard to **haptic interfaces**, the inclusion of force and tactile feedback in surgical simulators improves the virtual immersion of trainees. Thus, force feedback generates forces that can be large enough to stop the motion of the user, for instance when colliding with a virtual object, whereas tactile feedback produces pressures and vibrations that allow to feel the texture of a virtual surface [28]. The response frequency to obtain an acceptable haptic display is in the range of $500–1,000$ Hz, depending on the nature of the scene: slow or very soft objects, for instance, do not require high update rates [15].

Simulator core. In essence, the main objectives of the core of a surgical simulator are to perform a visual and haptic rendering of the models, and to process the interactions between virtual tools and virtual organs.

The input data provided by the sensors of the haptic interface are processed by the simulator, which calculates a collection of variables such as the position, the orientation, and the velocity of the virtual tool with respect to the virtual organs. Collision detection algorithms determine whether any interaction has occurred amongst the virtual objects [30]. When a collision is detected, a series of physically-based algorithms are applied to the virtual organs to simulate their behaviour, which depends on the type of interaction that the virtual tool produces. Some of the possible interactions to be modelled in surgical simulation are the deformation of soft tissues, cutting using blades or scissors, needle insertion, suturing, or tearing by electrocautery instruments, each one involving different degrees of complexity [15, 31, 32]. Recent strategies used to simulate some of these interactions are described throughout § 1.2.

After processing the results of the interactions, the simulator returns the visual and haptic feedback to the output interfaces. In practise, the high haptic feedback rates required to obtain a realistic sense of touch become an important bottleneck in this kind of real-time simulators, since an *instance* of the problem has to be solved every 1–2 milliseconds ($500–1,000$ Hz), regardless its complexity [15, 27].

1.2 State of the art

There exists a vast amount of literature regarding computer-based surgical simulation, which is distributed in several fields, encompassing computational biomechanics, computer graphics, virtual reality and robotics. To be in consonance with the scope of this thesis, the present review is solely focused on the computational methods that model the biomechanics of living soft tissues and their interactions with surgical instruments.

Certain related topics, such as (*i*) the characterisation of soft tissues, the measurement of their physical properties, and the material models that describe their behaviour; (*ii*) the specific healthcare disciplines on which the simulators are focused [12, 33]; (*iii*) the commercial simulators available on the market [12]; or (*iv*) the evaluation of the skills acquired during training sessions with simulators (which was outlined in § 1.1.2.2), are not covered in this review. Interested readers in any of the aforementioned topics should consult the suggested references.

This review is therefore not intended to be exhaustive, but provide an overall picture of the biomechanical models and computational methods used to simulate surgical procedures. The organisation of the review mainly follows a generational order, which is directly related to the classification established by Satava (§ 1.1.3.2), and distinguishes amongst the different types of interactions.

1.2.1 First generation

Satava only includes in this generation the geometric anatomy of realistic organs, referring strictly to the three-dimensional shapes of the models. Cueto and Chinesta [27], however, extend this definition to include also the models that, in response to external non-invasive interactions, exhibit simplistic or unnatural deformations: i. e. those that are not based on physics, are two-dimensional, or are not computed in real-time. Indeed, from their perspective, approaches whose deformations are based on simple constitutive models (such as linear elasticity) also belong to this category, since they are unable to produce realistic sensations. This idea is to some extent supported by Lim et al. [34], who consider that surgical simulators that do not incorporate the correct mechanics of soft tissues are rather "glorified video games with limited training value."

First computer-based surgical simulators were technologically limited, and presented their case studies using only text and sometimes static images. Basic interactivity and image representations of rigid models were not introduced until the late eighties [35]. As for the development of deformable models, initial approaches were more focused on generating visually acceptable results using *ad hoc* methods rather than on implementing models based on physics, far too complex to be achieved in real-time with the existing computing resources [27, 36]. The first deformable models that incorporated the physical properties of linear elasticity theory were developed by Terzopoulos et al. [37] in 1987, although they were not computed interactively. Since then, many other approaches were proposed to include the laws of continuum mechanics into computer models.

Because of their unrealistic behaviour or off-line performance, the first generation of simulators are of little interest, and all their functions have been superseded by the simulators of the second generation. There exist some reviews that summarise and classify the simulators that belong to the first generation, such as those of Gibson and Mirtich (1997) [38], Delingette and Ayache (2004) [15], Meier et al. (2005) [36], or Nealen et al. (2006) [39].

1.2.2 Second generation

Satava includes within this generation those models that incorporate, in addition to visual realism, physical properties. Amongst them are the deformations caused by virtual palpation and any invasive tool-tissue interaction, such as puncture, cutting, tearing, and other kinds of rupture of tissues. As discussed previously, linear elastic models are too poor to describe faithfully the behaviour of soft tissues and are not acceptable to be included in this generation. Thus, only models that consider at least geometric or material non-linearities are taken into account.

The interactions with the virtual organs have to be computed at runtime. This is a critical constraint in surgical simulation, and the chosen strategy depends in each case on the computer efficiency, the required accuracy, and the kind of manipulation to perform in the virtual organs [40]. In this review only real-time (or interactive) approaches are included.

A distinction between **invasive** and **non-invasive interactions** is made in some of the consulted references, depending on whether they involve tissue rupture (e. g. cutting, tearing, needle insertion, suturing) or not (e. g. deformation, collision) [32, 41]. A description of the surgical interactions most commonly found in the literature is given in the following sections. Emphasis is placed on the most relevant interactions for this thesis, namely **deformation** (§ 1.2.2.1), **cutting** (§ 1.2.2.2), and **tearing** (§ 1.2.2.3), although a section is also dedicated to other relevant procedures (§ 1.2.2.4).

Some recent reviews regarding the second generation of surgical simulators (mostly dealing only with deformations) can be found in Barbič and James (2005) [42], Misra et al. (2008) [32], Peterlík et al. (2010) [43], San Vicente (2011) [44], Filipovič (2012) [45], Cueto and Chinesta (2014) [27], Li (2014) [41], Horton (2015) [46], Jichuan (2015) [47], and Nisansala et al. (2015) [48]. Other reviews, particularly interesting with regard to the simulation of cutting, can be found in Jerábková (2007) [40], and Wu et al. (2014) [49].

1.2.2.1 Deformation

Deformation is the transformation that a model suffers when it is subjected to external loads or body forces. In the context of surgical simulation, it is important to perform a convincing simulation of deformations because any medical procedure, no matter how simple it is, involves the application of external loads.

In this section, two broad sets of methods for modelling deformations are analysed: **mesh-based methods** and **meshfree methods**. Generally speaking, they differ from each other in that the former require connection amongst the nodes of the domain (a *mesh*), whereas the latter not. Within the mesh-based methods, the numerical technique that predominates is the Finite Element Method (FEM); consequently, no other numerical technique is addressed in this section. The most relevant strategies followed by each of these two methods are described below.

Mesh-based methods. The inaccuracies that linear FEM exhibits in large deformation problems produce unrealistic representations in deformable models (Fig. 1.4). To obtain results similar to the real behaviour of the living soft tissues, non-linear FEM approaches are then considered. Two sources of non-linearities are usually described in this case: the geometry and the material properties of the model. First, geometric non-linearities establish a non-linear relation between displacement and strain, which allows a preservation of the volume of the models when they undergo large deformations, and results in a more realistic behaviour. Second, material non-linearities appear when non-linear relations between stress and strain are introduced. This class of non-linearities modify the force response of the model, and are relevant for achieving a realistic simulation of living soft tissues. A number of constitutive models exist which allow to characterise these non-linearities.

Hyperelastic models, for instance, obtain their stress-strain relation from a strain energy density function. Saint Venant-Kirchhoff (SVK) model is the simplest hyperelastic model, since it is just a **geometrically non-linear** extension

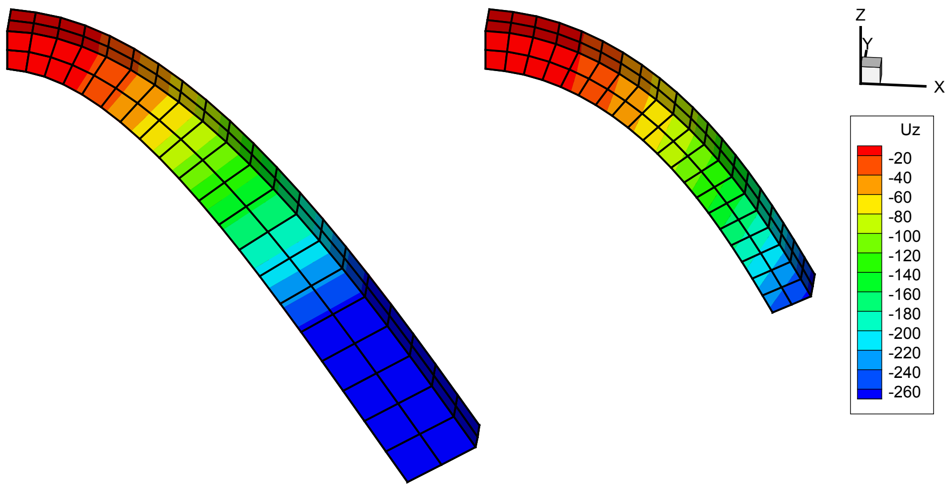


Figure 1.4. Cantilever beam undergoing large deflections modelled using linear elasticity laws (left) and geometrically non-linear strain measures (right). Note that the scale in the displacement legend is the same for both beams. It is noteworthy the apparent gain in volume when linear strain measures are used [27].

of the linear elastic material model. One of the first successful implementations of a SVK model was achieved by Zhuang and Canny in 1999 [50, 51]. They used a quadratic strain tensor to manage the large rigid body motions correctly and applied the explicit Newmark integration scheme to reformulate the non-linear system into linear equations. Other authors developed different real-time implementations of this model subsequently [52–54].

An interactive model with both **geometric and material non-linearities** was introduced for the first time in the paper by Wu et al. [55] in 2001. Neo-hookean and Mooney-Rivlin constitutive models were implemented by means of an explicit integration scheme and a novel multi-resolution method that was named *dynamic progressive meshing*. **Multi-resolution methods** were first proposed in [56] for the real-time deformation of linear elastic models. These methods reduce the number of unnecessary computations by using a coarse mesh in most part of the virtual object while keeping a good accuracy by using finer meshes only in certain areas of interest (e. g. the deformation areas). In the particular case of dynamic progressive meshing, a hierarchical mesh structure is computed off-line, and the convenience of refining the mesh locally is determined on-line by an error estimator. Other real-time approaches using multi-resolution methods can be found in [57–61].

Another strategy based on a multi-resolution method was introduced by Müller and Gross [62] in 2004. This approach is known as the *assumed-shape method*, and consists in approximating a geometrically complex model by a much simpler model (the assumed shape), whose analytical solution can be solved beforehand. At runtime, the surface of the complex model deforms according to the displacement field of the assumed shape, thus improving the computational efficiency. The authors claim that several material behaviours, including elasto-plasticity, can be realistically simulated in real-time, although no error measures are provided. The assumed-shape method has also been subsequently implemented in [63] and in some physics engines [64].

In 2007, Miller et al. developed in 2007 the **total Lagrangian explicit dynamics** (TLED) algorithm for non-linear real-time models [65]. Until that moment, the great majority of FEM implementations used the updated Lagrangian formulation, in which all variables refer to the current configuration of the system. In contrast to this formulation, all the FEM equations in the TLED algorithm are expressed in reference to the original configuration of the system (for further information on kinematic descriptions, see [66]), which translates into a gain of computing efficiency. TLED algorithm enabled the first implementation of a non-linear FEM solver in a graphic processing unit (GPU) [67]. After that, similar approaches were developed for physics engines and surgical simulators [68–70]. In general, fast responses are obtained using these approaches in relatively small-sized models, but still insufficient for haptic feedback compatibility.

Additionally, an interesting approach based on neural networks was presented in [71] in 2009. A commercial software is used to pre-compute offline a set of solutions for the model of a particular organ. Then, the data are fed into a **radial basis neural network** which is associated to the nodes of a finite element mesh. During the online simulation, the neural network is able to reconstruct both the deformation fields and the reaction forces as the surgical tool interacts with the model.

However, strategies dealing with **model order reduction** (MOR) methods deserve special mention. In a general sense, the main objective of MOR methods is to reduce the computational complexity of a numerical model (in terms of dimensions or degrees of freedom), so that it can be computed in much less time. In exchange, a loss of accuracy of the solution is obtained. In the particular field of computational mechanics, FEM-based MOR techniques seek an approximation of the nodal displacements by projecting the solution on a reduced set of global basis functions (the so-called *reduced bases*), which are specific for each model. FEM-based MOR techniques merit particular attention in this review since they fall within the scope of this thesis.

The first MOR strategy for the visualisation of the deformation of a non-linear model was proposed by Barbič and James [42] in 2005. The method exploits the fact that, for a SVK model, the basis can be estimated as a set of cubic and quadratic polynomials, whose coefficients are constant for each given geometry, and can therefore be pre-computed off-line. The method achieves a good real-time performance and, in some simple models, haptic feedback rates are obtained.

An interesting alternative for generating suitable low-dimensional bases is the principal component analysis (PCA), better known in the field of mechanical engineering as **proper orthogonal decomposition** (POD) [72, 73]. This method extracts the dominant components (*modes*) from a set of observations (*snapshots*) of the full model, and then uses those modes to form the reduced basis of the system. The number of modes included in the basis is directly related to the accuracy of the solution, but also inversely related to the computational efficiency. Several approaches focused on real-time non-linear deformable models have been recently developed using POD techniques. For example, in [74, 75] the projection onto the reduced basis eliminates the higher frequency content of the model response, which allows to use larger integration time steps in their explicit FEM algorithm. A different strategy is followed in [76–78], where several authors apply the reduced basis that is optimal for a particular problem in other scenarios with similar characteristics. Lastly, asymptotic numerical methods (ANM) [79, 80] were used together with POD in [81–83] for the simulation of hyperelastic soft tissues. In this case, the asymptotic expansion of the variables of interest helped solve the problem as a set of linear equations.

POD techniques and related approaches are usually classified as *a posteriori* MOR methods, since they require to know in advance several observations of the full model to compute the best reduced basis for the problem. In contrast, the concept of *a priori* MOR methods also exists, which can provide suitable reduced bases without knowing beforehand any observation of the problem. Such is the case of the **proper generalised decomposition** (PGD) [84, 85], which belongs to the latter class of methods, and is reviewed in detail in chapter 2. One of the most important features of PGD is its ability to generate the so-called *computational vademecums* [86], i. e. high-dimensional solutions of parametric problems, for every possible value of the parameters, which are very efficiently stored and accessed. An off-line pre-computed computational vademecum can be used on-line under strong real-time constraints, reaching high operating frequencies, which makes them compatible with the haptic feedback demanded by surgical simulators [87, 88]. Approaches based on explicit formulations of PGD [89], and its use in combination with ANM [90] have been proposed to deal with the non-linearities of soft-tissues. For these reasons, PGD emerges as an excellent alternative for the real-time simulation of surgical interactions.

Meshfree methods. Models that are simulated using meshfree methods (also known as *meshless methods*) are represented by a collection of particles with no fixed neighbourhood relationship. From a computational point of view, meshfree methods are more demanding than FEM, since they need to compute the node-to-node adjacency in every simulation step. Consequently, meshfree methods are mainly used to simulate those phenomena in which either fixed meshes are very restrictive or the connectivity of the nodes is difficult to maintain without introducing errors, e. g. complex physical effects that combine fluids and solids, such as melting or solidifying. [40, 49]. Most of the recent meshfree approaches that simulate deformations belong to the first generation of simulators, either because the interactions are not computed in real-time or because they are based on linear models. However, there are approaches, such as the following ones, that fall into the second generation.

Mass-spring models (MSM) are some of the most common meshfree methods used for simulating soft tissues. These models simulate the objects as a collection of point masses (*nodes*), which are linked together by means of ideal weightless springs, and arranged in a lattice structure. Although mass-spring models were techniques not initially based on continuum mechanics for which the stiffness of the springs had to be determined experimentally [32, 40], some physics-based approaches have been developed in recent years. A non-linear (biquadratic) MSM that reproduces the SVK constitutive model is described in [91]; and a cubical MSM for approximating living soft tissues is presented in [44].

The **point collocation-based method of finite spheres** (PCMFS) is an extension of the finite spheres method (FSM). It discretises the domain using particles with finite spherical influence zones [92, 93]. It has been applied to real-time simulation of surgery in [94, 95], and has also been successfully combined with POD techniques to simulate the non-linear hyperelastic response of soft tissues [96]. Another method developed by the same authors is the **point-associated finite field** (PAFF) [97, 98], which discretises the domain using scattered points that are interpolated by the least-squares method. PAFF is considered an extension of PCMFS that allows the development of multi-resolution strategies.

1.2.2.2 Cutting

Surgical procedures involve, by definition, the performance of incisions using medical instruments. The realistic simulation of surgical cuts in soft tissues is a

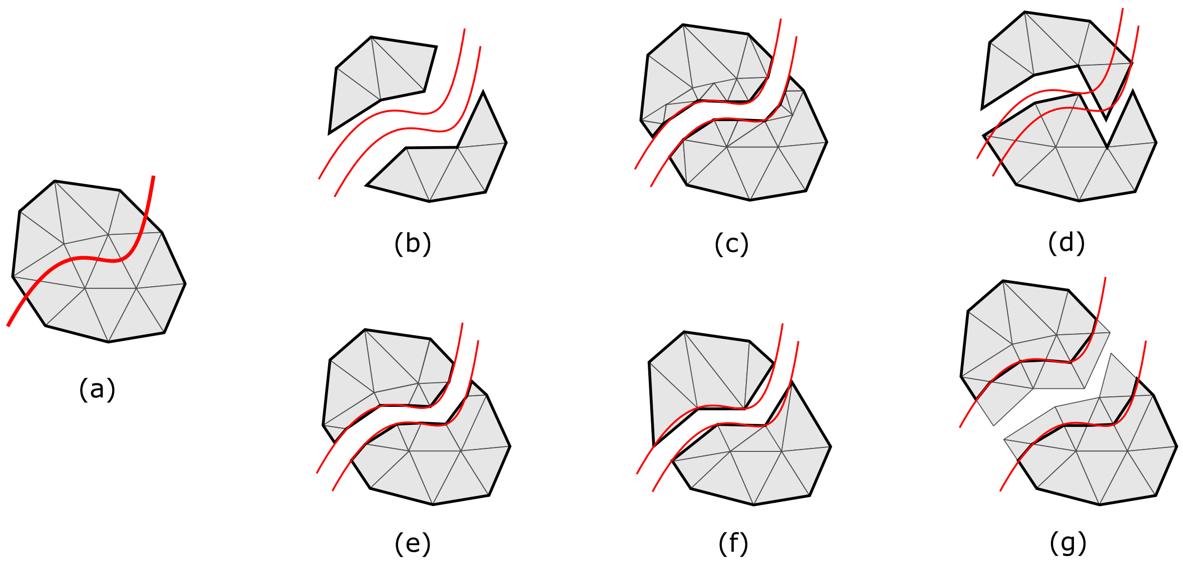


Figure 1.5. Different strategies for incorporating cuts into a mesh: (a) original cutting configuration; (b) deletion of elements; (c) refinement of elements; (d) splitting along existing faces; (e) splitting using polyhedral elements; (f) snapping of vertices; (g) duplication of elements. The red line in (a) indicates the cutting trajectory, which separates the models in (b)–(g) into two disconnected parts. The surfaces of the models are marked with a bold black line. Figure adapted from [49].

significant source of difficulties in real-time modelling, since it requires to modify the geometry and topology of the domain and its associated mesh (if any), without penalising the computation time. The review in this section broadly classifies the most common strategies to simulate cutting in deformable bodies, rather than describing particular approaches. These strategies are mainly focused on mesh-based techniques (FEM particularly), although some representative meshfree techniques are also mentioned in the last paragraph.

Deletion of elements. One of the simplest strategies to include cuts in a deformable body consists in removing the elements that have been touched by the virtual cutting tool (Fig. 1.5 b). However, this method has significant disadvantages. On the one hand, the new exposed surface is jagged, contains unpleasant visual artefacts, and does not conform to the smooth surface made by a cutting tool. On the other hand, it produces a loss of volume and violates the mass preservation law, leading to an unrealistic behaviour of the model. The deletion of elements in volumetric graphical objects is described in [99], and some approaches have been developed for tetrahedral elements [100] and hexahedral elements [101].

Refinement of elements. To adapt accurately to the cuts, the elements of the model can be locally refined or re-meshed (Fig. 1.5 c). Bielser et al. [102] introduced a subdivision method to split tetrahedral elements affected by the cuts according to a series of predefined templates. Each of these templates establishes which vertices have to be duplicated to create the new topological configuration, and can be precomputed in advance. These methods have been subsequently extended and improved using different approaches [103–107]. The main drawback of these methods is that it is possible to generate elements with a volume close to zero. This may lead to ill-shaped elements (known as *slivers*) that cause numerical instability to the system [40, 49]. Some recent approaches, though, provide regular hexahedral discretisations which effectively remove the possibility of generating such ill-shaped elements [101, 108].

Splitting along existing faces. This strategy consists in splitting the object along existing element faces (Fig. 1.5 d). It provides satisfactory results if the surfaces to be cut are known before establishing the initial discretisation. However, if cuts are performed arbitrarily, it may result in a jagged surface. Some examples can be found in [109–111], which are sometimes combined with the strategy of vertex snapping (see below).

Splitting using polyhedral elements. When models are discretised using tetrahedral or hexahedral elements, and cuts are performed by element refinement, the resulting elements have to be also tetrahedra or hexahedra. Polyhedral discretisations remove this constraint and allow the modelling of cuts by splitting an element into disconnected parts (Fig. 1.5 e). No re-meshing is needed to decompose the elements into smaller polyhedra. However, to avoid numerical problems, the new elements have to be forced to be convex, and the possibility of generating ill-shaped elements remains. Although this issues make this strategy non-trivial, some approaches can be found in [112, 113].

Snapping of vertices. A strategy that does not require the creation of new elements consists in **snapping** the nodes of the nearby existing elements onto the surface of the cut (Fig. 1.5 f). Nienhuys [114] used this idea as a previous step to split the faces of the elements that are located along the cut edge. However, this method may also give rise to degenerated elements which need further processing [40, 49]. To solve this, Steinemann et al. [115] combined vertex snapping with an algorithm to refine the mesh in case it is needed.

Duplication of elements. To avoid the numerical problems of the ill-shaped elements, Molino et al. [116] introduced the *virtual node algorithm*. The key idea is to create several **replicas** of the cut element and distribute portions of its material connectivity amongst each replica (Fig. 1.5 g). This results in new elements that contain both material connectivity components and empty regions. Thus, the volumetric and surface representations are topologically consistent. The main limitation is that this method is more suitable for off-line simulation rather than for real-time interaction and, consequently, it is not compatible with the second generation of surgical simulators. Several improvements of this approach were made in [117], although still not sufficient for on-line operation.

XFEM techniques. A recent strategy makes use of the **extended finite element method** (XFEM) [118]. This method *extends* the classical FEM by enriching locally the solution of the model with discontinuous functions. XFEM presents a series of advantages with respect to the rest of strategies. First, XFEM represents the cut with more accuracy than techniques based on re-meshing. Second, although additional vertices are added to the system as the cut is being performed, the original mesh remains unaltered. And third, since no new elements are created, the impact on the performance of the simulation is minimised and the possibility of generating ill-shaped elements is eliminated [40, 41].

The first reported use of XFEM in surgical simulation is described in Vigneron et al. (2004) [119], although they considered a static two-dimensional model with small deformations. A real-time three-dimensional approach was developed by Jeřábková and Kuhlen [120], from which other works derived [101, 121]. Lastly, Niroomandi et al. [83] combined XFEM with POD techniques to obtain a reduced system from some initial off-line simulations which can be solved efficiently in real-time.

Meshfree approaches. Certain approaches also exist to simulate cutting in meshfree models. Some of them are based on mass-spring models, in which the spring networks are progressively subdivided and re-meshed as the cut advances [122]. Other hybrid strategies, instead, combine meshfree models with some kind of mesh-based structures [123, 124]. Lastly, it is also worth mentioning a recent algorithm based on clouds of points, the so-called MTLADR [125]. Although only two-dimensional examples are reported using this algorithm, their authors assert that three-dimensional implementations can be also developed.

1.2.2.3 Tearing

Some surgical interventions require the tearing of certain soft tissues such as the lens capsule in cataract surgery and the adipose tissue around the liver in cholecystectomy. As in the case of cutting, tearing is a challenging invasive interaction whose simulation requires the real-time modification of the domain of the model.

There is no much literature that addresses the interactive simulation of tearing of soft tissues. Earliest approaches date back from the year 2000 and were not visually realistic [126]. Cotin et al. [100] proposed a real-time hybrid method, using FEM and a tensor-mass model, in which cutting and tearing of soft tissues were simulated as two similar phenomena by deleting the affected elements. In recent approaches, particular attention is paid to the propagation of the tear, which depends on the direction of the applied forces. These approaches, though, are mainly focused on the simulation of the *capsulorhexis* (i. e. a surgical intervention for removing cataracts) that can be implemented using two-dimensional meshes. Webster et al. [127] and Allard et al. [128] developed their approaches using real-time commercial frameworks. The former used a linear hybrid FEM–mass-spring system, whereas the latter used FEM to simulate an anisotropic material model. Lastly, Vegamanti et al. [129] developed a TLED-based algorithm for a neo-Hookean material which was not intended for real-time simulation. No recent approaches have been found that either applies tearing in three-dimensional models or makes use of the theory of continuum damage mechanics, but an excellent survey on damage models for soft tissues can be consulted in [130].

1.2.2.4 Other interactions

Apart from the non-invasive interaction of deformation, and the invasive interactions of cutting and tearing (described in the previous sections), other procedures exist that can also be taken into consideration in simulators belonging to the second generation, such as surgical suturing or rupture of soft tissues.

Suturing consists in holding soft tissues together, after an injury or surgery, by performing surgical knots with needles and thread. According to Delingette and Ayache [15], suturing (and also cutting) tissues is of great importance for designing a surgery simulation system. In terms of tissue modelling, such procedures are considerably complex to simulate in a realistic manner, since they should include features such as two-handed interaction, tissue undermining and tissue repositioning [131]. In this regard, only a few simple real-time approaches, based on both linear FEM [132, 133], and also physics engines [134], have been developed.

Rupture of soft tissues is associated to procedures in which the insertion of sharp instruments, such as needles, is produced. A review about needle insertion can be found in Misra et al. (2008) [32], and some recent approaches can be found in [135, 136].

1.2.3 Third generation

The third generation of surgical simulators should incorporate the simulation of physiological phenomena, such as glandular secretions, bleeding, blood pressure, and body temperature, to mention a few. However, a realistic real-time simulation of physiology is, at present, still far from being accomplished [27]. Although the introduction of physiology into surgical simulation is not strictly necessary in all the applications (it does not appear to be an essential requirement for the training of minimally-invasive procedures, for instance), there are some, such as the simulation-based planning of medical procedures or surgical interventions, for which it is often required [137].

In the context of surgical simulation, not all the physiological phenomena are researched to the same extent. **Blood flow**, for example, is by far the most studied physiological feature. Some recent reviews can be found that address bleeding of surgical wounds [138], and blood circulation and haemodynamics [47]. **Body temperature** has also been studied, although in a much lesser degree. In [139], a simulator of minimally-invasive procedures is described, which includes, apart from visual and force feedback, thermal feedback. This simulator is useful for training the detection of unstable arterial plaques and tumours. No literature has been found concerning other kinds of physiological simulations.

1.2.4 Fourth and fifth generations

The development of simulators that add features belonging to generations fourth and fifth, such as microanatomy or biochemistry, is tremendously challenging with the current technology, and no prototype is known which implements them [15, 27]. A few specific interventions exist that may require simulators with such a degree of detail. It is the case, for instance, of vein graft surgery, whose simulation should combine both macroscopic and biochemical levels. This is because it is thought that the shearing forces that blood exerts on the vein wall modulate a particular *gene regulatory network* (i. e. the set of gene interactions that control a specific cell function), which determines an adaptive response. However, simulation at the level of gene regulatory networks represents a significant challenge, since stochastic descriptions of the reactions taking place inside the cells should be developed [140]. This can lead to a complex set of equations with an extremely high number of system variables. In this regard, it is worth of mention that Ammar et al. propose in [141] a technique using PGD to solve the chemical master equation, which governs gene regulatory networks and cell signalling processes.

1.3 Motivation and objectives

As discussed throughout this chapter, real-time requirements pose severe constraints that determine the chosen simulation strategies. One of the major limitations that these strategies should face is related to computing time. Indeed, obtaining the displacements of an organ with a certain degree of realism is a very demanding task, even more if some surgical interactions—complex from the point of view of computational mechanics, but fairly common in the operating theatre—are taken into account. This is the reason why only a few works have been able to solve some of these interactions with relative success.

The main objective of this thesis is to analyse whether the Proper Generalised Decomposition, and more specifically computational vademecums, can be used for achieving a realistic interactive simulation of surgical procedures, in particular, the cutting and tearing of soft tissues. As will be seen in the following chapters, PGD not only provides outstanding results in both cases, but also opens the door to innovative simulation strategies, which can be considered as a real progress in the state of the art.

In order to meet the objective, it can be subdivided into two separate problems or sub-goals, each focused on solving a different surgical interaction. The first problem is the development of a method for the real-time simulation of any arbitrary surgical cut. This can be achieved through a continuous–discontinuous multiscale approach, in which computational vademecums used as reduced bases are combined with XFEM techniques. The second problem consists in developing a method for the real-time simulation of tearing of soft tissues based on the theory of continuum damage mechanics. To this end, a computational vademecum of a parametric damage problem can be used in an explicit framework as a sort of time integrator.

1.4 Thesis outline

PGD constitutes the backbone of this thesis, since it is the primary method used to address the objectives. For this reason, chapter 2 provides a general overview of PGD. Due to the large amount of information that can be found about PGD, this chapter only emphasises the most interesting aspects related to the achievement of the objectives. This includes descriptions of PGD as a model order reduction method, and as a multidimensional and multiparametric solver. Furthermore, a generic computational vademecum of the displacements of an organ, for any position of a unit load applied at any point of a region of its surface, is formulated. This vademecum will be taken as a reference throughout the rest of the work, since it serves as a basis for solving the proposed goals. Chapter 3 describes the methodology used to address the real-time simulation of surgical cutting, and presents an implementation focused on corneal surgery for the correction of visual anomalies. Chapter 4 details an approach for the real-time simulation of tearing of soft tissues based on an isotropic damage model, and shows an application developed for simulating the removal of the adipose tissue that surrounds the walls of the gallbladder. Lastly, some final conclusions are drawn in chapter 5, including thesis contributions, future work, and publications.

Chapter 2

Proper Generalised Decomposition

2.1 Introduction

Section 1.2 showed that the **Proper Generalised Decomposition** (PGD) has proven to be a highly efficient method for the real-time simulation of the deformations of an organ. Moreover, as will be seen throughout this thesis, PGD allows the development of real-time simulations to be effectively extended to other surgical interactions, such as cutting and tearing of soft tissues. For all these reasons, it is appropriate to describe in this chapter a general outline of PGD and its implementations.

The PGD can be roughly defined as a numerical method for solving boundary-value problems (BVP). The key idea behind the method is to assume that the solution of a multidimensional problem can be expressed in **separated representation**, i. e. as a sum of products of functions, each depending (in the best of cases) only on one coordinate. Starting from the weighted residual form of the problem, PGD uses a greedy algorithm to construct the solution by successive enrichment, whereby each functional product is determined sequentially. Because solving decoupled problems is computationally much less expensive than solving a single multidimensional problem, PGD can be considered a **model order reduction** (MOR) method (see § 2.2) [142].

The process by which the PGD framework solves multidimensional problems is so efficient that it can circumvent the issues associated to problems defined in high-dimensional spaces (e. g. the so-called **curse of dimensionality**). In this regard, parametric problems can also be cast into multidimensional approaches just by including the parameters as extra coordinates of the problem, leading to the computation of a sort of general meta-model or response surface (see § 2.3). Since the PGD separated representation is solved and stored in an orderly manner, any particular solution for a desired set of values of the parameters can be obtained very fast from the meta-model. Due to its ready accessibility, this meta-model is often called **computational vademecum**.

Within the field of surgical simulation, the generation of computational vademecums is of great significance, since they allow to obtain, amongst many other things, the deformations of a model in which the position of the load is considered a parameter of the problem. The development of a computational vademecum for this purpose is described in detail in § 2.4. Alternatively, chapters 3 and 4 show novel efficient ways of using the computational vademecum of an organ to implement, in combination with other techniques, surgical interactions different from deformations.

2.2 PGD as a model order reduction method

In general terms, **model order reduction** (**MOR**) methods capture the essential features of a model while reducing its computational complexity. They are especially useful to circumvent the problems of brute-force approaches, thus paving the way for the implementation of real-time applications [143].

MOR methods can be classified into the two following categories [144]: **a posteriori methods**, in which the reduced model is obtained *after* some processing of the solution; and **a priori methods**, in which the reduced model is constructed with *no* previous knowledge of the solution. According to this, some representative a posteriori MOR methods are the Reduced Basis Method (RBM) [145, 146] and the Proper Orthogonal Decomposition (POD) [72, 73] (described in § 4.5.1), whereas PGD [84] is an a priori MOR method.

When solving a multidimensional problem, the reduction of the computational complexity may require the computation of either a reduced basis of low dimension or an approximation of the solution. POD, for example, is identified as a method that builds a reduced basis from a set of data related to the solution of the problem (the so-called *snapshots*), and does not explicitly provide any approximation. By contrast, PGD always generates an approximation of the solution without the need to define beforehand a reduced basis [147, 148]. This does not mean that the PGD approximation of the solution cannot be also used as a reduced basis. In fact, this is the approach followed in this work to perform cutting.

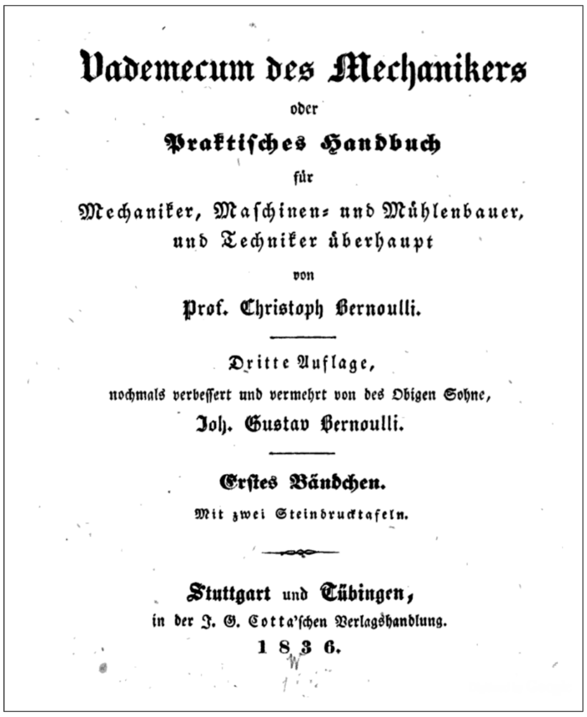
With regard to the reduction of complexity achieved by a MOR method, it can be performed in two different ways, depending on whether an approximation of the solution has been computed or not. The computing workload can either be distributed into an off-line stage and an on-line stage or remain only in the off-line stage. In the first case, a reduced basis is computed in the off-line stage (instead of an approximation of the solution), and during the on-line stage, a simple system of equations is solved for each instance of the problem to obtain the associated weights. Alternatively, the computing workload can be completely left to the off-line stage. In this case, an approximation of the solution is obtained. Since the whole work has been done beforehand, the solution of each instance of the problem is directly available and can be retrieved quasi-instantaneously [147].

In this thesis, the reduced model used for the simulation of surgical cutting (chapter 3) follows the first strategy, i. e. an off-line–on-line approach, whereas the reduced model for the tearing of soft tissues (chapter 4) only requires an off-line approach.

2.3 PGD as a multidimensional and multiparametric solver

In spite of the dramatic improvements that mathematical modelling and computing techniques have experienced during the last decades, certain problems in science and engineering still remain hardly manageable or directly intractable. This is the case of models defined in high-dimensional spaces, which arise in fields such as dynamics of complex fluids, quantum chemistry, and biological systems, to name a few [149]. One of the main reasons why these problems cannot be solved directly using traditional numerical techniques is the phenomenon commonly known as the **curse of dimensionality**, which refers to the fact that the amount of computational resources required to solve a particular problem increases exponentially with the number of dimensions.

PGD has been extensively proven to be particularly suitable for dealing with high-dimensional problems [86, 87, 149–156], thus overcoming the limitations of classical approaches. Furthermore, one of the major advantages of PGD is its ability to address standard multidimensional problems, not necessarily high-dimensional [149, 151]. Parametric problems, in this respect, play an outstanding role, since they can be re-adapted into a multidimensional framework



Vademecum des Mechanikers

oder

Praktiſches Handbuch

für

Mechaniker, Maſchinen- und Mühlenbauer, und Techniker überhaupt

von

Prof. Chriſtoph Bernoulli.

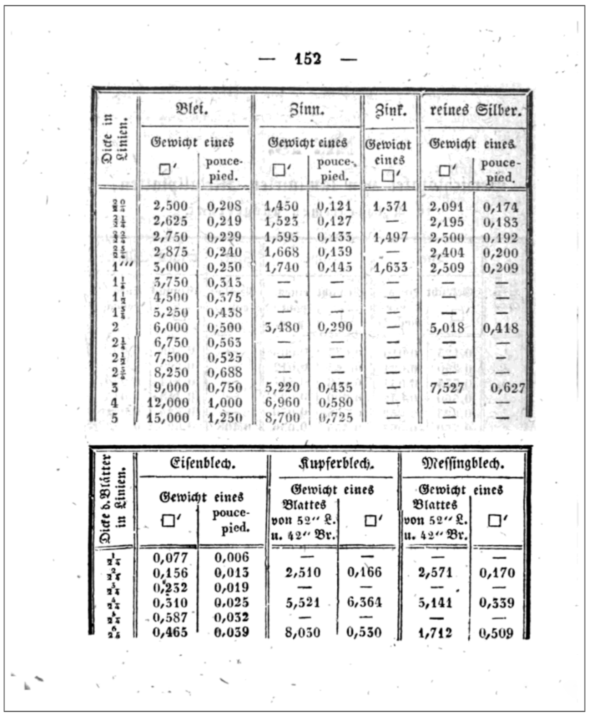
Dritte Auflage, nochmals verbeſſert und vermehrt von des Obigen Sohne, Joh. Guſtav Bernoulli.

Erſtes Bändchen.

Mit zwei Steindrucktafeln.

Stuttgart und Tübingen, in der J. G. Cotta'ſchen Verlagshandlung.

1 8 3 6.



— 152 —

Dicke in Linien.	Blei.		Zinn.		Zink.	reines Silber.	
	Gewicht eines		Gewicht eines		Gewicht eines	Gewicht eines	
	□′	pouce-pied.	□′	pouce-pied.	□′	□′	pouce-pied.
[illegible]	2,500	0,208	1,450	0,121	1,371	2,091	0,174
[illegible]	2,625	0,219	1,523	0,127	—	2,195	0,183
[illegible]	2,750	0,229	1,595	0,133	1,497	2,300	0,192
[illegible]	2,875	0,240	1,668	0,139	—	2,404	0,200
1‴	3,000	0,250	1,740	0,145	1,633	2,509	0,209
$1\frac{1}{4}$	3,750	0,313	—	—	—	—	—
$1\frac{1}{2}$	4,500	0,375	—	—	—	—	—
$1\frac{3}{4}$	5,250	0,438	—	—	—	—	—
2	6,000	0,500	3,480	0,290	—	5,018	0,418
$2\frac{1}{4}$	6,750	0,563	—	—	—	—	—
$2\frac{1}{2}$	7,500	0,525	—	—	—	—	—
$2\frac{3}{4}$	8,250	0,688	—	—	—	—	—
3	9,000	0,750	5,220	0,435	—	7,527	0,627
4	12,000	1,000	6,960	0,580	—	—	—
5	15,000	1,250	8,700	0,725	—	—	—

Dicke d. Blätter in Linien.	Eiſenblech.		Kupferblech.		Meſſingblech.	
	Gewicht eines		Gewicht eines		Gewicht eines	
	□′	pouce-pied.	Blattes von 52″ L. u. 42″ Br.	□′	Blattes von 52″ L. u. 42″ Br.	□′
$\frac{1}{24}$	0,077	0,006	—	—	—	—
$\frac{2}{24}$	0,156	0,013	2,510	0,166	2,571	0,170
$\frac{3}{24}$	0,232	0,019	—	—	—	—
$\frac{4}{24}$	0,310	0,025	5,521	6,364	5,141	0,339
$\frac{5}{24}$	0,587	0,032	—	—	—	—
$\frac{6}{24}$	0,465	0,039	8,030	0,530	1,712	0,509

Figure 2.1. Two pages of the book *Vademecum des Mechanikers*, by C. Bernoulli (1836) [158]. Vademecums were handbooks designed to be easily consulted and provide quick answers in a particular area. Current computational vademecums can be considered as the updated computerised version of these ancient handbooks.

by setting the parameters as extra coordinates of the model. From the resulting model, PGD can be directly applied to obtain a general meta-model that includes all the solutions for every possible value of the parameters. The computation of the meta-model, though, is not immediate and may take some time. However, once it has been obtained, any particular solution of the parametric problem can be reconstructed very fast and on demand simply by particularising the meta-model for the requested values of the parameters. Because this concept resembles an updated version of the ancient vademecums, i. e. the quick reference handbooks used by engineers and technicians in the past to obtain quick numerical solutions on a particular topic (see Figure 2.1), PGD meta-models are often called **computational vademecums**. They were first defined and described in Chinesta et al. [86], whereas computer implementations can be found in the book by Cueto et al. [157]; furthermore, § 2.4 shows their construction in detail.

Following this logic, the combination of both an off-line stage for obtaining a PGD-based computational vademecum and an on-line stage for its subsequent extensive use opens the door to the implementation of highly complex problems in which real-time performance is a constraint. This is the case of the computer-based simulation of surgery. The realistic representation of living soft tissues implies the use of both non-linear constitutive models and techniques to reproduce the virtual tool–organ interactions (see § 1.2.2). Additionally, a feedback response rate of 500–1,000 Hz is needed to obtain a proper haptic rendering or, in other words, an *instance* of the problem has to be solved every 1–2 ms (see § 1.1.3.3). In response to these concerns, the development of PGD-based solutions arise as a new paradigm [151] to decrease the computing time in such high-demanding problems and, therefore, to solve problems that would remain intractable adopting brute-force approaches [86, 149].

2.4 Construction of a computational vademecum

Computational vademecums play a crucial role in the development of strategies for solving complex problems in science and engineering, including the ones posed in this thesis for the real-time simulation of surgery. For this reason, it is fundamental to devote a section to the thorough description of their construction using PGD. To that end, the computation of the vademecum of the displacements of an organ, for any position of a unit load applied at any point of a region of its surface, is taken here as an illustrative example.

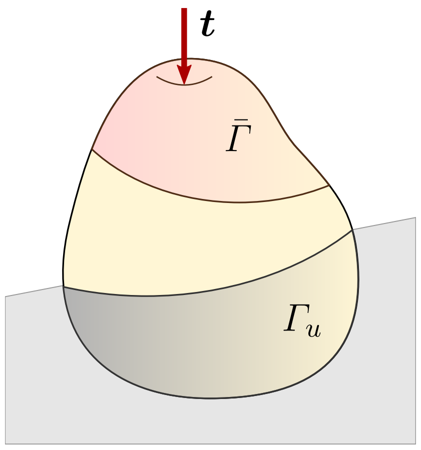


Figure 2.2. Schematic model of an organ subjected to a load. The most relevant portions of its boundary have been identified.

Consider the model of an organ such as the one depicted in Figure 2.2. A part of the boundary of the model, Γ_u, remains fixed, emulating the ligaments that attach the organ to the neighbouring tissues. In addition, it is assumed that only an area $\bar{\Gamma}$ of the remaining boundary Γ_t, is accessible by the surgical instruments. The interaction of these instruments with the organ generates traction loads $\boldsymbol{t}$ on any point of $\bar{\Gamma}$, which produce in the model the displacements $\boldsymbol{u}(\boldsymbol{x})$ that should be computed. However, since the computation of the vademecum for this model requires that the load position $\boldsymbol{s}$ is contemplated as a parameter of the problem, the displacements of the model should be characterised as $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}; \boldsymbol{s})$. Depending on the problem needs, any other variables of interest, such as, for instance, the load magnitude or the properties of the constitutive model, could also be considered as parameters. Different parametric scenarios can be found in [86].

2.4.1 PGD formulation of the problem

To solve the displacements of the proposed model, the static **equilibrium equation** of a three-dimensional linear elastic solid is considered in direct tensor notation,

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0} \text{ in } \Omega, \tag{2.1}$$

where $\boldsymbol{\nabla}\cdot$ denotes the divergence of a tensor field, $\boldsymbol{\sigma}$ is the stress tensor, $\boldsymbol{b}$ is the body force tensor, and Ω is the domain of the solid. The **constitutive equation** relates stress and strain in the model by means of the equation

$$\boldsymbol{\sigma} = \mathbf{C} : \boldsymbol{\varepsilon}, \tag{2.2}$$

where $\mathbf{C}$ is the fourth-order elasticity tensor, and $\boldsymbol{\varepsilon}$ is the elasticity tensor, which is in turn related to the displacements of the model $\boldsymbol{u}$ through the **kinematics equation**,

$$\boldsymbol{\varepsilon} = \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}. \tag{2.3}$$

Additionally, the solid is subjected to the following **boundary conditions**:

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \text{ on } \Gamma_u, \tag{2.4}$$

$$\boldsymbol{t} = \boldsymbol{\sigma}\boldsymbol{n} = \bar{\boldsymbol{t}} \text{ on } \Gamma_t. \tag{2.5}$$

Eq. (2.4) specifies an essential boundary condition in which the displacement field $\boldsymbol{u}$ must satisfy the prescribed value $\bar{\boldsymbol{u}}$ on the portion Γ_u of the boundary Γ; whereas Eq. (2.5) specifies a natural boundary condition in which the internal traction $\boldsymbol{t}$ must equal the prescribed surface traction $\bar{\boldsymbol{t}}$ on the complementary portion Γ_t of the boundary ($\Gamma = \Gamma_u \cup \Gamma_t$). Thus, the problem consists in finding the displacement field $\boldsymbol{u}$, at any point of the solid, for any position of the load $s \in \bar{\Gamma} \subset \Gamma_t$, with $\bar{\Gamma}$ being the portion of the boundary on which the load can be applied.

The PGD formulation assumes that the solution $\boldsymbol{u}$ can be approximated in separated form, i. e. as a finite sum of separable functions such as

$$u_j^N(\boldsymbol{x};\boldsymbol{s}) = \sum_{k=1}^{N} F_j^k(\boldsymbol{x}) \cdot G_j^k(\boldsymbol{s}), \tag{2.6}$$

where the expression u_j refers to the j-th component of the displacement vector, The values of both the rank N and the functions $\boldsymbol{F}^k$ and $\boldsymbol{G}^k$ are a priori unknown.

It is important to remark that the PGD formulation used throughout this work makes use of the Bubnov-Galerkin method (the commonly known as **Galerkin PGD** formulation). This is currently the most taught and implemented PGD formulation [149, 159], and the one used in the first papers by Ammar et al. [84, 85] that gave rise to the PGD method in 2006–2007. Other PGD formulations exist, which can be found in [147, 159].

Under these assumptions, the standard **weak form** of the problem is obtained by multiplying the strong form, Eqs. (2.1)–(2.5), by an arbitrary test function $\boldsymbol{u}^*$, and integrating it over $\mathit{\Omega}$ and $\bar{\mathit{\Gamma}}$. Then, assuming small strain response and negligible body forces, the problem is equivalent to find the displacement field $\boldsymbol{u} \in \mathcal{H}^1(\mathit{\Omega}) \times L_2(\bar{\mathit{\Gamma}})$ such that for all $\boldsymbol{u}^* \in \mathcal{H}_0^1(\mathit{\Omega}) \times L_2(\bar{\mathit{\Gamma}})$,

$$\int_{\bar{\mathit{\Gamma}}} \int_{\mathit{\Omega}} \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u} \, \mathrm{d}\mathit{\Omega} \, \mathrm{d}\bar{\mathit{\Gamma}} = \int_{\bar{\mathit{\Gamma}}} \int_{\mathit{\Gamma}_{t2}} \boldsymbol{u}^* \cdot \boldsymbol{t} \, \mathrm{d}\mathit{\Gamma} \, \mathrm{d}\bar{\mathit{\Gamma}}, \tag{2.7}$$

where $\mathcal{H}_0^1(\mathit{\Omega})$ and $\mathcal{H}^1(\mathit{\Omega})$ denote the Sobolev spaces of homogeneous and non-homogeneous functions (respectively) with first order square-integrable derivatives; $L_2(\bar{\mathit{\Gamma}})$ is the Lebesgue space; $\boldsymbol{\nabla}_{\mathrm{s}} = \frac{1}{2}(\nabla + \nabla^{\mathrm{T}})$ is the symmetric gradient operator; and $\mathit{\Gamma}_t = \mathit{\Gamma}_{t1} \cup \mathit{\Gamma}_{t2}$ are the regions of homogeneous and non-homogeneous natural boundary conditions, respectively.

Then, an iterative algorithm is used for computing the values of the unknown functions $\boldsymbol{F}^k$ and $\boldsymbol{G}^k$. Assuming that, at iteration n of the procedure, convergence has been reached, the approximation of the solution can be expressed as

$$u_j^n(\boldsymbol{x};\boldsymbol{s}) = \sum_{k=1}^{n} F_j^k(\boldsymbol{x}) \cdot G_j^k(\boldsymbol{s}). \tag{2.8}$$

If the desired accuracy of the solution is not obtained with this rank-n approximation, it is necessary to proceed further and look for the $(n+1)$-th term, which can be written as

$$u_j^{n+1}(\boldsymbol{x};\boldsymbol{s}) = u_j^n(\boldsymbol{x};\boldsymbol{s}) + R_j(\boldsymbol{x}) \cdot S_j(\boldsymbol{s}), \tag{2.9}$$

where $\boldsymbol{R}(\boldsymbol{x})$ and $\boldsymbol{S}(\boldsymbol{s})$ are the new functions that enrich the approximation, and which should be computed.

The test function can be obtained from the equation above by applying the standard rules of variational calculus (see [84] for an extended explanation), which results in

$$u_j^*(\boldsymbol{x};\boldsymbol{s}) = R_j^*(\boldsymbol{x}) \cdot S_j(\boldsymbol{s}) + R_j(\boldsymbol{x}) \cdot S_j^*(\boldsymbol{s}). \tag{2.10}$$

To determine the new functions $\boldsymbol{R}$ and $\boldsymbol{S}$, a linearisation strategy should be adopted. The **fixed-point iteration** has shown empirically to produce very good results. It consists in a greedy iterative algorithm which finds, at each stage, a local optimal value for the functions $\boldsymbol{R}$ and $\boldsymbol{S}$ alternatively. Further details of this strategy are provided in the next section.

With regard to the load term, i. e. the right-hand side term in Eq. (2.7), it is assumed for simplicity to have unitary modulus and act along the vertical axis: $\boldsymbol{t} = \boldsymbol{e_k} \cdot \delta(\boldsymbol{x} - \boldsymbol{s})$, where δ represents the Dirac-delta function and $\boldsymbol{e_k}$ the unit vector along the z-coordinate axis.

The Dirac-delta term needs to be approximated by a truncated series of separable functions, following the example of the PGD method, i. e.

$$t_j \approx \sum_{i=1}^{m} f_j^i(\boldsymbol{x}) \cdot g_j^i(\boldsymbol{s}), \tag{2.11}$$

where m represents the order of truncation and f_j^i, g_j^i represent the j-th component of vectorial functions in space and boundary position, respectively.

2.4.2 Fixed-point iteration

As mentioned before, enrichment functions $\boldsymbol{R}$ and $\boldsymbol{S}$ are computed in an alternating direction strategy. At each stage of the algorithm, $S_j^{n+1}(\boldsymbol{s})$ is computed assuming that $R_j^n(\boldsymbol{x})$ is known, and then $R_j^{n+1}(\boldsymbol{x})$ is computed from the already obtained $S_j^{n+1}(\boldsymbol{x})$.

Consider that the solution at iteration n is known, and a new iteration $n+1$ has to be computed. From Eq. (2.7),

$$\int_{\bar{\Gamma}} \int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}^{n+1} \, \mathrm{d}\Omega \, \mathrm{d}\bar{\Gamma} = \int_{\bar{\Gamma}} \int_{\Gamma_{t2}} \boldsymbol{u}^* \cdot \boldsymbol{t} \, \mathrm{d}\Gamma \, \mathrm{d}\bar{\Gamma}.$$

Then, from Eq. (2.9),

$$\int_{\bar{\Gamma}} \int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \left(\boldsymbol{u}^n + \boldsymbol{R} \circ \boldsymbol{S}\right) \mathrm{d}\Omega \, \mathrm{d}\bar{\Gamma} = \int_{\bar{\Gamma}} \int_{\Gamma_{t2}} \boldsymbol{u}^* \cdot \boldsymbol{t} \, \mathrm{d}\Gamma \, \mathrm{d}\bar{\Gamma},$$

where $R_j(\boldsymbol{x}) \cdot S_j(\boldsymbol{s})$ is expressed as $\boldsymbol{R} \circ \boldsymbol{S}$, with the symbol $\circ$ denoting the so-called entry-wise Hadamard (or Schur) multiplication of vectors. Since the symmetric gradient operates only on spatial variables, $\boldsymbol{\nabla}_{\mathrm{s}}\left(\boldsymbol{R} \circ \boldsymbol{S}^*\right)$ can be expressed as $\boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{R} \circ \boldsymbol{S}^*$. By rearranging terms,

$$\int_{\bar{\Gamma}} \int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}^* : \mathsf{C} : \left(\boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{R} \circ \boldsymbol{S}\right) \mathrm{d}\Omega \, \mathrm{d}\bar{\Gamma} = \int_{\bar{\Gamma}} \int_{\Gamma_{t2}} \boldsymbol{u}^* \cdot \boldsymbol{t} \, \mathrm{d}\Gamma \, \mathrm{d}\bar{\Gamma} - \int_{\bar{\Gamma}} \int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}^* : \mathsf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}^n \, \mathrm{d}\Omega \, \mathrm{d}\bar{\Gamma}. \quad (2.12)$$

The fixed-point iteration process for obtaining the solution at iteration $n+1$ is detailed in the following sections, taking Eq. (2.12) as reference.

2.4.2.1 Computation of $\boldsymbol{S}(\boldsymbol{s})$ assuming $\boldsymbol{R}(\boldsymbol{x})$ is known

According to variational calculus, if $\boldsymbol{R}$ is known then $R_j^*(\boldsymbol{x}) = 0$. As a consequence, the admissible variation of the displacement in Eq. (2.10) becomes

$$u_j^*(\boldsymbol{x}; \boldsymbol{s}) = R_j(\boldsymbol{x}) \cdot S_j^*(\boldsymbol{s}), \quad (2.13)$$

By substituting Eqs. (2.8), (2.11) and (2.13) into Eq. (2.12), the resulting expression is

$$\begin{aligned} \int_{\bar{\Gamma}} \int_{\Omega} \left(\boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{R} \circ \boldsymbol{S}^*\right) : \mathsf{C} : \left(\boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{R} \circ \boldsymbol{S}\right) \mathrm{d}\Omega \, \mathrm{d}\bar{\Gamma} &= \int_{\bar{\Gamma}} \int_{\Gamma_{t2}} \left(\boldsymbol{R} \circ \boldsymbol{S}^*\right) \cdot \left(\sum_{k=1}^{m} \boldsymbol{f}^k \circ \boldsymbol{g}^k\right) \mathrm{d}\Gamma \, \mathrm{d}\bar{\Gamma} \\ &\quad - \int_{\bar{\Gamma}} \int_{\Omega} \left(\boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{R} \circ \boldsymbol{S}^*\right) : \mathsf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \left(\sum_{k=1}^{n} \boldsymbol{F}^k \circ \boldsymbol{G}^k\right) \mathrm{d}\Omega \, \mathrm{d}\bar{\Gamma}. \end{aligned}$$

Since all the terms depending on $\boldsymbol{x}$ are known, the integrals over Ω and Γ_{t_2} can be computed and, therefore, an equation for $\boldsymbol{S}(\boldsymbol{s})$ can be obtained.

2.4.2.2 Computation of $\boldsymbol{R}(\boldsymbol{x})$ assuming $\boldsymbol{S}(\boldsymbol{s})$ is known

The same approach is followed when $\boldsymbol{S}$ is known. In this case, $S_j^*(\boldsymbol{s}) = 0$, and then Eq. (2.10) becomes

$$u_j^*(\boldsymbol{x}; \boldsymbol{s}) = R_j^*(\boldsymbol{x}) \cdot S_j(\boldsymbol{s}). \quad (2.14)$$

The substitutions of Eqs. (2.8), (2.11) and (2.14) into Eq. (2.12) result in

$$
\begin{aligned}
\int_{\bar{\Gamma}}\int_{\Omega} (\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{R}^* \circ \boldsymbol{S}) : \mathbf{C} : (\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{R} \circ \boldsymbol{S}) \, \mathrm{d}\Omega \, \mathrm{d}\bar{\Gamma} \\
= \int_{\bar{\Gamma}}\int_{\Gamma_{t2}} (\boldsymbol{R}^* \circ \boldsymbol{S}) \cdot \left(\sum_{k=1}^{m} \boldsymbol{f}^k \circ \boldsymbol{g}^k \right) \mathrm{d}\Gamma \, \mathrm{d}\bar{\Gamma} \\
- \int_{\bar{\Gamma}}\int_{\Omega} (\boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{R}^* \circ \boldsymbol{S}) : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \left(\sum_{k=1}^{n} \boldsymbol{F}^k \circ \boldsymbol{G}^k \right) \mathrm{d}\Omega \, \mathrm{d}\bar{\Gamma}.
\end{aligned}
$$

This time, the terms depending on the load position $\boldsymbol{s}$ are known and can be integrated over $\bar{\Gamma}$, which allows to compute the function $\boldsymbol{R}(\boldsymbol{x})$.

2.4.3 Matrix formulation

To solve PGD on a computer, it is necessary to obtain a discrete form of the functions expressed in the previous sections. To that end, the mixed tensor–matrix formulation described in the book by Cueto et al. [157] is used here, since it has shown to be convenient to obtain the expressions that solve the problem. This formulation considers, for the sake of simplicity, that functions $\boldsymbol{F}^i(\boldsymbol{x})$ and $\boldsymbol{G}^i(\boldsymbol{s})$—also functions $\boldsymbol{R}(\boldsymbol{x})$, $\boldsymbol{S}(\boldsymbol{s})$, $\boldsymbol{f}(\boldsymbol{x})$, and $\boldsymbol{g}(\boldsymbol{s})$—can be defined using a finite element interpolation, which is assumed piecewise linear in this case. Thus, at iteration n, the approximation in Eq. (2.8) can be expressed as

$$
\boldsymbol{u}^n(\boldsymbol{x};\boldsymbol{s}) = \sum_{i=1}^{n} \boldsymbol{F}^i(\boldsymbol{x}) \cdot \boldsymbol{G}^i(\boldsymbol{s}) = \sum_{i=1}^{n} \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \; \boldsymbol{F}_i \; \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \; \boldsymbol{G}_i,
$$

where $\boldsymbol{N}$ and $\boldsymbol{M}$ are vectors containing the values of the finite element shape functions defined in each separated space, and $\boldsymbol{F}_i$ and $\boldsymbol{G}_i$ are vectors containing the nodal values of the finite element mesh for the functions $\boldsymbol{F}^i(\boldsymbol{x})$ and $\boldsymbol{G}^i(\boldsymbol{s})$, respectively. A similar notation can also be found in the first paper on PGD by Ammar et al. [84].

According to this criterion, the search for a new couple of functions in the approximation, Eq. (2.9), can then be expressed as

$$
\begin{aligned}
\boldsymbol{u}^{n+1}(\boldsymbol{x};\boldsymbol{s}) &= \boldsymbol{u}^n(\boldsymbol{x};\boldsymbol{s}) + \sum_{i=1}^{n} \boldsymbol{R}(\boldsymbol{x}) \; \boldsymbol{S}(\boldsymbol{s}) \\
&= \sum_{i=1}^{n} \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \; \boldsymbol{F}_i \; \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \; \boldsymbol{G}_i + \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \; \boldsymbol{R} \; \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \; \boldsymbol{S}. \qquad (2.15)
\end{aligned}
$$

Equivalently, Eq. (2.10) can be expressed as

$$
\boldsymbol{u}^*(\boldsymbol{x};\boldsymbol{s}) = \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \; \boldsymbol{R}^* \; \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \; \boldsymbol{S} + \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \; \boldsymbol{R} \; \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \; \boldsymbol{S}^*. \qquad (2.16)
$$

With regard to the source term, it may not be analytically possible to obtain a separated representation of the expression of the load, i. e. Eq. (2.11) may not be separable. However, the separation is possible in a discrete setting, expressed as:

$$\boldsymbol{t}(\boldsymbol{x};\boldsymbol{s}) \approx \sum_{j=1}^{m} \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{F}_{\boldsymbol{L}j}\ \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s})\ \boldsymbol{G}_{\boldsymbol{L}j}, \tag{2.17}$$

where m is the number of load states. To achieve the separation, $\boldsymbol{F}_{\boldsymbol{L}j}$ should be set as a matrix of zeros whose only non-vanishing entries are, for each column, the position of the nodes that can be subjected to a load. The value of these entries should be equal to the magnitude of the applied force in each case. In short, $\boldsymbol{F}_{\boldsymbol{L}j}$ can be seen as a matrix in which each column represents a load state. Concerning $\boldsymbol{G}_{\boldsymbol{L}j}$, it should be an identity matrix.

2.4.3.1 Computation of $\boldsymbol{S}$ assuming $\boldsymbol{R}$ is known

When $\boldsymbol{R}$ is known, $\boldsymbol{R}^* = \boldsymbol{0}$, and then Eq. (2.16) becomes

$$\boldsymbol{u}^*(\boldsymbol{x};\boldsymbol{s}) = \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R}\ \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s})\ \boldsymbol{S}^*. \tag{2.18}$$

By applying the second term in Eq. (2.15) and Eq. (2.18) into the weak form of the problem, the **left-hand side** of the equality in Eq. (2.12) can be written as

$$\int_{\bar{\Gamma}}\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}}\left(\boldsymbol{S}^{*\mathrm{T}}\ \boldsymbol{M}(\boldsymbol{s})\ \boldsymbol{R}^{\mathrm{T}}\ \boldsymbol{N}(\boldsymbol{x})\right) : \mathsf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\left(\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R}\ \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s})\ \boldsymbol{S}\right)\,\mathrm{d}\Omega\,\mathrm{d}\bar{\Gamma}.$$

After applying separation of variables, the previous equation becomes

$$\int_{\Omega} \boldsymbol{R}^{\mathrm{T}}\ \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{N}(\boldsymbol{x}) : \mathsf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R}\,\mathrm{d}\Omega \cdot \int_{\bar{\Gamma}} \boldsymbol{S}^{*\mathrm{T}}\ \boldsymbol{M}(\boldsymbol{s})\ \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s})\ \boldsymbol{S}\,\mathrm{d}\bar{\Gamma}.$$

Since $\boldsymbol{R}$ and $\boldsymbol{S}$ represent vectors of nodal values of the functions $\boldsymbol{R}(\boldsymbol{x})$ and $\boldsymbol{S}(\boldsymbol{s})$, respectively, they can be considered as constant vectors and, therefore, be moved outside the integrals, giving as a result

$$\begin{multline}\boldsymbol{R}^{\mathrm{T}}\left(\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{N}(\boldsymbol{x}) : \mathsf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\,\mathrm{d}\Omega\right)\boldsymbol{R}\cdot\boldsymbol{S}^{*\mathrm{T}}\left(\int_{\bar{\Gamma}} \boldsymbol{M}(\boldsymbol{s})\ \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s})\,\mathrm{d}\bar{\Gamma}\right)\boldsymbol{S} \\ = \boldsymbol{R}^{\mathrm{T}}\ \boldsymbol{K}\ \boldsymbol{R}\cdot\boldsymbol{S}^{*\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{S}}\ \boldsymbol{S}, \tag{2.19}\end{multline}$$

where $\boldsymbol{K}$ represents a sort of stiffness matrix for the spatial coordinates $\boldsymbol{x}$, and $\boldsymbol{M}_{\boldsymbol{S}}$ represents a mass matrix of the one-dimensional discretisation problem for the parameter $\boldsymbol{s}$.

The **first term on the right-hand side** of the equality in Eq. (2.12), after substituting Eqs. (2.17) and (2.18), has the form

$$\int_{\bar{\Gamma}} \int_{\Gamma_{t2}} \left(\boldsymbol{S}^{*\mathrm{T}} \; \boldsymbol{M}(\boldsymbol{s}) \; \boldsymbol{R}^{\mathrm{T}} \; \boldsymbol{N}(\boldsymbol{x}) \right) \cdot \left(\sum_{j=1}^{m} \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \; \boldsymbol{F}_{\boldsymbol{L}j} \; \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \; \boldsymbol{G}_{\boldsymbol{L}j} \right) \mathrm{d}\Gamma_{t2} \, \mathrm{d}\bar{\Gamma}.$$

After the variable separation

$$\sum_{j=1}^{m} \int_{\Gamma_{t2}} \boldsymbol{R}^{\mathrm{T}} \; \boldsymbol{N}(\boldsymbol{x}) \; \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \; \boldsymbol{F}_{\boldsymbol{L}j} \, \mathrm{d}\Gamma_{t2} \cdot \int_{\bar{\Gamma}} \boldsymbol{S}^{*\mathrm{T}} \; \boldsymbol{M}(\boldsymbol{s}) \; \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \; \boldsymbol{G}_{\boldsymbol{L}j} \, \mathrm{d}\bar{\Gamma}.$$

By moving the constant vectors outside the integrals,

$$\begin{aligned} &\sum_{j=1}^{m} \boldsymbol{R}^{\mathrm{T}} \left(\int_{\Gamma_{t2}} \boldsymbol{N}(\boldsymbol{x}) \; \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \, \mathrm{d}\Gamma_{t2} \right) \boldsymbol{F}_{\boldsymbol{L}j} \cdot \boldsymbol{S}^{*\mathrm{T}} \left(\int_{\bar{\Gamma}} \boldsymbol{M}(\boldsymbol{s}) \; \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \, \mathrm{d}\bar{\Gamma} \right) \boldsymbol{G}_{\boldsymbol{L}j} \\ &\qquad = \sum_{j=1}^{m} \boldsymbol{R}^{\mathrm{T}} \; \boldsymbol{M}_{\boldsymbol{X}} \; \boldsymbol{F}_{\boldsymbol{L}j} \cdot \boldsymbol{S}^{*\mathrm{T}} \; \boldsymbol{M}_{\boldsymbol{S}} \; \boldsymbol{G}_{\boldsymbol{L}j} = \sum_{j=1}^{m} \boldsymbol{R}^{\mathrm{T}} \; \boldsymbol{V}_{\boldsymbol{X}j} \cdot \boldsymbol{S}^{*\mathrm{T}} \; \boldsymbol{V}_{\boldsymbol{S}j}, \quad (2.20) \end{aligned}$$

where $\boldsymbol{M}_{\boldsymbol{X}}$ represents a mass matrix of the three-dimensional discretisation problem for the spatial coordinates $\boldsymbol{x}$. To simplify the notation, $\boldsymbol{V}_{\boldsymbol{X}j} = \boldsymbol{M}_{\boldsymbol{X}} \; \boldsymbol{F}_{\boldsymbol{L}j}$ and $\boldsymbol{V}_{\boldsymbol{S}j} = \boldsymbol{M}_{\boldsymbol{S}} \; \boldsymbol{G}_{\boldsymbol{L}j}$.

The **second term on the right-hand side** of the equality in Eq. (2.12) can be written, after applying Eq. (2.18) and the first term in Eq. (2.15), as

$$\int_{\bar{\Gamma}} \int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \left(\boldsymbol{S}^{*\mathrm{T}} \boldsymbol{M}(\boldsymbol{s}) \boldsymbol{R}^{\mathrm{T}} \boldsymbol{N}(\boldsymbol{x}) \right) : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \left(\sum_{i=1}^{n} \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \boldsymbol{F}_i \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \boldsymbol{G}_i \right) \mathrm{d}\Omega \, \mathrm{d}\bar{\Gamma}.$$

After applying variable separation, and moving constant vectors outside the integral, it results in

$$\begin{aligned} &\sum_{i=1}^{n} \boldsymbol{R}^{\mathrm{T}} \left(\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{N}(\boldsymbol{x}) : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \, \mathrm{d}\Omega \right) \boldsymbol{F}_i \\ &\qquad \cdot \boldsymbol{S}^{*\mathrm{T}} \left(\int_{\bar{\Gamma}} \boldsymbol{M}(\boldsymbol{s}) \; \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s}) \, \mathrm{d}\bar{\Gamma} \right) \boldsymbol{G}_i = \sum_{i=1}^{n} \boldsymbol{R}^{\mathrm{T}} \; \boldsymbol{K} \; \boldsymbol{F}_i \cdot \boldsymbol{S}^{*\mathrm{T}} \; \boldsymbol{M}_{\boldsymbol{S}} \; \boldsymbol{G}_i. \quad (2.21) \end{aligned}$$

Thus, by joining Eqs. (2.19), (2.20) and (2.21), the matrix form of Eq. (2.12) can be expressed as

$$\boldsymbol{R}^{\mathrm{T}} \; \boldsymbol{K} \; \boldsymbol{R} \cdot \boldsymbol{S}^{*\mathrm{T}} \; \boldsymbol{M}_{\boldsymbol{S}} \; \boldsymbol{S} = \sum_{j=1}^{m} \boldsymbol{R}^{\mathrm{T}} \; \boldsymbol{V}_{\boldsymbol{X}j} \cdot \boldsymbol{S}^{*\mathrm{T}} \; \boldsymbol{V}_{\boldsymbol{S}j} - \sum_{i=1}^{n} \boldsymbol{R}^{\mathrm{T}} \; \boldsymbol{K} \; \boldsymbol{F}_i \cdot \boldsymbol{S}^{*\mathrm{T}} \; \boldsymbol{M}_{\boldsymbol{S}} \; \boldsymbol{G}_i.$$

Finally, by solving for $\boldsymbol{S}$,

$$\boldsymbol{S} = \left(\boldsymbol{R}^{\mathrm{T}}\ \boldsymbol{K}\ \boldsymbol{R} \cdot \boldsymbol{S}^{*\mathrm{T}}\ \boldsymbol{M_S}\right)^{-1} \cdot \left(\sum_{j=1}^{m} \boldsymbol{R}^{\mathrm{T}}\ \boldsymbol{V_X}_j \cdot \boldsymbol{S}^{*\mathrm{T}}\ \boldsymbol{V_S}_j - \sum_{i=1}^{n} \boldsymbol{R}^{\mathrm{T}}\ \boldsymbol{K}\ \boldsymbol{F}_i \cdot \boldsymbol{S}^{*\mathrm{T}}\ \boldsymbol{M_S}\ \boldsymbol{G}_i\right). \quad (2.22)$$

The arbitrariness of the weight function $\boldsymbol{S}^{*\mathrm{T}}$ allows to be chosen as the needed value to prove the equivalence.

2.4.3.2 Computation of $\boldsymbol{R}$ assuming $\boldsymbol{S}$ is known

Equivalently to the previous section, when $\boldsymbol{S}$ is known, $\boldsymbol{S}^* = \boldsymbol{0}$, and then Eq. (2.16) becomes

$$\boldsymbol{u}^*(\boldsymbol{x};\boldsymbol{s}) = \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R}^*\ \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s})\ \boldsymbol{S}.$$

By operating in a similar way, Eq. (2.12) results now in

$$\boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{K}\ \boldsymbol{R} \cdot \boldsymbol{S}^{\mathrm{T}}\ \boldsymbol{M_S}\ \boldsymbol{S} = \sum_{j=1}^{m} \boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{V_X}_j \cdot \boldsymbol{S}^{\mathrm{T}}\ \boldsymbol{V_S}_j - \sum_{i=1}^{n} \boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{K}\ \boldsymbol{F}_i \cdot \boldsymbol{S}^{\mathrm{T}}\ \boldsymbol{M_S}\ \boldsymbol{G}_i.$$

$\boldsymbol{R}$ can be solved as

$$\boldsymbol{R} = \left(\boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{K}\ \cdot \boldsymbol{S}^{\mathrm{T}}\ \boldsymbol{M_S}\ \boldsymbol{S}\right)^{-1} \cdot \left(\sum_{j=1}^{m} \boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{V_X}_j \cdot \boldsymbol{S}^{\mathrm{T}}\ \boldsymbol{V_S}_j - \sum_{i=1}^{n} \boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{K}\ \boldsymbol{F}_i \cdot \boldsymbol{S}^{\mathrm{T}}\ \boldsymbol{M_S}\ \boldsymbol{G}_i\right). \quad (2.23)$$

As in the previous case, the weight function $\boldsymbol{R}^{*\mathrm{T}}$ is also arbitrary, and the required value that verifies the equation can be selected.

2.4.4 Computer implementation

This chapter would not be complete without suggesting a computer implementation of the PGD formulation. A simple algorithm is shown in Algorithm 2.1. Basically, it consists of a doubly-nested loop: the inner one (lines 8–13) is known as the *enrichment stage*, and computes the new modes of the PGD solution; and the outer one (lines 5–17) controls the end of the process.

The stopping criterion of the enrichment stage is triggered either when an appropriate measure of error ϵ_R becomes smaller than a certain tolerance tol_R

```
load geometry data ;
compute stiffness and mass matrices $\boldsymbol{K}$ and $\boldsymbol{M_S}$ ;
compute source terms $\boldsymbol{V_X}$ and $\boldsymbol{V_S}$ ;
initialise $\boldsymbol{F}$ and $\boldsymbol{G}$ (as void), and $\epsilon_F$ ;
while $\epsilon_F > tol_F$ and $i < i_{max}$ do
    i = i + 1 ;
    initialise $\boldsymbol{R}$, $\boldsymbol{S}$, and $\epsilon_R$ ;
    while $\epsilon_R > tol_R$ and $j < j_{max}$ do
        j = j + 1 ;
        compute $\boldsymbol{R}$ from Eq. (2.23) ;
        compute $\boldsymbol{S}$ from Eq. (2.22) ;
        compute new value of $\epsilon_R$ ;
    end
    append $\boldsymbol{R}$ to $\boldsymbol{F}$ as the $i$-th mode ;
    append $\boldsymbol{S}$ to $\boldsymbol{G}$ as the $i$-th mode ;
    compute new value of $\epsilon_F$ ;
end
```

Algorithm 2.1. Computer implementation of the PGD algorithm

or when the maximum number of iterations j_{max} has been reached by the current iteration j. This measure of error (line 12) compares how different the newly computed values of $\boldsymbol{R}_j$ and $\boldsymbol{S}_j$ are with respect to the previous ones, $\boldsymbol{R}_{j-1}$ and $\boldsymbol{S}_{j-1}$. Several error measures exist (such as, for example, the L_2 error norm) that can be used as stopping criterion [149].

Once outside the enrichment stage, the computed vectors $\boldsymbol{R}$ and $\boldsymbol{S}$ are appended to $\boldsymbol{F}$ and $\boldsymbol{G}$, respectively, becoming the new i-th modes of the solution. The stopping criterion of the outer loop is very similar to that of the inner loop, with the difference that the tolerance tol_F and the maximum number of iterations i_{max} do not necessarily need to have the same values as in the enrichment stage.

Chapter 3

Simulation of surgical cutting

3.1 Introduction

The problem of simulating surgical cutting is especially challenging, since it involves not only very complex physics, but also topological changes in the geometry of the modelled organ [104, 160–163]. As already discussed in § 1.2.2.2, recent strategies based on XFEM techniques have been successfully applied to this type of phenomena (recall, for instance, [83, 120], where XFEM is coupled to POD).

In this chapter, a combination of computational vademecums and XFEM is presented, which greatly simplifies the task of generating and manipulating displacement discontinuities such as those produced by cuts on the surface of the organ. As will be explained in § 3.2, the simulation requires a previous off-line computation of the vademecum of an organ, i. e. the response of the organ to any possible load produced by the surgical tool on its surface. However, pre-computing the result for any possible location and orientation of a cut or for any displacement discontinuity is not feasible, since the number of potential situations is enormous. This is the reason why the use of the PGD modes *à la* POD, i. e. by projecting onto the subspace spanned by the PGD separated functions, is briefly discussed in § 3.2.

In the approach here presented, PGD methods have been used to develop a vademecum, which is useful when no cut is performed, for simulating the manipulation or palpation of the organ. The main novelty in this approach, however, occurs when a cut is produced (see [161] for a detailed description of the physics prior to the appearance of cutting). At this moment, PGD modes are no longer used as a vademecum, but as global Ritz-like shape functions. These shape functions are parametric, and depend on the position of the contact between the surgical instrument and the organ. Once this parametric dependence has been particularised for a given position, an enriched XFEM-like model is constructed on the fly to take into account the presence of the cut. Results that verify the use of this technique in haptic environments are shown in § 3.5.

3.2 Real-time cutting simulation

Theoretically, a computational vademecum could be constructed for obtaining the response of an organ to the making of an incision with a surgical tool. This vademecum should be formulated to determine the displacements generated at any point of the model, for any position of the tool (a load), for any orientation and module of the force vector, and for any path of the cut. However, this brute force strategy is out of reach due to both the complexity of the resulting formulation and the high computational cost of the off-line stage of the method.

To solve this problem, the approach followed here consists in considering the solution $\boldsymbol{u}$ as given by a vademecum (like in [89, 90]), provided that no cut appears in the on-line (real-time) simulation. Otherwise, once a cut is made, the simulation leaves the vademecum and the model is enriched on the fly accordingly. Thus, the vademecum is considered as a reduced basis for representing the smooth global component of the deformation.

During the on-line interactive phase of the simulation in which the cuts are produced, the displacement field $\boldsymbol{u}$ is simulated in a multiscale framework, i. e. as a continuous approximation enriched by a discontinuous local field generated by the applied cuts,

$$\boldsymbol{u} = \boldsymbol{u}^{\text{cont}} + \boldsymbol{u}^{\text{disc}}.$$

This approach requires the off-line pre-calculation of both contributions, which are detailed below. The on-line combination is also detailed in the last part of this section. Whereas several possibilities arise to treat the discontinuous part of the displacement, such as [164] for instance, the use of the so-called **cracking node method** [165] has been preferred. It provides a discontinuous approximation that can be applied virtually unchanged, also for visualisation purposes. More details are given in the following sections.

3.2.1 Off-line stage for the continuous approximation

The continuous part of the displacement field, $\boldsymbol{u}^{\text{cont}}$, is simulated by taking the PGD modes of a computational vademecum as basis functions. These basis functions are in fact global and parametric, since they depend on the contact point between the scalpel and the organ. For a two-parameter approximation of the displacement, $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{s})$, and an elliptic equation, it has been shown that these modes coincide with the POD or SVD eigenfunctions, and therefore they are optimal in the sense that they incorporate most of the energy of the system with the minimal number of degrees of freedom [166].

Two different routes are considered for the use of PGD parametric bases *à la* POD. These routes depend on whether the projection is performed using a general vademecum (the **parametric reduced basis**), or a vademecum that has been particularised for a specific load position (the **space reduced basis**). Both approaches are described in the following sections. A matrix formulation is also provided in § 3.3, while some implementations and results are compared and analysed in § 3.5.

3.2.1.1 Parametric reduced basis (PRB)

The PRB formulation assumes the use of a computational vademecum as the one developed in § 2.4. Thus, the continuous part of the approximation, $\boldsymbol{u}^{\text{cont}}$, has the form

$$u_j(\boldsymbol{x}; \boldsymbol{s}) \approx \sum_{i=1}^{n} {\beta_{\text{prb}}}_j^i \cdot X_j^i(\boldsymbol{x}) \cdot Y_j^i(\boldsymbol{s}),$$

where the projection coefficients ${\beta_{\text{prb}}}_j^i$ will be determined during the on-line phase, as detailed in § 3.2.3.

3.2.1.2 Space reduced basis (SRB)

The SRB formulation uses a computational vademecum that has been particularised for a load position, $\boldsymbol{s} = s_p$. This allows to obtain any solution related to any source location with the maximum accuracy provided by PGD. Thus, the expression of the continuous part, $\boldsymbol{u}^{\text{cont}}$, can be approximated as

$$u_j(\boldsymbol{x}; \boldsymbol{s} = s_p) \approx \sum_{i=1}^{n} {\beta_{\text{srb}}}_j^i \cdot \tilde{X}_j^i(\boldsymbol{x}), \tag{3.1}$$

with $\tilde{X}_j^i(\boldsymbol{x}) = X_j^i(\boldsymbol{x}) \cdot Y_j^i(\boldsymbol{s} = s_p)$. By introducing this equation into the standard weak form of the problem, coefficients ${\beta_{\text{srb}}}_j^i$ can be computed, as detailed in § 3.2.3. In this regard, it is possible to show that, for the same vademecum, $\beta_{\text{prb}} \neq \beta_{\text{srb}}$ [167].

3.2.2 Off-line stage for the discontinuous approximation

The discontinuous local field of displacements, $\boldsymbol{u}^{\text{disc}}$, is computed using the extended finite element method (XFEM) [168]. This method constitutes an appealing alternative for the efficient modelling of discontinuities, such as cuts or cracks. Essentially, XFEM consists in enriching the nodes that surround a discontinuity with a global discontinuous function that multiplies the local shape functions, thus generating a local discontinuous enrichment. New degrees of freedom are added to the approximation, which represent the amplitude of the discontinuous field. Fig. 3.1 illustrates the method.

In this approach, a set of discontinuous local fields of displacements is precomputed off-line, using XFEM, for any node on the accessible surface of the model, and for any particular angle θ of the cut. The simulation of an arbitrary cut can be constructed from these pre-computed local fields using the cracking node method [165]. Basically, the cracking node method is a particular implementation of XFEM that allows the parametrisation of an arbitrary cut by using nodally-centred cuts, which extend up to the boundary of each element as depicted in Fig. 3.2. Thus, the true geometry of an arbitrary cut is approximated by a collection of cutting segments that pass through the nodes, rather than by a single cut. It should be noted, as a remark, that the cracking node method does not describe any propagative phenomenon. Cuts are arbitrarily generated by the motion of the virtual scalpel, which is driven by the user by means of the haptic device.

Cracking-node techniques are used to get $\boldsymbol{u}^{\text{disc}}$ through discontinuous enrichment functions $\tilde{H}_j$,

$$\boldsymbol{u}^{\text{disc}} = \boldsymbol{u}^{\text{XFEM}}(\boldsymbol{x}, \theta) = \sum_{i \in \mathcal{S}} N^i(\boldsymbol{x}) \tilde{H}^i(\boldsymbol{x}, \theta) \boldsymbol{q}^i(t),$$

where $N^i(\boldsymbol{x})$ denotes the usual finite element shape functions, $\boldsymbol{q}^i(t)$ represents the nodal coefficients, that may depend on time for dynamical applications (only quasi-static examples are considered here, related to a pseudo-time, rather than time), and physically control the amplitude of the displacement discontinuity. $\mathcal{S}$ represents the set of nodes affected by the cut, and therefore with enriched degrees of freedom. Their associated shape functions, $N^i(\boldsymbol{x}) \cdot \tilde{H}^i(\boldsymbol{x}, \theta)$ are implemented with a parametric dependence on the angle of the cut with respect to the reference coordinates of the element, θ. However, θ is not separated as an independent coordinate of the problem. Only the dependence of $\tilde{H}^i(\boldsymbol{x}, \theta)$ is highlighted here. More details of the implementation of the cracking node method are given in the original reference [165].

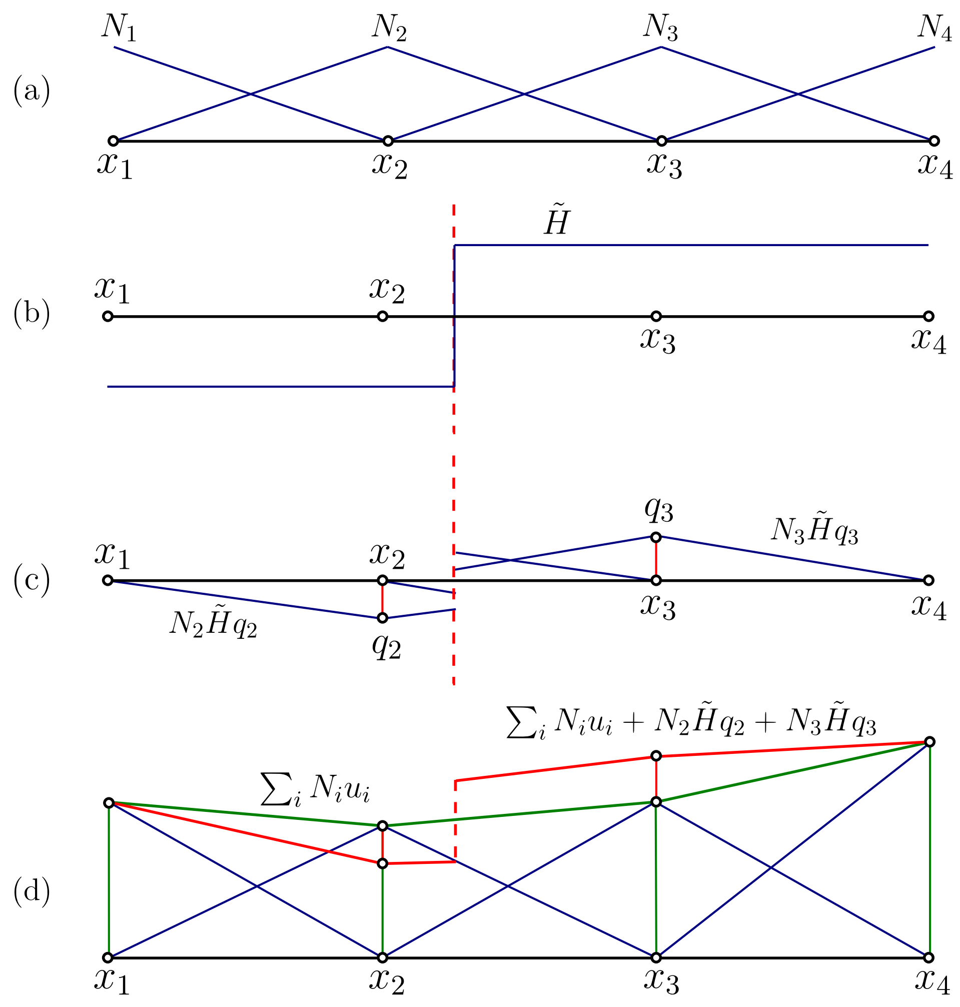


Figure 3.1. One-dimensional XFEM example. (a) Local FEM shape functions N_i; (b) global XFEM discontinuous function $\tilde{H}$; (c) local discontinuous enrichments q_i; (d) FEM solution (in green), and XFEM enriched solution (in red). Figure adapted from [40].

It is worth noting that the use of the cracking node method greatly simplifies the procedure introduced in [83] for the on-line part of the simulation. In particular, it avoids using enrichments through the s-FEM method [169] all along the path of the XFEM enriched model.

3.2.3 On-line stage

During the on-line stage, real-time cutting is performed interactively following the path indicated by the user with the haptic device. Displacements produced by the interactive cutting are computed by projecting the solution onto a subspace spanned by the approximation functions (both the continuous and discontinuous parts), which have been pre-computed in the off-line stage. The

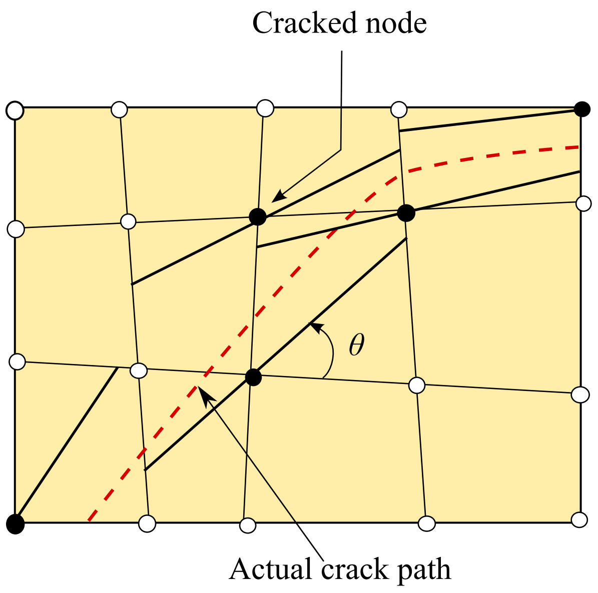


Figure 3.2. Sketch of the principles of the cracking node method. Figure adapted from [165].

parametric dependence of the global shape functions on the particular position of contact of the scalpel should be emphasised again here. Indeed, the computation time required at this stage must be compatible with the required haptic feedback rates (500-1000 Hz). To achieve this, an efficient method to compute the solution should be developed.

Both the PGD basis where to project $\boldsymbol{u}^{\text{cont}}$ and the discontinuous enrichment $\boldsymbol{u}^{\text{disc}}$ obtained in the off-line stage are used now as a reduced basis to project the solution $\boldsymbol{u}$. So to speak, PGD parametric basis functions are now used *à la* POD to find a suitable projection, enriched with the discontinuous XFEM field.

The weak form of the problem consists in finding the displacement $\boldsymbol{u}(\boldsymbol{x},\boldsymbol{s}) \in \mathcal{H}^1(\Omega \times \bar{\Gamma})$ such that for all $\boldsymbol{u}^* \in \mathcal{H}^1_0$,

$$\int_{\bar{\Gamma}} \int_{\Omega} \boldsymbol{\nabla}_{\text{s}} \boldsymbol{u}^* : \boldsymbol{\sigma}\, d\Omega\, d\bar{\Gamma} = \int_{\bar{\Gamma}} \int_{\Gamma_t} \boldsymbol{u}^* \cdot \boldsymbol{t}\, d\Gamma\, d\bar{\Gamma}. \tag{3.2}$$

Assuming the PRB formulation (see § 3.2.1.1), $\boldsymbol{u}$ has the following form:

$$u_j(\boldsymbol{x},\boldsymbol{s},\theta) \approx \underbrace{\sum_{k=1}^{n} \beta_j^k \cdot X_j^k(\boldsymbol{x}) \cdot Y_j^k(\boldsymbol{s})}_{\boldsymbol{u}^{\text{cont}}} + \underbrace{\sum_{\ell=1}^{m} q_j^\ell \cdot N^\ell(\boldsymbol{x}) \cdot \tilde{H}^\ell(\boldsymbol{x},\theta)}_{\boldsymbol{u}^{\text{disc}}}, \tag{3.3}$$

where β_j^k and q_j^ℓ (to be determined) can be considered as a sort of weighting coefficients that allow balancing the already known expressions $X_j^k(\boldsymbol{x})\cdot Y_j^k(\boldsymbol{s})$ and $N^\ell(\boldsymbol{x}) \cdot \tilde{H}^\ell(\boldsymbol{x},\theta)$, i. e. the PGD approximation and its discontinuous enrichment.

Continuity of the displacement field is ensured by the compact support of the typical finite element shape functions $N^\ell(\boldsymbol{x})$. The admissible variation of the displacement will thus be given by

$$u_j{}^*(\boldsymbol{x},\boldsymbol{s},\theta) = \sum_{k=1}^{n} \beta_j^{k^*} \cdot X_j^k(\boldsymbol{x}) \cdot Y_j^k(\boldsymbol{s}) + \sum_{\ell=1}^{m} q_j^{\ell^*} \cdot N^\ell(\boldsymbol{x}) \cdot \tilde{H}^\ell(\boldsymbol{x},\theta). \tag{3.4}$$

Note that in the case of the SRB approach, there is no need to use a doubly-week form, since the basis no longer contains functions of load location (depending on the s coordinate). Therefore, the weak form of the problem is the standard one, i. e. find the displacement $\boldsymbol{u}(\boldsymbol{x},\boldsymbol{s}) \in \mathcal{H}^1(\Omega \times \bar{\Gamma})$ such that $\forall \boldsymbol{u}^* \in \mathcal{H}_0^1$:

$$\int_\Omega \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{u}^* : \boldsymbol{\sigma} \, \mathrm{d}\Omega = \int_{\Gamma_t} \boldsymbol{u}^* \cdot \boldsymbol{t} \, \mathrm{d}\Gamma. \tag{3.5}$$

By substituting Eqs. (3.3) and (3.4) into the weak form of the problem, Eq. (3.2), a discrete expression for obtaining β_j^k, and q_j^ℓ can be established as:

$$\begin{pmatrix} \boldsymbol{K}^{\beta\beta} & \boldsymbol{K}^{\beta q_1} & \boldsymbol{K}^{\beta q_2} & \cdots & \boldsymbol{K}^{\beta q_m} \\ \boldsymbol{K}^{q_1\beta} & \boldsymbol{K}^{q_1 q_1} & \boldsymbol{K}^{q_1 q_2} & \cdots & \boldsymbol{K}^{q_1 q_m} \\ \boldsymbol{K}^{q_2\beta} & \boldsymbol{K}^{q_2 q_1} & \boldsymbol{K}^{q_2 q_2} & \cdots & \boldsymbol{K}^{q_2 q_m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{K}^{q_m\beta} & \boldsymbol{K}^{q_m q_1} & \boldsymbol{K}^{q_m q_2} & \cdots & \boldsymbol{K}^{q_m q_m} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{q}_m \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{pmatrix}. \tag{3.6}$$

Note that the angle of the cut, θ, is not considered as a parameter in the formulation. This means that no integral in θ is performed on the weak form of the problem. Instead, the value of θ imposed by the blade of the scalpel is particularised in $\tilde{H}^\ell(\boldsymbol{x},\theta)$ to provide a closed-form expression $\tilde{H}^\ell(\boldsymbol{x})$.

Note also that, in an interactive simulation, the number of nodes m belonging to the *active* set $\mathcal{S}$ which are affected by the cut increases as the user continues with the procedure. Therefore, the size of the stiffness matrix is also increasing during the interactive simulation. However, since the modes controlling the continuous part of the displacement of the organ are those related to the PGD solution, and therefore very limited in number, the size of the stiffness matrix does not increase drastically. In the numerical experiments performed so far, this size did not prevented the algorithm from running under real-time constraints, as will be demonstrated, even for non-optimised code written in the Matlab programming language.

3.3 Matrix formulation

To solve the problem on a computer, the functions expressed in the previous sections should be adapted to a discrete formulation. In the following sections, matrix expressions for both the PRB and the SRB approaches are developed.

3.3.1 PRB formulation

The matrix form of Eq. (3.3) can be written as

$$\boldsymbol{u}(\boldsymbol{x};\boldsymbol{s},\theta)=\sum_{i=1}^{I}\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{X}_i\ \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s})\ \boldsymbol{Y}_i\ \boldsymbol{\beta}_i+\sum_{k=1}^{K}\tilde{\boldsymbol{N}}_k^{\mathrm{T}}(\boldsymbol{x},\theta)\ \boldsymbol{q}_k, \tag{3.7}$$

where $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ are vectors containing the nodal values of the finite element mesh for the functions $\boldsymbol{X}^i(\boldsymbol{x})$ and $\boldsymbol{Y}^i(\boldsymbol{s})$, respectively; $\boldsymbol{\beta}_i$ and $\boldsymbol{q}_k$ are vectors containing the weighting coefficients of the continuous and the discontinuous parts, respectively, of the solution; $\tilde{\boldsymbol{N}}_k(\boldsymbol{x},\theta)$ is a vector containing the values of the finite element shape functions enriched by the XFEM Heaviside step funtions; I is the number of modes of the computational vademecum, and K is the number of cut nodes of the model.

The matrix form of the admissible variation of the displacement, Eq. (3.4), can be written as

$$\boldsymbol{u}^*(\boldsymbol{x};\boldsymbol{s},\theta)=\sum_{j=1}^{J}\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{X}_j\ \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{s})\ \boldsymbol{Y}_j\ {\boldsymbol{\beta}_j}^*+\sum_{\ell=1}^{L}\tilde{\boldsymbol{N}}_\ell^{\mathrm{T}}(\boldsymbol{x},\theta)\ {\boldsymbol{q}_\ell}^*, \tag{3.8}$$

where $J=I$, and $L=K$. With regard to the load $\boldsymbol{t}$, the same matrix form described in Eq. (2.17) is used.

By applying Eqs. (3.7), (3.8), and (2.17) into Eq. (3.5), it follows

$$\begin{aligned}
&\int_\Omega \boldsymbol{\nabla}_{\mathrm{s}}\left(\sum_{j=1}^{J}{\boldsymbol{\beta}_j}^{*\mathrm{T}}\ {\boldsymbol{Y}_j}^{\mathrm{T}}\ \boldsymbol{M}\ {\boldsymbol{X}_j}^{\mathrm{T}}\ \boldsymbol{N}\right):\mathbf{C}:\boldsymbol{\nabla}_{\mathrm{s}}\left(\sum_{i=1}^{I}\boldsymbol{N}^{\mathrm{T}}\ \boldsymbol{X}_i\ \boldsymbol{M}^{\mathrm{T}}\ \boldsymbol{Y}_i\ \boldsymbol{\beta}_i\right)\mathrm{d}\Omega\\
&\quad+\int_\Omega \boldsymbol{\nabla}_{\mathrm{s}}\left(\sum_{j=1}^{J}{\boldsymbol{\beta}_j}^{*\mathrm{T}}\ {\boldsymbol{Y}_j}^{\mathrm{T}}\ \boldsymbol{M}\ {\boldsymbol{X}_j}^{\mathrm{T}}\ \boldsymbol{N}\right):\mathbf{C}:\boldsymbol{\nabla}_{\mathrm{s}}\left(\sum_{k=1}^{K}\tilde{\boldsymbol{N}}_k^{\mathrm{T}}\ \boldsymbol{q}_k\right)\mathrm{d}\Omega\\
&\quad+\int_\Omega \boldsymbol{\nabla}_{\mathrm{s}}\left(\sum_{\ell=1}^{L}{\boldsymbol{q}_\ell}^{*\mathrm{T}}\ \tilde{\boldsymbol{N}}_\ell\right):\mathbf{C}:\boldsymbol{\nabla}_{\mathrm{s}}\left(\sum_{i=1}^{I}\boldsymbol{N}^{\mathrm{T}}\ \boldsymbol{X}_i\ \boldsymbol{M}^{\mathrm{T}}\ \boldsymbol{Y}_i\ \boldsymbol{\beta}_i\right)\mathrm{d}\Omega
\end{aligned}$$

$$
\begin{aligned}
&+ \int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \left(\sum_{\ell=1}^{L} \boldsymbol{q}_{\ell}^{*\mathrm{T}} \, \tilde{\boldsymbol{N}}_{\ell} \right) : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \left(\sum_{k=1}^{K} \tilde{\boldsymbol{N}}_{k}^{\mathrm{T}} \, \boldsymbol{q}_{k} \right) \mathrm{d}\Omega \\
&= \int_{\bar{\Gamma}} \left(\sum_{j=1}^{J} \boldsymbol{\beta}_{j}^{*\mathrm{T}} \, \boldsymbol{Y}_{j}^{\mathrm{T}} \, \boldsymbol{M} \, \boldsymbol{X}_{j}^{\mathrm{T}} \, \boldsymbol{N} \right) \cdot \left(\sum_{n=1}^{N} \boldsymbol{N}^{\mathrm{T}} \, \boldsymbol{F}_{\boldsymbol{L}n} \, \boldsymbol{M}^{\mathrm{T}} \, \boldsymbol{G}_{\boldsymbol{L}n} \right) \mathrm{d}\bar{\Gamma} \\
&\quad + \int_{\bar{\Gamma}} \left(\sum_{\ell=1}^{L} \boldsymbol{q}_{\ell}^{*\mathrm{T}} \, \tilde{\boldsymbol{N}}_{\ell} \right) \cdot \left(\sum_{n=1}^{N} \boldsymbol{N}^{\mathrm{T}} \, \boldsymbol{F}_{\boldsymbol{L}n} \, \boldsymbol{M}^{\mathrm{T}} \, \boldsymbol{G}_{\boldsymbol{L}n} \right) \mathrm{d}\bar{\Gamma}, \qquad (3.9)
\end{aligned}
$$

where the dependencies of the vectors containing the values of the finite element shape functions with $\boldsymbol{x}$ and $\boldsymbol{s}$ have been omitted for clarity.

After rearranging the matrices in a separate form, all the terms in Eq. (3.9) can be expressed more compactly as

$$
\begin{aligned}
&\sum_{j=1}^{J} \sum_{i=1}^{I} \boldsymbol{\beta}_{j}^{*\mathrm{T}} \, \boldsymbol{X}_{j}^{\mathrm{T}} \, \boldsymbol{K}_{00} \, \boldsymbol{X}_{i} \, \boldsymbol{Y}_{j}^{\mathrm{T}} \, \boldsymbol{M}_{\boldsymbol{S}} \, \boldsymbol{Y}_{i} \, \boldsymbol{\beta}_{i} \\
&\quad + \sum_{j=1}^{J} \sum_{k=1}^{K} \boldsymbol{\beta}_{j}^{*\mathrm{T}} \, \boldsymbol{X}_{j}^{\mathrm{T}} \, \boldsymbol{K}_{0k} \, \boldsymbol{Y}_{j}^{\mathrm{T}} \, \boldsymbol{M} \, \boldsymbol{q}_{k} + \sum_{\ell=1}^{L} \sum_{i=1}^{I} \boldsymbol{q}_{\ell}^{*\mathrm{T}} \, \boldsymbol{K}_{\ell 0} \, \boldsymbol{X}_{i} \, \boldsymbol{M}^{\mathrm{T}} \, \boldsymbol{Y}_{j} \, \boldsymbol{\beta}_{i} \\
&\quad + \sum_{\ell=1}^{L} \sum_{k=1}^{K} \boldsymbol{q}_{\ell}^{*\mathrm{T}} \, \boldsymbol{K}_{\ell k} \, \boldsymbol{q}_{k} = \sum_{j=1}^{J} \sum_{n=1}^{N} \boldsymbol{\beta}_{j}^{*\mathrm{T}} \, \boldsymbol{X}_{j}^{\mathrm{T}} \, \boldsymbol{N}_{\boldsymbol{X}} \, \boldsymbol{F}_{\boldsymbol{L}n} \, \boldsymbol{Y}_{j}^{\mathrm{T}} \, \boldsymbol{M}_{\boldsymbol{S}} \, \boldsymbol{G}_{\boldsymbol{L}n} \\
&\qquad + \sum_{l=1}^{L} \sum_{n=1}^{N} \boldsymbol{q}_{\ell}^{*\mathrm{T}} \, \tilde{\boldsymbol{N}}_{\boldsymbol{X}} \, \boldsymbol{F}_{\boldsymbol{L}n} \, \boldsymbol{M}_{\boldsymbol{S}} \, \boldsymbol{G}_{\boldsymbol{L}n}, \qquad (3.10)
\end{aligned}
$$

where the different matrices $\boldsymbol{K}$ are a sort of stiffness matrices of the system as follows:

$$
\begin{aligned}
\boldsymbol{K}_{00} &= \int_{\Omega_{\boldsymbol{x}}} \boldsymbol{B}^{\mathrm{T}} \, \mathbf{C} \, \boldsymbol{B} \, \mathrm{d}\Omega_{\boldsymbol{x}}, & \boldsymbol{K}_{0k} &= \int_{\Omega_{\boldsymbol{x}}} \boldsymbol{B}^{\mathrm{T}} \, \mathbf{C} \, \tilde{\boldsymbol{B}}_{k} \, \mathrm{d}\Omega_{\boldsymbol{x}}, \\
\boldsymbol{K}_{\ell 0} &= \int_{\Omega_{\boldsymbol{x}}} \tilde{\boldsymbol{B}}_{\ell}^{\mathrm{T}} \, \mathbf{C} \, \boldsymbol{B} \, \mathrm{d}\Omega_{\boldsymbol{x}} = \boldsymbol{K}_{0k}{}^{\mathrm{T}}, & \boldsymbol{K}_{\ell k} &= \int_{\Omega_{\boldsymbol{x}}} \tilde{\boldsymbol{B}}_{\ell}^{\mathrm{T}} \, \mathbf{C} \, \tilde{\boldsymbol{B}}_{k} \, \mathrm{d}\Omega_{\boldsymbol{x}};
\end{aligned}
$$

with $\boldsymbol{B}$ being the standard shape function derivative matrix, and $\tilde{\boldsymbol{B}}_{i}$ being the shape function derivative matrix that accounts for the discontinuities generated by the i-th node. Additionally, $\boldsymbol{N}_{\boldsymbol{X}} = \int_{\bar{\Gamma}_{\boldsymbol{x}}} \boldsymbol{N} \, \boldsymbol{N}^{\mathrm{T}} \, \mathrm{d}\bar{\Gamma}_{\boldsymbol{x}}$ is the mass matrix of the system, and $\tilde{\boldsymbol{N}}_{\boldsymbol{X}} = \int_{\bar{\Gamma}_{\boldsymbol{x}}} \tilde{\boldsymbol{N}}_{\ell} \, \boldsymbol{N}^{\mathrm{T}} \, \mathrm{d}\bar{\Gamma}_{\boldsymbol{x}}$ is a mass matrix that accounts for both the standard and the enriched finite element shape funtions. Since no common support exists between them, $\tilde{\boldsymbol{N}}_{\boldsymbol{X}} = \mathbf{0}$ and, therefore, the last term in the right-hand side of Eq. (3.10) can be omitted.

If the following substitutions are made,

$$\boldsymbol{K}^{\beta\beta} = \sum_{j=1}^{J}\sum_{i=1}^{I} \boldsymbol{X}_j{}^{\mathrm{T}}\ \boldsymbol{K}_{00}\ \boldsymbol{X}_i \cdot \boldsymbol{Y}_j{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{S}}\ \boldsymbol{Y}_i, \qquad \boldsymbol{K}^{q_\ell q_k} = \boldsymbol{K}_{\ell k},$$

$$\boldsymbol{K}^{q_\ell \beta} = \sum_{i=1}^{I} \boldsymbol{K}_{\ell 0}\ \boldsymbol{X}_i \cdot \boldsymbol{M}_{\boldsymbol{S}}{}^{\mathrm{T}}\ \boldsymbol{Y}_i, \qquad \boldsymbol{K}^{\beta q_k} = \sum_{j=1}^{J} \boldsymbol{X}_j{}^{\mathrm{T}}\ \boldsymbol{K}_{0k} \cdot \boldsymbol{Y}_j{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{S}},$$

$$\boldsymbol{f} = \sum_{j=1}^{J}\sum_{n=1}^{N} \boldsymbol{X}_j{}^{\mathrm{T}}\ \boldsymbol{N}_{\boldsymbol{X}}\ \boldsymbol{F}_{\boldsymbol{L}n} \cdot \boldsymbol{Y}_j{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{S}}\ \boldsymbol{G}_{\boldsymbol{L}n},$$

then Eq. (3.10) can be expressed as

$$\boldsymbol{\beta}^{*\mathrm{T}}\boldsymbol{K}^{\beta\beta}\boldsymbol{\beta} + \sum_{k=1}^{c} \boldsymbol{\beta}^{*\mathrm{T}}\boldsymbol{K}^{\beta q_k}\boldsymbol{q}_k + \sum_{\ell=1}^{c} \boldsymbol{q}_\ell{}^{*\mathrm{T}}\boldsymbol{K}^{q_\ell\beta}\boldsymbol{\beta} + \sum_{\ell=1}^{c}\sum_{k=1}^{c} \boldsymbol{q}_\ell{}^{*\mathrm{T}}\boldsymbol{K}^{q_\ell q_k}\boldsymbol{q}_k = \boldsymbol{\beta}^{*\mathrm{T}}\boldsymbol{f},$$

where c is the number of cut nodes of the model. Alternatively, the equation above can be written in the following form

$$\begin{pmatrix} \boldsymbol{\beta}^* \\ \boldsymbol{q}_1^* \\ \boldsymbol{q}_2^* \\ \vdots \\ \boldsymbol{q}_c^* \end{pmatrix}^{\mathrm{T}} \cdot \begin{pmatrix} \boldsymbol{K}^{\beta\beta} & \boldsymbol{K}^{\beta q_1} & \boldsymbol{K}^{\beta q_2} & \cdots & \boldsymbol{K}^{\beta q_c} \\ \boldsymbol{K}^{q_1\beta} & \boldsymbol{K}^{q_1 q_1} & \boldsymbol{K}^{q_1 q_2} & \cdots & \boldsymbol{K}^{q_1 q_c} \\ \boldsymbol{K}^{q_2\beta} & \boldsymbol{K}^{q_2 q_1} & \boldsymbol{K}^{q_2 q_2} & \cdots & \boldsymbol{K}^{q_2 q_c} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{K}^{q_c\beta} & \boldsymbol{K}^{q_c q_1} & \boldsymbol{K}^{q_c q_2} & \cdots & \boldsymbol{K}^{q_c q_c} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{q}_c \end{pmatrix} = \begin{pmatrix} \boldsymbol{\beta}^* \\ \boldsymbol{q}_1^* \\ \boldsymbol{q}_2^* \\ \vdots \\ \boldsymbol{q}_c^* \end{pmatrix}^{\mathrm{T}} \cdot \begin{pmatrix} \boldsymbol{f} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

which corresponds with the discrete expression in Eq. (3.6) for computing the weighting coefficients of the problem.

3.3.2 SRB formulation

By following Eq. (3.1), and operating similarly to the previous section, submatrices $\boldsymbol{K}$ of the system of equations take the following forms:

$$\boldsymbol{K}^{\beta\beta} = \sum_{j=1}^{J}\sum_{i=1}^{I} \boldsymbol{X}_j{}^{\mathrm{T}}\ \boldsymbol{K}_{00}\ \tilde{\boldsymbol{X}}_i, \qquad \boldsymbol{K}^{q_\ell q_k} = \boldsymbol{K}_{\ell k},$$

$$\boldsymbol{K}^{q_\ell \beta} = \sum_{i=1}^{I} \boldsymbol{K}_{\ell 0}\ \tilde{\boldsymbol{X}}_i, \qquad \boldsymbol{K}^{\beta q_k} = \sum_{j=1}^{J} \boldsymbol{X}_j{}^{\mathrm{T}}\ \boldsymbol{K}_{0k},$$

$$\boldsymbol{f} = \sum_{j=1}^{J}\sum_{n=1}^{N} \boldsymbol{X}_j{}^{\mathrm{T}}\ \boldsymbol{N}_{\boldsymbol{X}}\ \boldsymbol{F}_{\boldsymbol{L}n}.$$

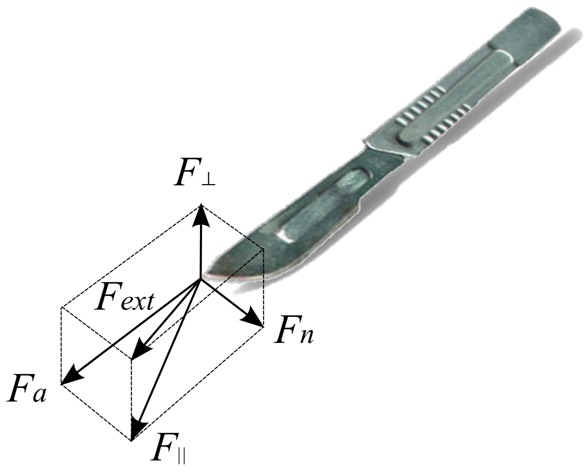


Figure 3.3. Force decomposition at the point of contact of the scalpel.

3.4 Simplified physics of the cutting procedure

The simplified physics of the cutting procedure is strictly assumed in [83]. In this reference, once contact between the surgical tool and the organ under consideration has been detected by a suitable contact algorithm (see [170] for instance, for a valid contact criterion in reduced model settings or, more adequately, [171] for an example using PGD), a criterion must be set in order to determine whether a cut is produced or not, thus possibly generating a new surface boundary in the domain. Although complex physics occur [161], the criterion established in [104] is followed here closely. This criterion has demonstrated to provide realistic enough results in haptic environments, even if it simplifies the actual physics taking place during surgery.

Since a scalpel produces a cut in the plane defined by its blade, the acting force is decomposed as indicated in Fig. 3.3:

$$\boldsymbol{F}_{\text{ext}} = \boldsymbol{F}_\perp + \boldsymbol{F}_{||} = \boldsymbol{F}_\perp + \boldsymbol{F}_a + \boldsymbol{F}_n.$$

In the aforementioned reference, a threshold value of the force F_{cut} is considered such that forces $\boldsymbol{F}_{||}$ with modulus smaller than F_{cut} produce friction, but no cut. Once $||\boldsymbol{F}_{||}||$ exceeds F_{cut}, the cut is produced and a discontinuity in the displacement field must be incorporated into the model. In this thesis a unit value of F_{cut} has been considered, in the absence of any experimental result.

In order to simplify the process and to make it simpler and (notably) faster, once the threshold value F_{cut} is reached, a whole finite element is then cut. No cut of length smaller than the element size is considered. If the finite element mesh is dense enough, this limitation does not very much affect the results. It is important to recall that the size of the global finite element mesh does not alter the number of degrees of freedom of the reduced model, as will be clear hereafter.

3.5 Numerical results

The method described in this chapter has been tested using simple patch test models before developing an application for the simulation of corneal surgery. Both implementations of the method are analysed in this section.

3.5.1 Patch test

A three-dimensional model consisting of a vertical grid of 5×5 unit regular hexahedral elements is considered in this example. One of the nodes in the bottom corner of the model is fully fixed, while the remaining nodes of the base allow certain displacements along the horizontal plane. A straight cut involving two nodes is present on the surface of the model, in such a manner that it opens slightly when a vertical load of 1 N is applied to one of the upper nodes. The constitutive material is linear elastic, with Young's modulus $E = 5$ Pa, and Poisson's ratio $\nu = 0.3$. The solution has been computed using both PGD with the PRB and SRB formulations, and XFEM (with no PGD), taking the latter as a reference solution. PGD solutions required the computation of 11 modes before reaching convergence. As it can be checked by visual assessment of Fig. 3.4, both PGD formulations provide results that are quite similar to the reference model.

3.5.2 Cutting of the cornea

In this section an example of corneal surgery is studied [172]. In particular, astigmatism surgery by means of radial keratotomy is considered. This type of surgery consists in performing radial incisions in the corneal tissue with a diamond knife. The objective is producing a change of the curvature of the cornea, by reducing its lack of sphericity, and potentially eliminating the defects associated to myopia or astigmatism.

This numerical example aims to reproduce the cut on the cornea with reasonable accuracy, compatible with the physical sensation felt by a surgeon. To that end, a finite element model of the cornea, developed by Calvo et al. [173, 174], is used.

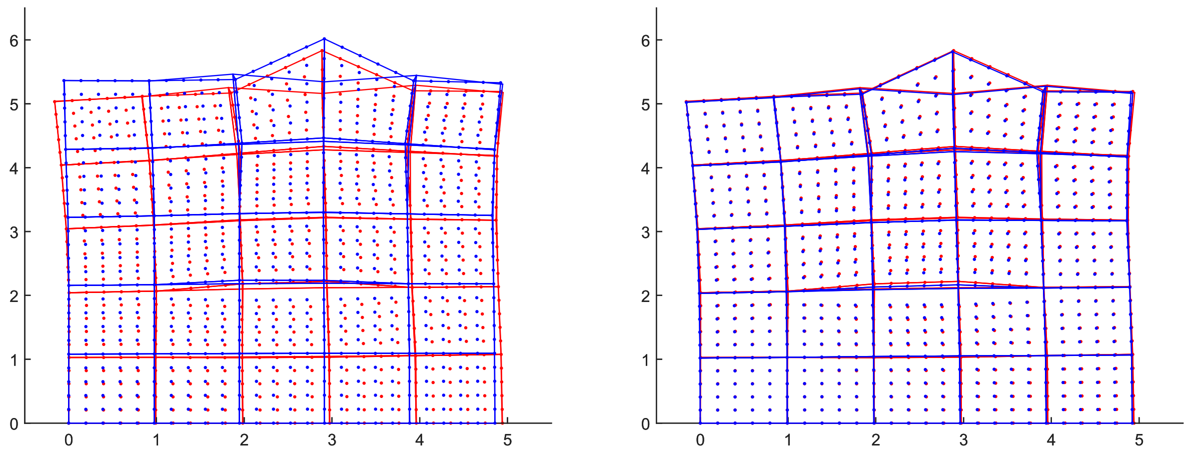


Figure 3.4. Patch test results. The FE mesh of the patches, and a series of dots marked on their frontal surfaces are shown. In red, the reference XFEM model. Superimposed in blue, on the left, the PGD model using the PRB formulation; on the right, the PGD model using the SRB formulation.

3.5.2.1 Finite element model of the cornea

The model of the cornea was meshed using trilinear hexahedral elements. It consists of 8 514 nodes and 6 840 elements. Two views of the mesh are shown in Figure 3.5. The model is clamped at its base, resulting in a dome-like geometry.

In accordance to the vast majority of the literature, corneal tissue is assumed as hyperelastic [90, 173, 174]. However, for simplicity of exposition, a linear elastic behaviour is assumed in this example. The material properties of the cornea considered in this example are $E = 2$ MPa, and $\nu = 0.48$ [173]. More sophisticated material behaviours can also be efficiently considered. For instance, Niroomandi et al. developed in [89] the PGD formulation of a St. Venant–Kirchhoff material, with which it is possible to obtain a computational vademecum.

3.5.2.2 Radial incisions on the cornea

Under the assumptions commented before, the cornea model is subjected to different patterns of radial spoke-shaped incisions. See Fig. 3.6 for different frames of the cutting procedure.

PGD modes, used as a Ritz basis for the problem, are shown in Fig. 3.5.2.2. The portion of the cornea that could suffer a cut is a strip of 3×18 nodes around the area where the incision is made. 54 modes are required to reproduce the cutting procedure with an error tolerance in the PGD algorithm of 10^{-4}. In general, the obtained results produce a very realistic sensation, both from their appearance and from the perceived haptic response: the method very much

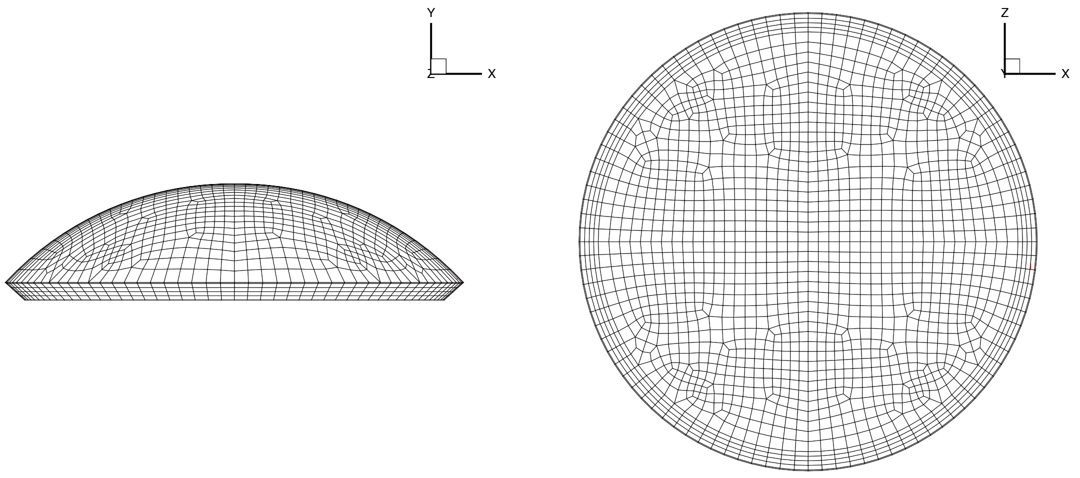


Figure 3.5. Geometry of the finite element model of the human cornea [173].

improves the quality of a previous approach [83] in which XFEM and POD were coupled, and ran only under visual constraints (around 25 Hz). L_2-error norms are computed by taking a full XFEM simulation (with no PGD) as a reference solution. These error norms are reported in Fig. 3.8. In all our simulations, the reported error versus a full XFEM simulation remained below $7 \cdot 10^{-3}$.

3.5.2.3 Timings

The results presented in this chapter have been obtained with a 64-bit laptop, with a 2.5 GHz Intel Core i5 processor, and 4 GB RAM, running Matlab 2014b on Windows 7. In spite of the use of non-optimised Matlab code, the examples of the cornea run with a response rate which is always faster than 1 kHz. This value is sufficient for meeting both the visual and haptic real-time requirements. A summary of the performed tests, involving different cut lengths, is shown in Fig. 3.9. In this example, in which up to sixteen nodes can take part in the cut, the results seem to be almost independent of the length of the cut. It is assumed that there should exist some limit from which the response rate is slower than the required haptic constraints.

3.6 Conclusions

In this chapter, a new method for the simulation of surgical cutting under real-time constraints, valid for haptic environments, is developed. This method, for which two different approaches are formulated (PRB and SRB), is based on

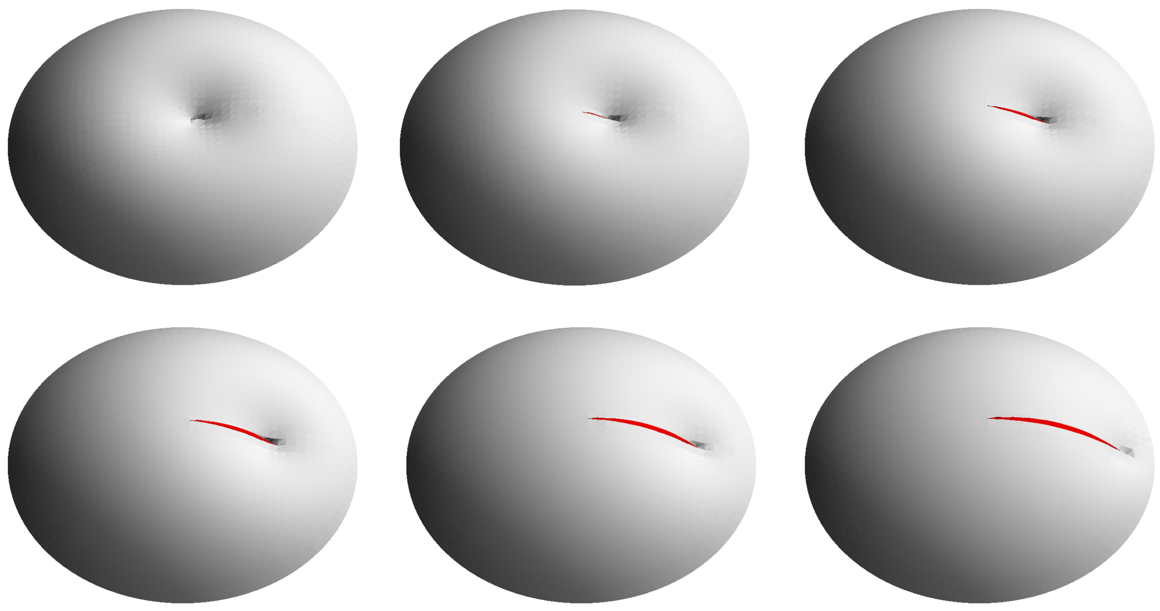


Figure 3.6. Six frames showing the radial cutting procedure in a rendered model (from left to right and from top to bottom). The interior of the cut has been coloured in red, and its amplitude has been artificially increased by 100 times to make it visible.

the combination of PGD computational vademecums with the cracking node method.

On the one hand, a computational vademecum of the displacements of the model is obtained for any possible contact position of the scalpel, covering the response of the organ (in this case, the human cornea). This vademecum is then enriched, at run-time, with discontinuous XFEM-like shape functions. This enrichment cannot be reasonably covered by a single vademecum, since the results for any possible position, orientation, and length of a cut cannot be easily compressed. In other words, the problem is not *separable* in the form given by Eq. (2.6).

In this work, computational vademecums introduce two notable features. The first one is the use of the multi-dimensional PGD results *à la* POD, which allows to solve the problem by a Galerkin projection onto a multidimensional basis. The second one is that these bases are obtained by PGD techniques, and are much richer than those obtained by POD (such as in [83]). Moreover, they do not require the computation of snapshots, i. e. solutions of complete problems under different loading conditions (see § 4.5.1).

On the other hand, the use of the cracking node method [165], a particular application of the XFEM technique, greatly simplifies the treatment of the discontinuous enrichments, thus achieving substantial CPU savings. To this end,

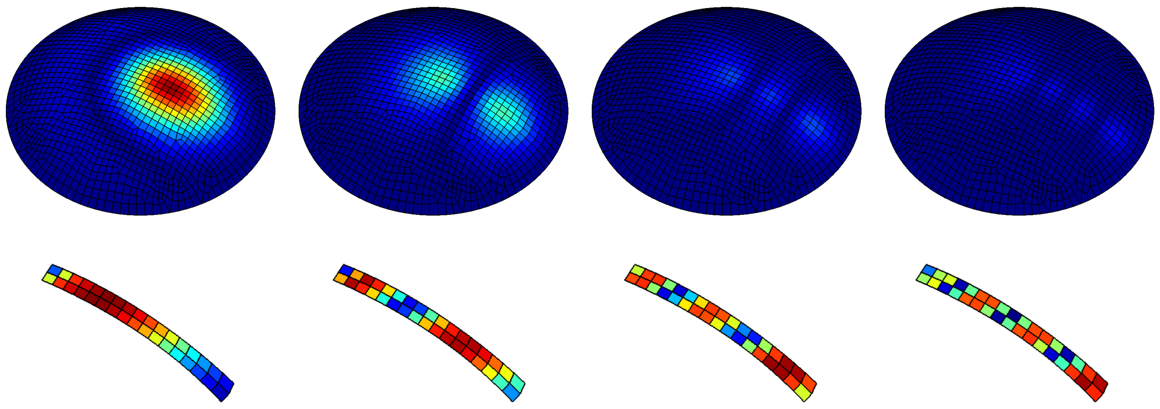


Figure 3.7. Normalised values of the first four PGD modes (from left to right) obtained in the simulation of radial incision. Top images show the spatial modes; bottom images show the modes of the load position. Scales of top and bottom images are not the same.

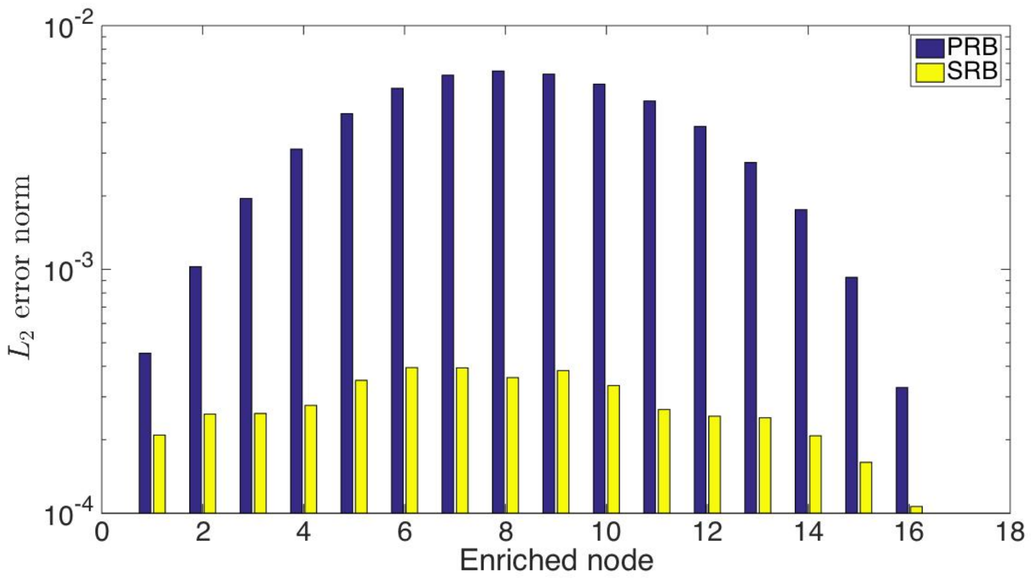


Figure 3.8. L_2 error norm of the simulation shown in Fig. 3.8 as the cut advances. Errors are measured with respect to an XFEM reference model to which the same cut is applied. PRB and SRB formulations (blue and yellow, respectively) are represented. The abscissa axis shows the number of cut (or enriched) nodes.

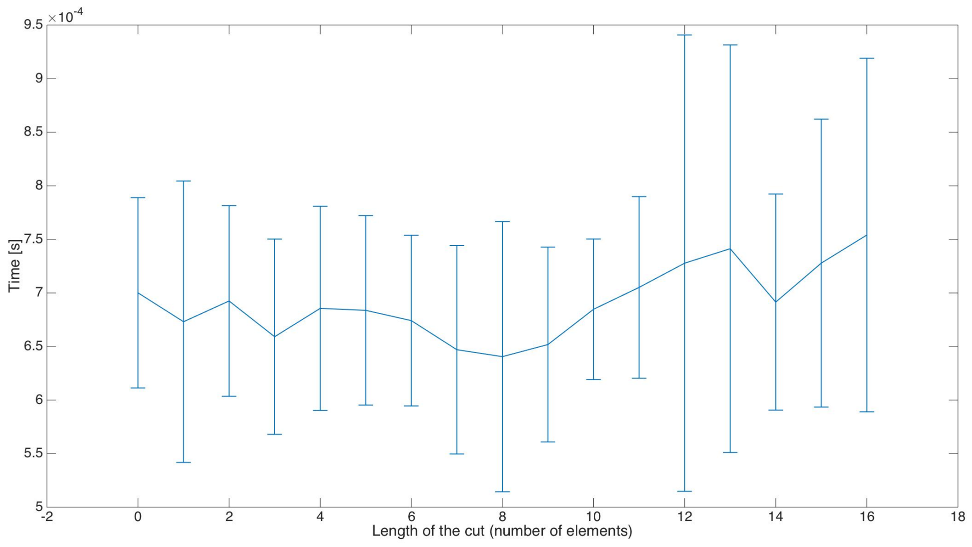


Figure 3.9. Statistics of the computing time required by a simulation with respect to the cut length. Note that all the examples run below 1 ms.

many of the matrices needed to perform the simulation are stored in memory, as explained in § 3.2.2.

With regard to the two proposed approaches, the results obtained with the SRB formulation have lower levels of error than those computed with the PRB formulation. However, while the SRB formulation requires the computation of different sub-matrices $\boldsymbol{K}^{\beta\beta}$ and $\boldsymbol{K}^{q\beta}$ for each load position in the on-line resolution equation Eq. (3.6), the PRB formulation uses the same sub-matrices in all cases. The reduced size of the matrices in the case of the SRB formulation, though, does not affect the storage in memory to any great extent.

In sum, the successful combination of both PGD bases (used as POD) and the cracking node method allows to achieve very efficient real-time simulations. This method has been tested on a *Phantom Omni* haptic peripheral by *Geomagic Sensable* providing excellent results.

Chapter 4

Simulation of tissue tearing

4.1 Introduction

A method for the real-time simulation of tearing of soft tissues is presented. It uses a parametric formulation of the continuum damage mechanics equations, and combines the model order reduction methods of proper generalised decomposition (PGD) and proper orthogonal decomposition (POD). The key idea of the method consists in generating a computational vademecum which can be regarded as a sort of direct **time integrator** [175]. This vademecum allows the system to behave as a black box, in which the inputs are fed with the damage field of the last computed step, and the outputs provide the solution for the incoming step. This means that, for each step, a new problem has to be solved, in which the values of the damage field obtained in the last step are considered the **initial conditions** with which to compute the subsequent step.

Such a time integrator is achieved by posing the problem in a parametric framework. Thus, the initial conditions of each step should be contemplated as parameters of the problem. However, under this premise, the number of parameters can be enormous. There may be several values of the damage field per element, and in the case of surgical simulators, the models may consist of thousands of elements. In practise, it is essentially infeasible for PGD to cope with a system with tens (or hundreds) of thousands of parameters—again, an issue that can be attributed to the already mentioned *curse of dimensionality*. To alleviate this problem, a reduced-order basis of the damage field is obtained using POD. This MOR method guarantees an efficient parametrisation of the space of the initial conditions (the damage field) with only a few parameters. Still, the number of parameters obtained following this approach may reach some tens.

The computing workload is distributed following an off-line–on-line strategy (see § 2.2). The computational vademecum is solved off-line, once and for all, for *any* value of the parameters. Then, the on-line stage can be executed under severe real-time constraints by simply feeding the vademecum with the results of the previous computed step. The approach presented for solving this quasi-static problem uses an explicit incremental formulation. In principle, the (pseudo-)time steps of this incremental formulation need not be of the same size. Instead, they can be as small as required to satisfy the stability constraints imposed by the solver, provided that the magnitude of the applied load is also considered as another parameter of the problem.

In short, the objective of this chapter is to develop appropriate numerical techniques for the real-time surgical simulation of tearing procedures. To that end, a computational vademecum of the system, subjected to any type of force and initial conditions of the damage field, is considered. Each of the aspects of this method is explained in greater detail in the following sections.

4.2 Model of damage

Damage can be defined as the progressive physical process by which a material breaks—and the study (using mechanical variables) of these processes, when the materials are subjected to a load, is made by damage mechanics [176]. Damage shows different phenomena depending on the observed scale. At a micro-scale level, damage can be viewed as an accumulation of micro-stresses in the vicinity of defects which eventually harm the material (by means of molecular bond breaking, dislocations, or micro-crack generation). At a meso-scale level, damage represents the growth and merging of micro-cracks or micro-voids which, together, start one crack. Lastly, at a macro-scale level, damage stands for the growth of that crack. It should be noted that continuum mechanics variables such as stress, strain, or damage can describe phenomena only at the meso-scale and macro-scale levels [176, 177].

Although damage mechanics is commonly studied in metals [177], a model of damage for solving the problem of tearing of soft tissues is described here. The use of a model damage in this approach is justified by the fact that the torn tissue exhibits a behaviour 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. These instruments generate a continuous increasing degradation of the tissue which eventually breaks apart.

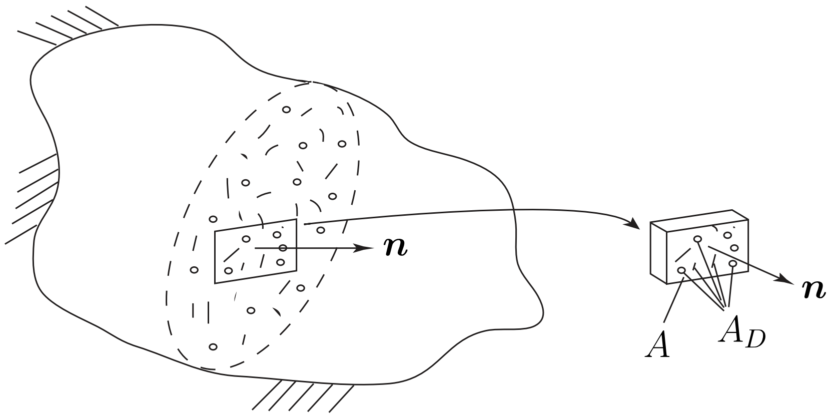


Figure 4.1. A damaged element showing areas A and A_D along with the normal vector $\boldsymbol{n}$. Figure adapted from [177].

Therefore, an approach based on damage mechanics can be considered valid for simulating this process of tissue degradation.

The model of damage used in this approach, together with a proposed finite element formulation for its implementation, are described in the following sections.

4.2.1 Damage variable

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 [177]. A commonly used quantity to describe damage, first defined in [178], 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 (see Fig 4.1). If micro-stress concentrations and defect interactions are handled carefully, this damage variable D is well suited for continuum mechanics [177]. For this reason, this is the definition of damage variable used in the presented approach.

If damage is considered as isotropic (i. e. cracks and voids are distributed uniformly in all directions), as is the case, then D is a scalar value; otherwise, the damage variable depends on the orientation $\boldsymbol{n}$, and a damage vector or tensor is required. The scalar values in the isotropic case range from zero to one, where $D = 0$ characterises an undamaged (virgin) state, $D \leq 1$ represents the initiation of a macro-crack, and $D = 1$ describes a fully damaged state.

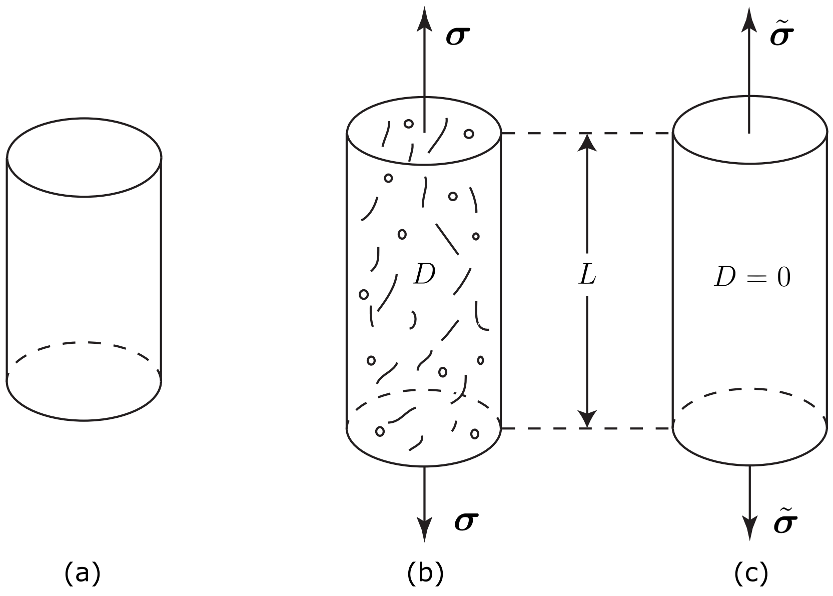


Figure 4.2. Different configurations of the damaged material: (a) virgin material, (b) damaged material, (c) equivalent virgin material. Figure adapted from [177].

4.2.2 Effective stress

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

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

4.2.3 Principle of strain equivalence

This principle 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. 4.2). For example, the linear elastic law of a damaged material can be written as follows:

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

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 [176, 177].

4.2.4 State potential

Lemaitre [176] formulated the thermodynamic constitutive equations for coupling elasto-plasticity and damage at the meso-scale level. This model uses the **method of local state** (which proposes that the laws that are valid for a macroscopic system are also valid for infinitesimal parts of it), and **state variables** that represent the effects of damage on the stiffness of the material.

Within the hypothesis of small strains and small displacements, the state variables that are taken into account in this Chapter 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 or plasticity tensors, 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 generalised 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

$$\boldsymbol{\sigma} = \frac{\partial \boldsymbol{\Psi}}{\partial \boldsymbol{\varepsilon}} \qquad \text{and} \qquad \bar{Y} = \frac{\partial \boldsymbol{\Psi}}{\partial D}. \tag{4.1}$$

4.2.5 Evolution of damage

The evolution of the damage described in this approach is based on the three-dimensional isotropic damage model developed by Simó in [179], 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. Thus, an equivalent strain energy can be defined by the expression

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

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

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

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

$$\varPhi_T \leq \varPhi^m\,,$$

with $\varPhi_T$ being the equivalent strain energy at current time, $\varPhi_T = \sqrt{2\ \boldsymbol{\varPsi}_0\left(\boldsymbol{\varepsilon}(T)\right)}$. The case in which $\varPhi_T = \varPhi^m$ and, simultaneously, a load is applied from a damage 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

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

Here, $h(\varPhi, D)$ is a given function that characterises the damage process in the material. According to Simó [179], if h does not depend on D, the isotropic damage model can be expressed in the following equivalent form,

$$\boldsymbol{\sigma}(T) = g(\varPhi^m)\ \frac{\partial\boldsymbol{\varPsi}_0\left(\boldsymbol{\varepsilon}(T)\right)}{\partial\boldsymbol{\varepsilon}}, \tag{4.2}$$

with $g(\varPhi^m)$ being a potential of dissipation. By applying the principle of strain equivalence to the stress tensor in Eq. (4.1), and comparing it with Eq. (4.2), it is clear that $g = 1 - D$, which can be followed by setting

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

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

$$g(\varPhi^m) = 1 - D = \beta + (1-\beta)\,\frac{1 - \mathrm{e}^{-\varPhi^m/\alpha}}{\varPhi^m/\alpha}, \tag{4.3}$$

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

$$h(\varPhi^m) = -\frac{\mathrm{d}g(\varPhi^m)}{\mathrm{d}\varPhi^m} = \frac{1-\beta}{(\varPhi^m)^2}\left(\alpha - \mathrm{e}^{-\frac{\varPhi^m}{\alpha}}(\alpha + \varPhi^m)\right). \tag{4.4}$$

With all these definitions, it is possible to completely determine an expression for the stress tensor $\boldsymbol{\sigma}$. From Eq. (4.2), 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:

$$\boldsymbol{\sigma}(t) = \begin{cases} \left(g(\Phi^m)\, \mathbf{C}_0 - \dfrac{h(\Phi^m)}{\Phi^m} \left(\boldsymbol{\sigma}_0 \otimes \boldsymbol{\sigma}_0 \right) \right) : \boldsymbol{\varepsilon}, & \text{if } \Phi_T = \Phi^m \text{ and loading,} \\ g(\Phi^m)\, \mathbf{C}_0 : \boldsymbol{\varepsilon}, & \text{otherwise.} \end{cases} \tag{4.5}$$

4.3 Finite element formulation

The damage evolution described in the previous section is implemented using an incremental approach, which requires an explicit formulation of the finite element method. The basic idea behind this approach is to compute the simulation in a stepwise approach so that the result of the last computed step can be re-used as an initial condition for computing the following step.

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

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

where λ and μ are the Lamé parameters [180].

To obtain the explicit formulation, 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

$$\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}^*_{t+\Delta t} : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}_{t+\Delta t} \, \mathrm{d}\Omega = \int_{\Gamma_{t2}} \boldsymbol{u}^*_{t+\Delta t} \cdot \boldsymbol{f}_{t+\Delta t} \, \mathrm{d}\Gamma, \tag{4.6}$$

where the subscript $t + \Delta t$ denotes the following step, and $\boldsymbol{f}$ is the surface traction. The variables $\boldsymbol{u}$ and $\boldsymbol{f}$, 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. Correspondingly, the admissible variation of $\boldsymbol{u}_{t+\Delta t}$ can be obtained as

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

Thus, Eq. (4.6) 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

$$\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \Delta \boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \boldsymbol{u}_t \, \mathrm{d}\Omega + \int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \Delta \boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \Delta \boldsymbol{u} \, \mathrm{d}\Omega = \\ = \int_{\Gamma_{t2}} \boldsymbol{\nabla}_{\mathrm{s}} \Delta \boldsymbol{u}^* \cdot \boldsymbol{f}_t \, \mathrm{d}\Gamma + \int_{\Gamma_{t2}} \boldsymbol{\nabla}_{\mathrm{s}} \Delta \boldsymbol{u}^* \cdot \Delta \boldsymbol{f} \, \mathrm{d}\Gamma. \quad (4.7)$$

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 a known relation from the previous step, both terms cancel each other in Eq. (4.7). As a consequence, the incremental formulation can be finally expressed as

$$\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}} \Delta \boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}} \Delta \boldsymbol{u} \, \mathrm{d}\Omega = \int_{\Gamma_{t2}} \boldsymbol{\nabla}_{\mathrm{s}} \Delta \boldsymbol{u}^* \cdot \Delta \boldsymbol{f} \, \mathrm{d}\Gamma. \quad (4.8)$$

From Eq. (4.8), the matrix expression $\boldsymbol{K}_t \cdot \Delta \boldsymbol{u} = \Delta \boldsymbol{f}$ can be derived, where $\boldsymbol{K}_t$ represents the global stiffness matrix of the problem. It is a symmetric square sparse matrix with as many rows (or columns) as degrees of freedom of the model.

The proposed incremental procedure to solve the damage evolution of a model using standard FEM is shown in Algorithm 4.1. It is important to remark that, since variables g and h constitute a scalar field, their values have to be computed at different points of the model to obtain an accurate representation of damage. For simplicity, the chosen points have been taken to coincide with the Gauss integration points of each element, which are used to compute the elasticity tensor $\mathbf{C}_t$ (line 5), and each of the elemental stiffness matrices $\boldsymbol{K}^e_t$ that lead to $\boldsymbol{K}_t$ (line 7). Similarly, variables that allow to check changes in the evolution of damage, such as Ψ_t, Φ_t, or Φ^m (lines 12 and 14), are also computed at the Gauss integration points.

Another remark can be made concerning the relationship between the load increment $\Delta \boldsymbol{f}$ and the time increment Δt. The proposed algorithm is intended to be used in a simulator, driven by a haptic device. For each Δt, the algorithm has to translate the force exerted by the user to a value of $\Delta \boldsymbol{f}$ (line 3), which can give rise to a loading state, a neutral loading, or an unloading state. For a given Gauss integration point, whenever the value of $\varPhi_t$ exceeds the threshold $\varPhi^m$ as a consequence of a loading state, the values of the damage variables g and h are updated (lines 15 and 16). Otherwise, g maintains the value of the last step, and h is set to zero, as established by Eq. (4.5).

```
1  **initialisation:** $t = 0$, $\boldsymbol{u}_t = \boldsymbol{0}$, $g_t^p = 1$, $h_t^p = 0$, $\varPhi_t^p = 0$, $(\varPhi^m)^p = 0$, $\boldsymbol{\sigma}_t^p = \boldsymbol{0}$,
     for each Gauss point $p$ ;
2  **while** *$t$ has not reached the end of simulation* **do**
3      determine $\Delta \boldsymbol{f}$ from $\Delta t$ ;
4      **for** *each Gauss point $p$* **do**
5          $\mathbf{C}_t^p = g_t^p \cdot \mathbf{C}_0 + h_t^p \cdot (\boldsymbol{\sigma}_0^p \otimes \boldsymbol{\sigma}_0^p)_t$ ;
6      **end**
7      $\boldsymbol{K}_t = \int_\Omega \boldsymbol{B}^{\mathrm{T}} \cdot \mathbf{C}_t \cdot \boldsymbol{B}^{\mathrm{T}} \mathrm{d}\Omega$ ;
8      $\Delta \boldsymbol{u} = \boldsymbol{K}_t^{-1} \cdot \Delta \boldsymbol{f}$ ;
9      $\boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_t + \Delta \boldsymbol{u}$ ;
10     **for** *each Gauss point $p$* **do**
11         $\boldsymbol{\varepsilon}_{t+\Delta t}^p = \boldsymbol{B} \cdot \boldsymbol{u}_{t+\Delta t}$ ;
12         compute $\varPsi_{t+\Delta t}^p$ and $\varPhi_{t+\Delta t}^p$ ;
13         **if** $\varPhi_{t+\Delta t}^p > (\varPhi^m)^p$ **then**
14             $(\varPhi^m)^p = \varPhi_{t+\Delta t}^p$ ;
15             compute new values of $g_{t+\Delta t}^p$ from Eq. (4.3) ;
16             compute new values of $h_{t+\Delta t}^p$ from Eq. (4.4) ;
17         **else**
18             $g_{t+\Delta t}^p = g_t^p$, $h_{t+\Delta t}^p = 0$ ;
19         **end**
20         $(\boldsymbol{\sigma}_0)_{t+\Delta t}^p = \mathbf{C}_0 \cdot \boldsymbol{\varepsilon}_{t+\Delta t}^p$ ;
21     **end**
22     $t = t + \Delta t$ ;
23 **end**
```

Algorithm 4.1. Explicit formulation of the FEM algorithm.

However, this algorithm is 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 $\boldsymbol{K}_t^{-1}$ (line 8). 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). In consequence, although this algorithm is effective, it is not sufficiently efficient to be compatible with the real-time requirements of surgical simulators. In the following sections, other strategies are developed that improve this algorithm. Section 4.4 analyses an approach using PGD, while § 4.5 describes a method that combines this mentioned PGD approach with POD model reduction techniques.

4.4 PGD approximation

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.

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 vectors of Gauss point values, $\boldsymbol{g}_n$ and $\boldsymbol{h}_n$, at time step t_n, can be addressed in a parametric, multidimensional form. 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 the phase space

$$\Delta\boldsymbol{u} : \Omega \times \mathcal{G}_1 \times \ldots \times \mathcal{G}_{\mathtt{ngp}} \times \mathcal{H}_1 \times \ldots \times \mathcal{H}_{\mathtt{ngp}} \times \bar{\Gamma} \to \mathbb{R}^3, \tag{4.9}$$

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 current whole body) as parameters of the formulation implies to consider their values at every Gauss point of the model. For meshes of practical interest, the number of resulting parameters will be prohibitive even for PGD techniques. This problem will be addressed later. Until then, to obtain a parametric solution for any initial damage condition, within these intervals, the weak form of the problem, Eq. (4.8), has to be redefined as follows:

$$\int_{\bar{\Gamma}} \int_{\mathcal{H}_{\mathrm{ngp}}} \cdots \int_{\mathcal{H}_1} \int_{\mathcal{G}_{\mathrm{ngp}}} \cdots \int_{\mathcal{G}_1} \int_{\Omega} \mathcal{U} \;\mathrm{d}\Omega \,\mathrm{d}\mathcal{G}_1 \cdots \mathrm{d}\mathcal{G}_{\mathrm{ngp}} \,\mathrm{d}\mathcal{H}_1 \cdots \mathrm{d}\mathcal{H}_{\mathrm{ngp}} \,\mathrm{d}\bar{\Gamma} = \int_{\bar{\Gamma}} \int_{\Gamma_{t_2}} \Delta \boldsymbol{u}^* \cdot \Delta \boldsymbol{f} \,\mathrm{d}\Gamma \,\mathrm{d}\bar{\Gamma}, \quad (4.10)$$

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. (4.5) as

$$\mathbf{C} = g\, \mathbf{C}_0 - \frac{h}{\Phi^m} \left(\boldsymbol{\sigma} \otimes \boldsymbol{\sigma} \right). \quad (4.11)$$

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

$$\Delta u_j^n(\boldsymbol{x}; g_1, \ldots, g_{\mathrm{ngp}}, h_1, \ldots, h_{\mathrm{ngp}}, \boldsymbol{s}) = \sum_{i=1}^{n} F_j^i(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathrm{ngp}} G_{Kj}^{\,i}(g_K) \cdot \prod_{K=1}^{\mathrm{ngp}} H_{Kj}^{\,i}(h_K) \cdot J_j^i(\boldsymbol{s}).$$

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

$$\Delta u_j^{n+1}(\boldsymbol{x}; g_1, \ldots, g_{\mathrm{ngp}}, h_1, \ldots, h_{\mathrm{ngp}}, \boldsymbol{s}) = \Delta u_j^n(\boldsymbol{x}; g_1, \ldots, g_{\mathrm{ngp}}, h_1, \ldots, h_{\mathrm{ngp}}, \boldsymbol{s}) + R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathrm{ngp}} S_{Kj}(g_K) \cdot \prod_{K=1}^{\mathrm{ngp}} T_{Kj}(h_K) \cdot U_j(\boldsymbol{s}).$$

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

$$\begin{aligned}
\Delta u_j^*(\boldsymbol{x}; g_1, \ldots, g_{\mathrm{ngp}}, h_1, \ldots, h_{\mathrm{ngp}}, \boldsymbol{s}) &= R_j^*(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathrm{ngp}} S_{Kj}(g_K) \cdot \prod_{K=1}^{\mathrm{ngp}} T_{Kj}(h_K) \cdot U_j(\boldsymbol{s}) \\
&+ R_j(\boldsymbol{x}) \cdot S_{1j}^*(g_1) \cdot \ldots \cdot S_{\mathrm{ngp}_j}(g_{\mathrm{ngp}}) \cdot \prod_{K=1}^{\mathrm{ngp}} T_{Kj}(h_K) \cdot U_j(\boldsymbol{s}) + \ldots \\
&+ R_j(\boldsymbol{x}) \cdot S_{1j}(g_1) \cdot \ldots \cdot S_{\mathrm{ngp}_j}^*(g_{\mathrm{ngp}}) \cdot \prod_{K=1}^{\mathrm{ngp}} T_{Kj}(h_K) \cdot U_j(\boldsymbol{s}) \\
&+ R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathrm{ngp}} S_{Kj}(g_K) \cdot T_{1j}^*(h_1) \cdot \ldots \cdot T_{\mathrm{ngp}_j}(h_{\mathrm{ngp}}) \cdot U_j(\boldsymbol{s}) + \ldots \\
&+ R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathrm{ngp}} S_{Kj}(g_K) \cdot T_{1j}(h_1) \cdot \ldots \cdot T_{\mathrm{ngp}_j}^*(h_{\mathrm{ngp}}) \cdot U_j(\boldsymbol{s})
\end{aligned}$$

$$+ R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathrm{ngp}} S_{Kj}(g_K) \cdot \prod_{K=1}^{\mathrm{ngp}} T_{Kj}(h_K) \cdot U_j^*(\boldsymbol{s}).$$

By substituting the approximations $\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{u}^*$ into the weak form of the problem, Eq. (4.10), a computational vademecum is obtained that can suppress lines 4–8 of Algorithm 4.1. Instead, $\Delta\boldsymbol{u}$ can be obtained almost automatically just by particularising the computational vademecum with the values of $\boldsymbol{x}, g_1, \ldots, g_p, h_1, \ldots, h_p$, 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. The following section details the reduction of this complexity.

4.5 Reduction of the parametrisation of the initial conditions

Standard finite element techniques compute the value of damage variables at the Gauss integration points. As mentioned before, considering each one of these values in meshes of hundreds of thousands of nodes is prohibitive. Therefore, it seems reasonable to optimise somehow the parametrisation of the space of initial conditions. To achieve this, an optimal basis is constructed using POD from the results of complete similar problems.

4.5.1 Proper Orthogonal Decomposition

The **Proper Orthogonal Decomposition** (POD) is a statistical procedure that transforms a series of observations (the so-called *snapshots*) of possibly correlated variables into a set of orthogonal vectors of linearly uncorrelated variables. The number of resulting vectors is less than or equal to the number of original snapshots.

To describe the computation of a POD orthogonal basis set, assume that a series of numerical simulations have been performed off-line (the snapshots) and, for each of them, the value of a certain field of interest is known. By way of example, the damage field $g(\boldsymbol{x}, t)$ is considered here, which is expressed in a discrete form, depending on both certain positions of the model $\boldsymbol{x}_i$ and on some instants of time $t_m = m \cdot \Delta t$, with $i \in [1, \ldots, M]$ and $m \in [0, \ldots, P]$. For the sake of simplicity, the notation $g(\boldsymbol{x}_i, t_m) \equiv g^m(\boldsymbol{x}_i) \equiv g_i^m$ is used, and $\boldsymbol{g}^m$ is defined as the vector of nodal values g_i^m at time t_m. The key goal of POD is to find the most characteristic structure $\phi(\boldsymbol{x})$ amongst these $g^m(\boldsymbol{x})$, $\forall m$ [144]. To that end, it is necessary to maximise the scalar quantity

$$\alpha = \frac{\sum_{m=1}^{P} \left(\sum_{i=1}^{M} \phi(\boldsymbol{x}_i) \cdot g^m(\boldsymbol{x}_i) \right)^2}{\sum_{i=1}^{M} \left(\phi(\boldsymbol{x}_i) \right)^2}.$$

The maximisation leads to

$$\sum_{m=1}^{P} \left(\sum_{i=1}^{M} \tilde{\phi}(\boldsymbol{x}_i) \cdot g^m(\boldsymbol{x}_i) \right) \cdot \left(\sum_{j=1}^{M} \phi(\boldsymbol{x}_j) \cdot g^m(\boldsymbol{x}_j) \right) = \alpha \sum_{i=1}^{M} \tilde{\phi}(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_i),$$

for all $\tilde{\phi}$. This equation can be rewritten as

$$\sum_{i=1}^{M} \left(\sum_{j=1}^{M} \left(\sum_{i=1}^{P} g^m(\boldsymbol{x}_i) \cdot g^m(\boldsymbol{x}_j) \cdot \phi(\boldsymbol{x}_j) \right) \tilde{\phi}(\boldsymbol{x}_i) \right) = \alpha \sum_{i=1}^{M} \tilde{\phi}(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_i),$$

for all $\tilde{\phi}$. This is equivalent to solve the eigenvalue problem

$$\tilde{\boldsymbol{\phi}}^{\mathrm{T}}\, \boldsymbol{c}\, \boldsymbol{\phi} = \alpha\, \tilde{\boldsymbol{\phi}}^{\mathrm{T}}\, \boldsymbol{\phi},$$
$$\boldsymbol{c}\, \boldsymbol{\phi} = \alpha\, \boldsymbol{\phi},$$

with the vector $\boldsymbol{\phi}$ having $\phi(\boldsymbol{x}_i)$ as the i-th component, and $\boldsymbol{c}$ being the two-point correlation matrix

$$c_{ij} = \sum_{m=1}^{P} g^m(\boldsymbol{x}_i) \cdot g^m(\boldsymbol{x}_j); \qquad \boldsymbol{c} = \sum_{m=1}^{P} \boldsymbol{g}^m \cdot (\boldsymbol{g}^m)^{\mathrm{T}}, \tag{4.12}$$

which is symmetric and positive definite. If the matrix $\boldsymbol{Q}$ is defined as

$$\boldsymbol{Q} = \begin{pmatrix} g_1^1 & g_1^2 & \cdots & g_1^P \\ g_2^1 & g_2^2 & \cdots & g_2^P \\ \vdots & \vdots & \ddots & \vdots \\ g_M^1 & g_M^2 & \cdots & g_M^P \end{pmatrix},$$

then the matrix $\boldsymbol{c}$ in Eq. (4.12) results in

$$\boldsymbol{c} = \boldsymbol{Q} \cdot \boldsymbol{Q}^{\mathrm{T}}.$$

To obtain a reduced-order model from a set of snapshots, the eigenvalue problem in Eq. (4.12) should be solved. The N eigenvectors $\boldsymbol{\phi}_i$ associated to the highest eigenvalues α_i can be selected according to the requirements of the

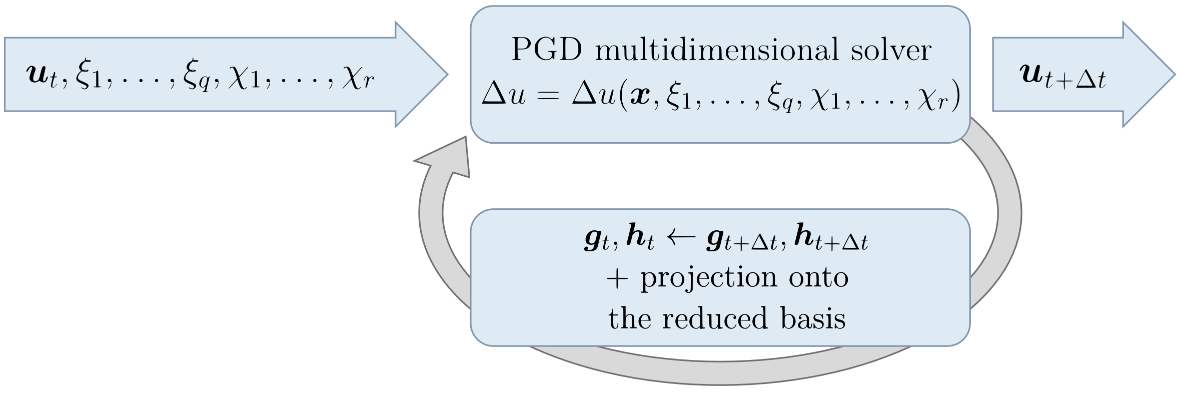

$\boldsymbol{u}_t, \xi_1, \ldots, \xi_q, \chi_1, \ldots, \chi_r$
PGD multidimensional solver
$\Delta u = \Delta u(\boldsymbol{x}, \xi_1, \ldots, \xi_q, \chi_1, \ldots, \chi_r)$
$\boldsymbol{u}_{t+\Delta t}$
$\boldsymbol{g}_t, \boldsymbol{h}_t \leftarrow \boldsymbol{g}_{t+\Delta t}, \boldsymbol{h}_{t+\Delta t}$
+ projection onto
the reduced basis

Figure 4.3. Diagram of the proposed algorithm. A POD reduced basis is used for the parametrisation of the space of initial conditions. Figure adapted from [175].

problem and used to approximate the solution $g^m(\boldsymbol{x})$, $\forall m$. The orthogonal basis set can be expressed as the following matrix $\boldsymbol{B} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N]$, i. e.

$$\boldsymbol{B} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_1) & \cdots & \phi_N(\boldsymbol{x}_1) \\ \phi_1(\boldsymbol{x}_2) & \phi_2(\boldsymbol{x}_2) & \cdots & \phi_N(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_M) & \phi_2(\boldsymbol{x}_M) & \cdots & \phi_N(\boldsymbol{x}_M) \end{pmatrix}.$$

4.5.2 POD–PGD approach

In this section, a combined POD–PGD approach is considered [175]. A suitable parametrisation of the space of initial conditions can be obtained by applying the POD decomposition to a set of solutions from similar problems. The POD decomposition provides an optimal basis (in a given norm) onto which 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. A diagram of the method is shown in Fig. 4.3.

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

$$\boldsymbol{g} = \sum_{i=1}^{q} \xi_i \cdot \boldsymbol{v}_i \qquad \text{and} \qquad \boldsymbol{h} = \sum_{j=1}^{r} \chi_j \cdot \boldsymbol{\rho}_j, \tag{4.13}$$

where $\boldsymbol{v}_i$ 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 $q, r \ll \texttt{ngp}$ (for practical purposes, not higher than ten), and not necessarily $q = r$.

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

$$\int_{\bar{\Gamma}} \int_{\mathcal{X}_r} \cdots \int_{\mathcal{X}_1} \int_{\mathcal{E}_q} \cdots \int_{\mathcal{E}_1} \int_{\Omega} \mathcal{U} \, \mathrm{d}\Omega \, \mathrm{d}\mathcal{E}_1 \cdots \mathrm{d}\mathcal{E}_q \, \mathrm{d}\mathcal{X}_1 \cdots \mathrm{d}\mathcal{X}_r \, \mathrm{d}\bar{\Gamma} = \int_{\bar{\Gamma}} \int_{\Gamma_{t2}} \Delta \boldsymbol{u}^* \cdot \Delta \boldsymbol{t} \, \mathrm{d}\Gamma \, \mathrm{d}\bar{\Gamma}, \quad (4.14)$$

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

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

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

$$\Delta u_j^{n+1}(\boldsymbol{x}; \xi_1, \ldots, \xi_q, \chi_1, \ldots, \chi_r, \boldsymbol{s}) = \Delta u_j^n(\boldsymbol{x}; \xi_1, \ldots, \xi_q, \chi_1, \ldots, \chi_r, \boldsymbol{s}) + R_j(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\rho}) \cdot \prod_{K=1}^{q} S_{Kj}(\xi_K) \cdot \prod_{L=1}^{r} T_{Lj}(\chi_L) \cdot U_j(\boldsymbol{s}). \quad (4.15)$$

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

$$\begin{aligned}
\Delta u_j^*(\boldsymbol{x}; \xi_1, \ldots, \xi_q, \chi_1, \ldots, \chi_r, \boldsymbol{s}) &= R_j^*(\boldsymbol{x}) \cdot \prod_{K=1}^{q} S_{Kj}(g_K) \cdot \prod_{L=1}^{r} T_{Lj}(h_L) \cdot U_j(\boldsymbol{s}) \\
&+ R_j(\boldsymbol{x}) \cdot S_{1}{}_j^*(g_1) \cdot \ldots \cdot S_{q_j}(g_q) \cdot \prod_{L=1}^{r} T_{Lj}(h_L) \cdot U_j(\boldsymbol{s}) + \ldots \\
&+ R_j(\boldsymbol{x}) \cdot S_{1j}(g_1) \cdot \ldots \cdot S_{q}{}_j^*(g_q) \cdot \prod_{L=1}^{r} T_{Lj}(h_L) \cdot U_j(\boldsymbol{s}) \\
&+ R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{q} S_{Kj}(g_K) \cdot T_{1}{}_j^*(h_1) \cdot \ldots \cdot T_{rj}(h_r) \cdot U_j(\boldsymbol{s}) + \ldots \\
&+ R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{q} S_{Kj}(g_K) \cdot T_{1j}(h_1) \cdot \ldots \cdot T_{r}{}_j^*(h_r) \cdot U_j(\boldsymbol{s})
\end{aligned}$$

$$+ R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{q} S_{Kj}(g_K) \cdot \prod_{L=1}^{r} T_{Lj}(h_L) \cdot U_j^*(\boldsymbol{s}).$$

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. Consequently, the explicit algorithm of the on-line stage (see Algorithm 4.2) also requires some modifications.

```
1 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 $p$ ;
2 while $t$ has not reached the end of simulation do
3     determine $\Delta \boldsymbol{f}$ from $\Delta t$ ;
4     obtain $\xi_t$ by $L_2$ projection of $\boldsymbol{g_t}$ onto $\boldsymbol{\upsilon}$ ;
5     obtain $\chi_t$ by $L_2$ projection of $\boldsymbol{h_t}$ onto $\boldsymbol{\rho}$ ;
6     particularise computational vademecum to obtain $\Delta \boldsymbol{u}$ ;
7     $\boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_t + \Delta \boldsymbol{u}$ ;
8     follow lines 10–22 of Algorithm 4.1 ;
9 end
```

Algorithm 4.2. Explicit formulation of the POD–PGD algorithm.

4.5.3 Simplification of the model

A simplification of the damage model is described here, which will be used in the following sections for obtaining the matrix formulation (§ 4.5.4), and developing numerical examples (§ 4.6). It consists in considering that the function $h(\Phi^m)$ that characterises the damage process, Eq. (4.4), is zero in all cases. In this way, $h(\Phi^m) = 0$ not only when $\Phi_T \leq \Phi^m$, but also when $\Phi_T > \Phi^m$ (check lines 13–19 of Algorithm 4.1). This greatly simplifies the expression of the elasticity tensor $\mathbf{C}$, and reduces the complexity of the computational vademecum.

This simplification is made with the sole purpose of obtaining a less complex expression of damage, with which to check that the method proposed in this chapter operates as expected. Therefore, although it is possible to implement a full description of the damage model by following the POD–PGD approach presented in the previous section, it has been decided to leave it for future developments of the method, and use only the simplified version. Obviously, the behaviour of the simplified model is different from that of the full model,

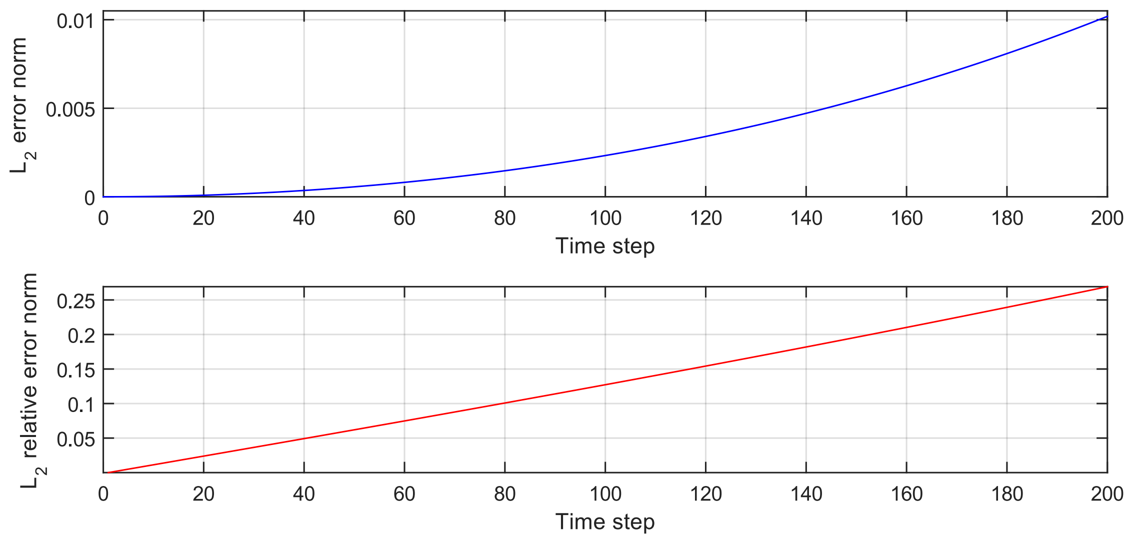


Figure 4.4. L_2 error norm (top) and L_2 relative error norm (bottom) of the nodal displacements between full and simplified damage models.

and may vary depending on the implemented application. Just to analyse how different the two approaches are, a comparison between the full model and the simplified model has been performed in a particular example.

In the comparison, a distributed uniform load is applied vertically to the upper surface of a unit cube, which is fixed at its base and allows displacements along the horizontal plane. The load is applied in increments of 1 N during 200 pseudo-time steps. The material is linear elastic, with a Young's modulus of $E = 5$ Pa, a Poisson's ratio of $\nu = 0.3$, and damage parameters of $\alpha = 0.4$, and $\beta = 0.1$, similar to the examples that will be described in § 4.6.1. The full model and the simplified model have been compared with each other, and L_2 error norms have been computed for the nodal displacements at each step of the simulation (Fig. 4.4). The results show that the differences between models in this example are appreciable, and increase as the simulation advances. It has been checked that, in the simplified model, the values of damage D grow faster than in the full model. However, because of its simplicity, the model in which $h(\Phi^m) = 0$ is a good starting point for developing the method.

4.5.4 Matrix formulation

The practical simplification described in the previous section, by which $h = 0$, is followed here. For clarity of exposition in this formulation, parameter $\boldsymbol{s}$ governing the position of the load is also omitted.

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. (4.14) as

$$\int_{\bar{\Gamma}}\int_{\mathcal{E}_q}\cdots\int_{\mathcal{E}_1}\int_{\Omega}\mathcal{U}\ \mathrm{d}\Omega\,\mathrm{d}\mathcal{E}_1\cdots\mathrm{d}\mathcal{E}_q\,\mathrm{d}\bar{\Gamma}=\int_{\bar{\Gamma}}\int_{\Gamma_{t_2}}\Delta\boldsymbol{u}^*\cdot\Delta\boldsymbol{f}\,\mathrm{d}\Gamma\,\mathrm{d}\bar{\Gamma}, \tag{4.16}$$

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

Assuming that the POD basis of the scalar field of damage $\boldsymbol{g}$ consists of q vectors, Eq. (4.13 left), the matrix form for the computation of a new mode of $\Delta\boldsymbol{u}$ in Eq. (4.15) can be expressed, following the notation already described in § 2.4 [157], as

$$\begin{aligned}&\Delta\boldsymbol{u}(\boldsymbol{x};\xi_1,\ldots,\xi_q)\\&=\sum_{i=1}^{n}\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{F}^i\cdot\prod_{K=1}^{q}\boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K)\ \boldsymbol{G}_K{}^i+\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R}\cdot\prod_{K=1}^{q}\boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K)\ \boldsymbol{S}_K,\end{aligned} \tag{4.17}$$

where $\boldsymbol{M}_K$ 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

$$\begin{aligned}\Delta\boldsymbol{u}^*(\boldsymbol{x};\xi_1,\ldots,\xi_q)&=\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R}^*\cdot\prod_{K=1}^{q}\boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K)\ \boldsymbol{S}_K\\&\quad+\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R}\cdot\boldsymbol{M}_1{}^{\mathrm{T}}(\xi_1)\ \boldsymbol{S}_1{}^*\cdot\ldots\cdot\boldsymbol{M}_q{}^{\mathrm{T}}(\xi_q)\ \boldsymbol{S}_q+\ldots\\&\quad+\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R}\cdot\boldsymbol{M}_1{}^{\mathrm{T}}(\xi_1)\ \boldsymbol{S}_1\cdot\ldots\cdot\boldsymbol{M}_q{}^{\mathrm{T}}(\xi_q)\ \boldsymbol{S}_q{}^*.\end{aligned} \tag{4.18}$$

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.

4.5.4.1 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 $\boldsymbol{S}_1{}^*=\boldsymbol{S}_2{}^*=\ldots=\boldsymbol{S}_q{}^*=\boldsymbol{0}$ and Eq. (4.18) can be expressed as

$$\Delta\boldsymbol{u}^*(\boldsymbol{x};\xi_1,\ldots,\xi_q)=\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R}^*\cdot\prod_{K=1}^{q}\boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K)\ \boldsymbol{S}_K. \tag{4.19}$$

By incorporating Eqs. (4.13 left), (4.17), and (4.19) into the left-hand side of Eq. (4.16), $\mathcal{U}$ results in

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

and the right-hand side of Eq. (4.16), after substituting Δu^* by Eq. (4.19), can be expressed as

$$\int_{\bar{\Gamma}} \int_{\Gamma_{t2}} \left(\boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{N}(\boldsymbol{x}) \cdot \prod_{K=1}^{q} \boldsymbol{S}_K{}^{\mathrm{T}}\ \boldsymbol{M}_K(\xi_K) \right) \cdot \Delta \boldsymbol{f}\ \ \mathrm{d}\Gamma\, \mathrm{d}\bar{\Gamma}.$$

Finally, after applying variable separation, rearranging the terms, and calculating the integrals, Eq. (4.16) can be written as

$$\begin{aligned}
&\sum_{i=1}^{n} \left(\boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{K}_1\ \boldsymbol{F}^i \right) \cdot \left(\boldsymbol{S}_1{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{\xi}_1}\ \boldsymbol{G}_1{}^i \right) \cdot \ldots \cdot \left(\boldsymbol{S}_q{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{G}_q}\ \boldsymbol{G}_q{}^i \right) + \ldots \\
&\qquad + \sum_{i=1}^{n} \left(\boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{K}_q\ \boldsymbol{F}^i \right) \cdot \left(\boldsymbol{S}_1{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{G}1}\ \boldsymbol{G}_1{}^i \right) \cdot \ldots \cdot \left(\boldsymbol{S}_q{}^{\mathrm{T}} \boldsymbol{M}_{\boldsymbol{\xi}_q}\ \boldsymbol{G}_q{}^i \right) \\
&\qquad + \left(\boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{K}_1\ \boldsymbol{R} \right) \cdot \left(\boldsymbol{S}_1{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{\xi}_1}\ \boldsymbol{S}_1 \right) \cdot \ldots \cdot \left(\boldsymbol{S}_q{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{G}_q}\ \boldsymbol{S}_q \right) + \ldots \\
&\qquad\quad + \left(\boldsymbol{R}^{*\mathrm{T}}\ \boldsymbol{K}_q\ \boldsymbol{R} \right) \cdot \left(\boldsymbol{S}_1{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{G}1}\ \boldsymbol{S}_1 \right) \cdot \ldots \cdot \left(\boldsymbol{S}_q{}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{\xi}_q}\ \boldsymbol{S}_q \right) \\
&\qquad\qquad\qquad = \boldsymbol{R}^{*\mathrm{T}}\ \Delta \boldsymbol{f} \cdot \boldsymbol{S}_1{}^{\mathrm{T}}\ \boldsymbol{m}_{\boldsymbol{G}1} \cdot \ldots \cdot \boldsymbol{S}_q{}^{\mathrm{T}}\ \boldsymbol{m}_{\boldsymbol{G}q},
\end{aligned} \tag{4.20}$$

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

$$\boldsymbol{K}_K = \int_{\Omega} \boldsymbol{v}_K\ \boldsymbol{B}^{\mathrm{T}}\ \mathbf{C}\boldsymbol{B}\, \mathrm{d}\Omega, \tag{4.21}$$

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. (4.13 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. (4.20), the matrix $\boldsymbol{M}_{\boldsymbol{G}L}$ does not exist when $L = K$. Instead, matrices $\boldsymbol{M}_{\boldsymbol{\xi}K}$ 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, $\boldsymbol{m}_{\boldsymbol{G}K}(\xi_K)$ 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. (4.20).

4.5.4.2 Computation of an arbitrary 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. (4.18) can be expressed as

$$\Delta \boldsymbol{u}^*(\boldsymbol{x}; \xi_1, \ldots, \xi_q) \\ = \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\ \boldsymbol{R} \cdot {\boldsymbol{M}_1}^{\mathrm{T}}(\xi_1)\ \boldsymbol{S}_1 \cdot \ldots \cdot {\boldsymbol{M}_a}^{\mathrm{T}}(\xi_a)\ {\boldsymbol{S}_a}^* \cdot \ldots \cdot {\boldsymbol{M}_q}^{\mathrm{T}}(\xi_q)\ \boldsymbol{S}_q. \quad (4.22)$$

By incorporating Eqs. (4.13 left), (4.17), and (4.22) into Eq. (4.16), 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) \cdot \ldots \cdot \left({\boldsymbol{S}_a}^{*\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{G}a}\ {\boldsymbol{G}_a}^i\right) \cdot \ldots \\ \cdot \left({\boldsymbol{S}_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) \cdot \ldots \\ \cdot \left({\boldsymbol{S}_a}^{*\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{G}a}\ {\boldsymbol{G}_a}^i\right) \cdot \ldots \cdot \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) \cdot \ldots \cdot \left({\boldsymbol{S}_a}^{*\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{G}a}\ \boldsymbol{S}_a\right) \cdot \ldots \cdot \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) \cdot \ldots \cdot \left({\boldsymbol{S}_a}^{*\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{G}a}\ \boldsymbol{S}_a\right) \cdot \ldots \\ \cdot \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}_K}^{\mathrm{T}}\ \boldsymbol{M}_{\boldsymbol{S}},$$

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

4.6 Numerical examples

To illustrate better how the proposed method behaves, some examples have been studied. In particular, § 4.6.1 analyses some basic problems using a unit cube to test the validity of the approach, and § 4.6.2 shows how the method can be applied to simulate the surgical removal of adipose tissue from the gallbladder. All the examples have been implemented using the POD-PGD approach and the simplified model of damage described in §§ 4.5.2 and 4.5.3.

4.6.1 Unit cube

A series of simple test examples have been developed to show the applicability of the method. The model common to these simple test examples is a unit cube

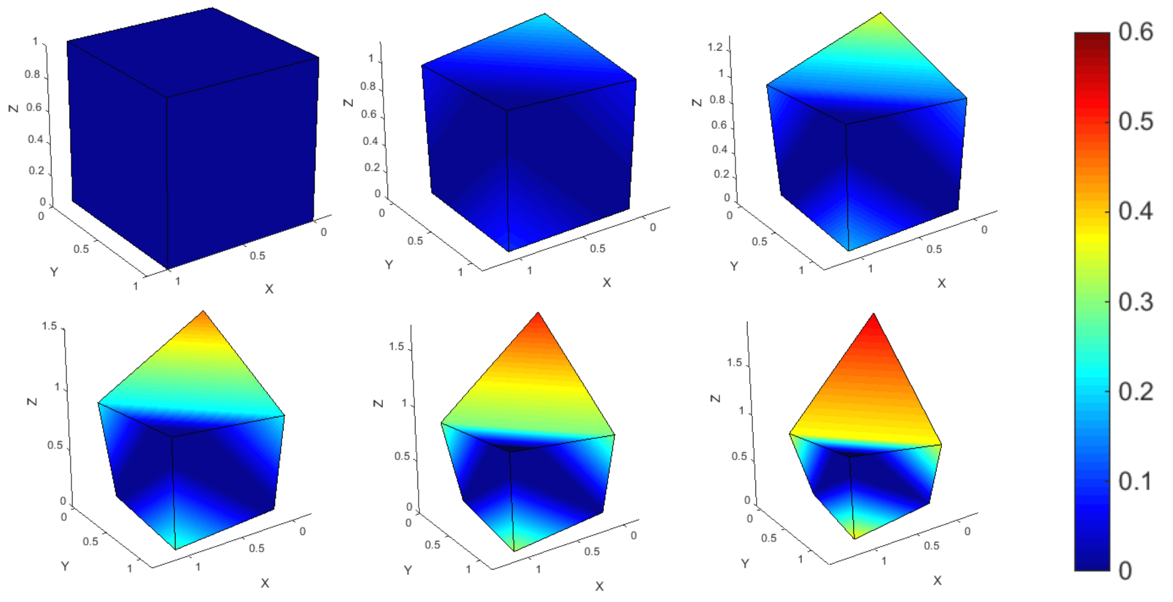


Figure 4.5. Damage evolution of a unit cube loaded at a node. From left to right and from top to bottom, initial configuration, and pseudo-time steps 2, 4, 6, 8, and 10 are depicted. Colours show the values of the interpolated variable of damage *D*.

consisting of a single regular eight-node hexahedral element. The cube is fixed at its base, and different loads are applied on its upper part. § 4.6.1.1 describes the damage evolution of a cube to which one of its upper corners is pulled upwards, while § 4.6.1.2 describes the damage evolution a cube undergoing an ideal pure shear deformation.

4.6.1.1 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 plane Z. 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 $E = 5$ 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 field

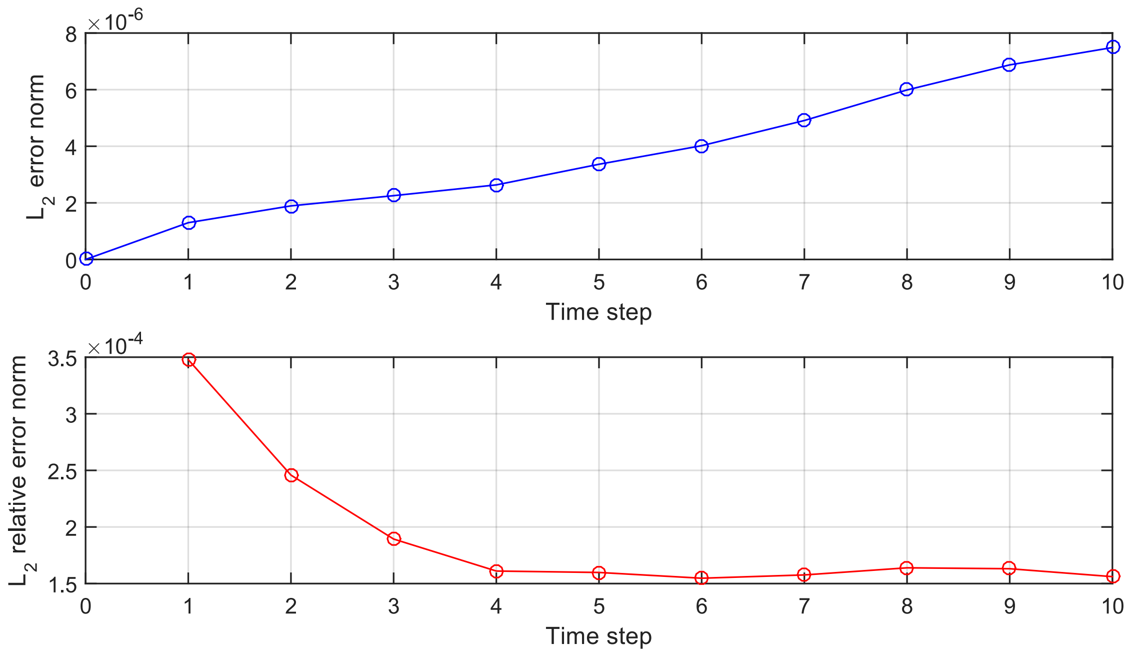


Figure 4.6. L_2 error norm (top) and L_2 relative error norm (bottom) of the unit cube loaded at a node for each of the 10 computed pseudo-time steps.

of damage $\boldsymbol{g}$ is recorded at the Gauss integration points of the element. 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 has an associated value of explained variance, the selected vectors to form the reduced basis $\boldsymbol{v}$ are those that accounts 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, 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 discretisation of coefficients ξ_K (with $K = 1, 2, 3$), represented by matrices $\boldsymbol{G}_K$ in § 4.5.4. 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 discretisation of coefficients

ξ_K, PGD required 28 modes before reaching convergence.

The computed vademecum can be incorporated into the explicit incremental POD–PGD algorithm (Algorithm 4.2). Again, ten pseudo-time steps are computed in real-time, six of which are shown in Fig. 4.5. In analysing the sequence, it can be observed that damage concentrates around the corner that 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. 4.6), which allow to compare the differences between the results obtained by the POD–PGD algorithm and the reference solution.

4.6.1.2 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. 4.7. 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.

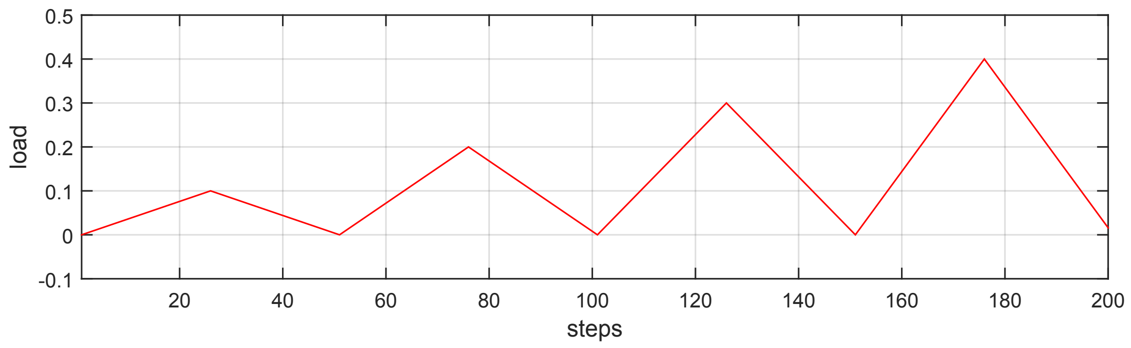


Figure 4.7. Total load $\boldsymbol{f}$ applied at each simulation step to a unit cube undergoing pure shear deformation.

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.

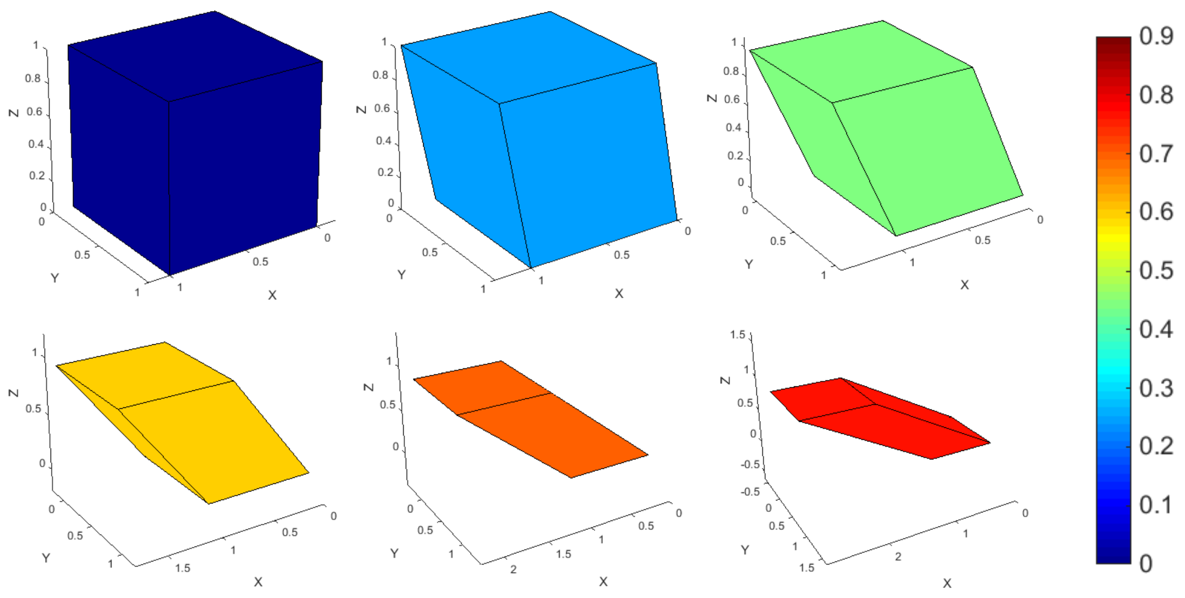


Figure 4.8. Damage evolution of a unit cube undergoing pure shear deformation. Initial configuration and steps 20, 70, 120, 170, and 175 are shown (from left to right, and from top to bottom), which correspond to values of total applied force f of 0, 0.08, 0.16, 0.24, 0.32, and 0.4 N, respectively.

The chosen POD reduced basis consists of 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 4.2 allows the real-time computation of the 200 pseudo-time step, six of which are shown in Fig. 4.8. 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. 4.9. Lastly, L_2 error norms have been computed for each step (see Fig. 4.10), which allow to determine how the results obtained by the POD–PGD algorithm differ from the reference solution.

4.6.2 Removal of fatty tissue from gallbladder

In laparoscopic surgery, procedures based on tearing and ripping soft tissues are much more common than those involving cutting. Thus, to achieve realistic laparoscopic simulations, it is of utmost importance to be able to reproduce all the tearing processes that can take place. For instance, in laparoscopic chole-

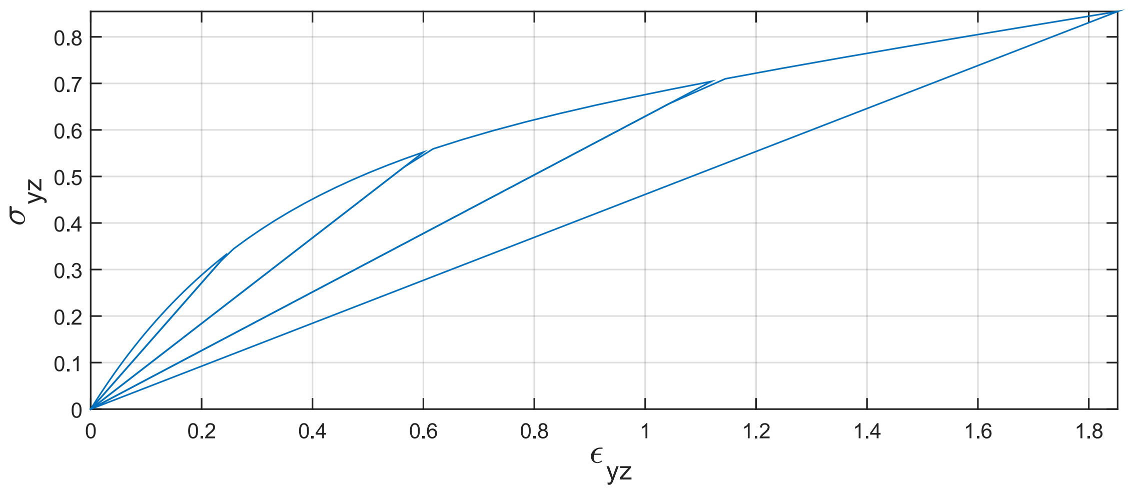


Figure 4.9. Stress–strain curve.

cystectomy (i. e. the removal of the gallbladder), prior to performing the single cut that separates the gallbladder from the cystic duct, the anatomical region known as Calot's (or hepatobiliary) triangle must be dissected to unveil the cystic duct and the cystic artery. During this process, the adipose tissue needs to be removed to obtain a view of the underlying structures [181]. This removal is made simply by tearing the adipose tissue with the help of laparoscopic instruments equipped with either scraping tools or hook-shaped electrocautery tips. 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.

A model of the Calot's triangle region is considered that includes the common hepatic duct, the cystic duct, and the right hepatic duct, along with the gallbladder, see Fig.4.11. 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. 4.12. 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 contact of these areas with the liver and other surrounding connective tissues.

The constitutive material is linear elastic, although the consideration of hyperelastic models do not induce any additional difficulty and have been previously studied in [89, 90]). Unlike the previous examples, two different material properties are considered, one for the gallbladder and another for the fat. Experimental studies [182–184] suggest that the material properties of the gallbladder can be selected with values $E_G = 1.15$ kPa and $\nu_G = 0.48$, whereas the fat can

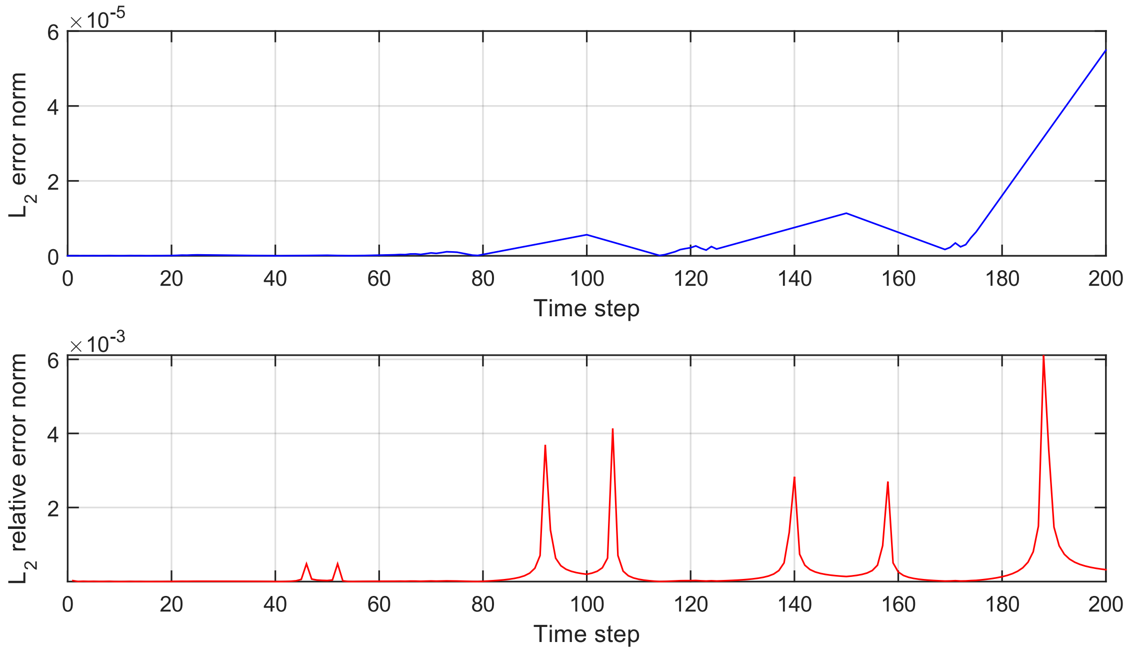


Figure 4.10. L_2 error norm (top) and L_2 relative error norm (bottom) of the unit cube undergoing pure shear deformation for each of the 200 computed pseudo-time steps.

be characterised with $E_F = 0.5$ MPa and $\nu_F = 0.495$. The fact that the model is constituted with two different materials implies to modify the computation of the stiffness matrix $\boldsymbol{K}_K$ in Eq. (4.21), which has to be obtained now as a sum of its parts,

$$\boldsymbol{K}_K = \int_{\Omega_G} \boldsymbol{B}^{\mathrm{T}}\, \mathsf{C}_G \boldsymbol{B}\, \mathrm{d}\Omega_G, + \int_{\Omega_F} \boldsymbol{\upsilon}_K\, \boldsymbol{B}^{\mathrm{T}}\, \mathsf{C}_F \boldsymbol{B}\, \mathrm{d}\Omega_F,$$

where Ω_G represents the domain of the gallbladder, Ω_F represents the domain of the fat, and C_G and C_F are the elasticity tensors of the gallbladder and the fat, respectively. Damage, then, is only considered in the fat but not in the gallbladder. Considered values of damage parameters are $\alpha = 1$, and $\beta = 0$.

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 in this early stage, traction is applied only on one node; later developments of the method will consider a broader set of nodes on the surface. 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 the 100% of the explained variance. Coefficients ξ_i are discretised with the following interval limits, and number of uniformly distributed mesh nodes: $\xi_1 \in [-152.1, -145.2]$ using 6 900 nodes; $\xi_2 \in [-6.3, 5.8]$ using 4 840 nodes; and $\xi_3 \in$

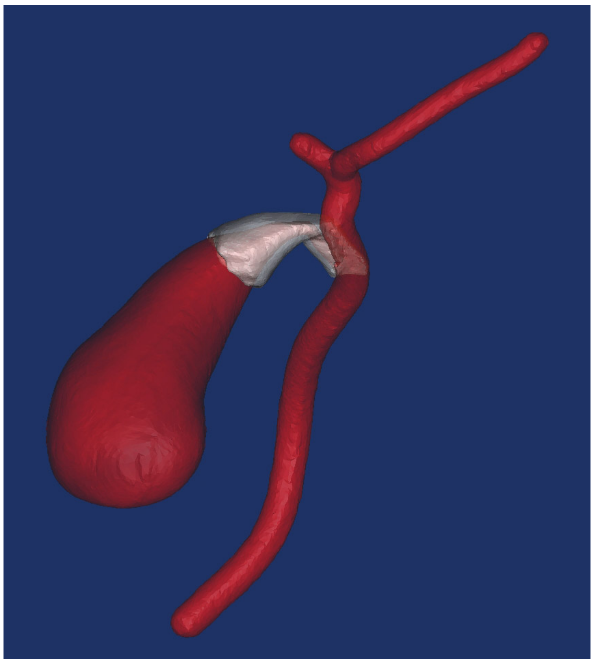


Figure 4.11. Model of the Calot's triangle region.

$[-0.3, 0.4]$ 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. The sequence, depicted in Fig. 4.13, simulates the interactive tearing of the adipose tissue by means of a laparoscopic instrument (not pictured), which grasps and pulls the 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.

For visualisation purposes, elements whose Gauss points have reached a prescribed damage threshold, say 0.8, are removed from the rendering process. Indeed, grasping is considered lost and therefore elastic recovery is perceived from the perspective of the user, see Fig. 4.14. To increase realism, the damaged surface is rendered using artificial roughness textures, to indicate the user that the tissue is broken.

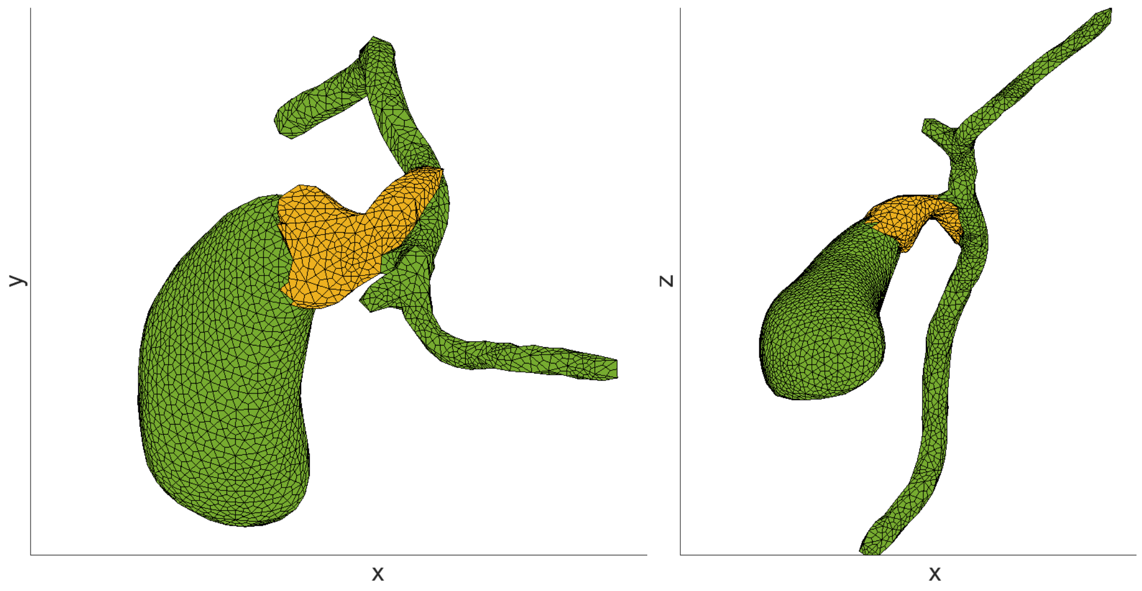


Figure 4.12. Geometry of the finite element model of the gallbladder (in green) and its adipose tissue (in yellow).

4.6.3 Timings

The results presented in this section have been obtained using a 64-bit laptop, running Matlab 2014b with an i5 processor running at 2.50 GHz, and 4 Gb RAM memory. In spite of the use of non-optimised Matlab code prototypes, computation times in the example of the gallbladder were always under 1 ms, which is enough for visual as well as haptic real-time requirements.

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

4.6.4 Conclusions

In this chapter, 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, an explicit incremental finite element algorithm is introduced. However, it requires the computation of the inverse of a large stiffness matrix at each step, which makes its implementation incompatible with the real-time execution.

To address this problem, the use of PGD computational vademecums is proposed, since the allow to obtain quick direct solutions with a low computational

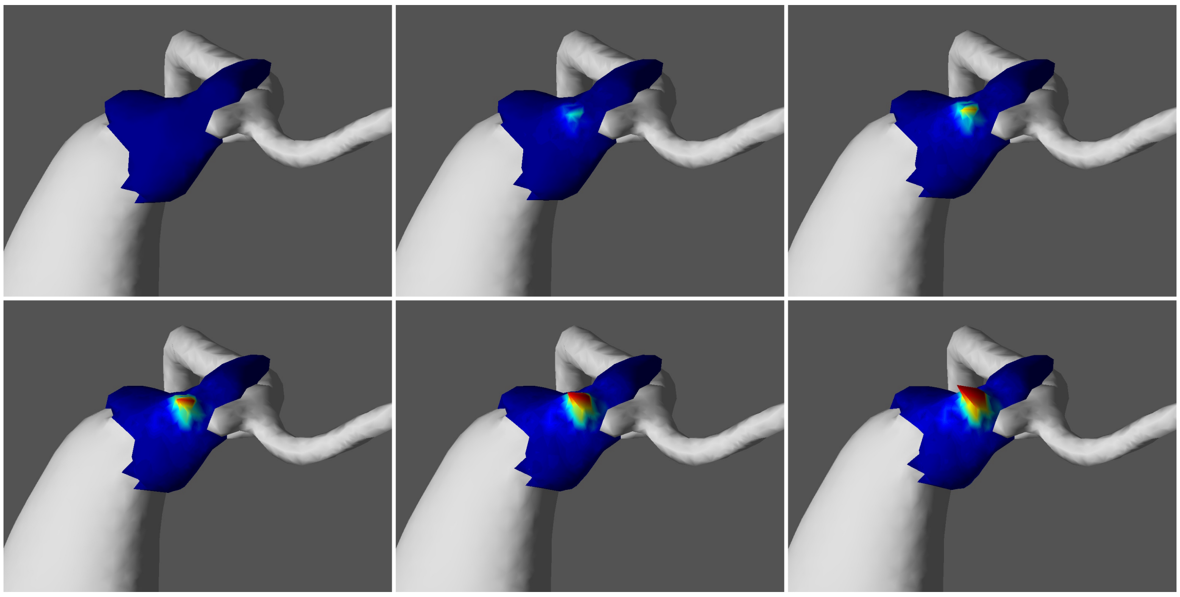


Figure 4.13. Damage sequence of the removal of fatty tissue from gallbladder.

cost. Nevertheless, the parametrisation of the damage field for this kind of problems may be huge, and a reduced order model technique is required to minimise 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 simple and complex examples with excellent results. A surgical application focused on the simulation of the tearing of adipose tissue is implemented. 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. The method achieves a high degree of realism in the simulation of tearing of soft tissues. In sharp contrast to existing approaches, it does not require any update in the model mesh, and no element is modified or removed.

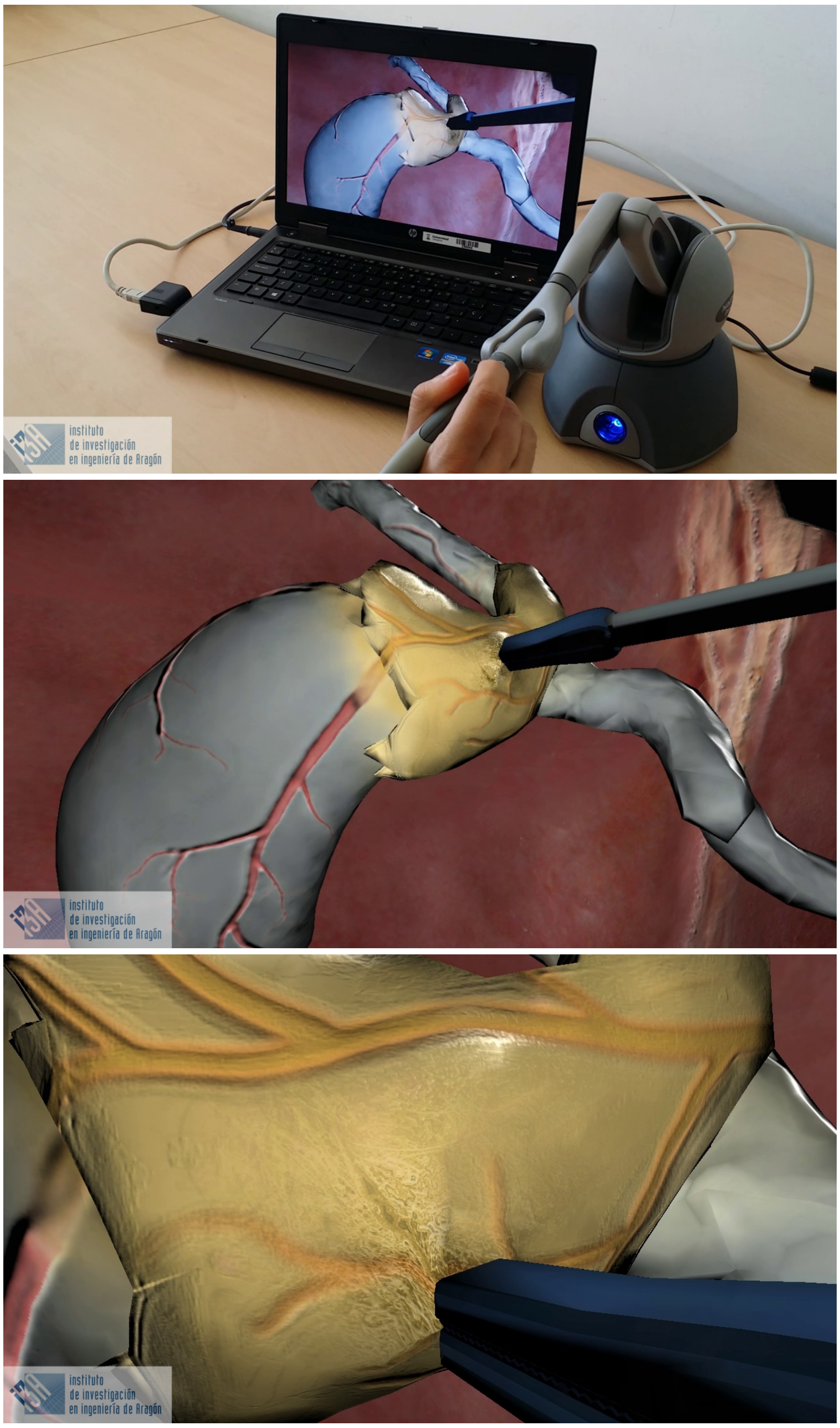


Figure 4.14. Images of the implemented simulator. Top: full view of the simulator, consisting of a laptop and a haptic interface that controls the virtual laparoscopic instrument. Middle: screenshot showing the process of tearing of the adipose tissue. Bottom: close-up of the damage caused to the tissue.

Chapter 5

Conclusions

5.1 Concluding remarks

In this thesis, two novel strategies have been developed for the real-time simulation of two interactions that are common in the operating theatre: surgical cutting, and tearing of soft tissues. These strategies are based on the use of model reduction techniques, mainly the Proper Generalised Decomposition (PGD), which has been extensively used in its form of computational vademecum, i. e. a general solution of a parametric high-dimensional problem that can be evaluated at very fast feedback rates.

Simulators have proved to be very beneficial for the training of surgical skills and their transfer to the clinical setting. Classic surgical simulators, such as live animals or human cadavers, provide a realistic working environment. However, they present a series of disadvantages (e. g. high costs, complex maintenance, access to special facilities, or cultural issues, to mention a few) that make computer-based surgical simulators an appealing alternative that have attracted great interest during the recent years.

Yet, computer-based surgical simulation is considered to be in an initial phase. During the early years of its development, simulators failed to meet a minimal level of realism because of both the limited computing power and the lack of algorithms based on physics. By contrast, over the last decade, several numerical methods based on computational continuum mechanics have allowed the real-time simulation of deformations using simple constitutive materials for the organs. Nevertheless, other surgical invasive interactions, such as the cutting and tearing of soft tissues, had not been adequately addressed.

The computational vademecums provided by the PGD have shown, on their own, to be a very efficient tool for simulating the deformations of linear and non-linear constitutive materials, but have not been used so far to simulate other surgical interactions. In this thesis, PGD appears in combination with other numerical methods and model order reduction techniques to achieve the real-time simulation of cut and tearing of soft tissues.

First, an approach for the simulation of surgical cutting of soft tissues is studied. It is valid for haptic environments, which require 500–1000 Hz of feedback response, and combines PGD computational vademecums and XFEM techniques in a continuous–discontinuous multi-scale framework. The realistic real-time simulation of surgical cuts is a major source of difficulties, since it demands the modification of the geometry and the topology of both the domain and its associated mesh without penalising the computation time.

The key idea to achieve this objective consists in pre-computing, at an off-line stage, the vademecum of an organ for every possible location of an applied load (i. e. the continuous solution). In addition, all the displacements generated by cutting each node of the accessible surface of the model, for any possible orientation of those cuts, are also computed off-line using XFEM techniques (i. e. the discontinuous solutions). Then, at the interactive on-line stage, the continuous solution is used as a reduced basis, which is enriched by the required discontinuous solutions, to obtain the solution of the cut model. From a practical point of view, the on-line stage only involves the computation of a simple matrix system of equations that can be solved very fast. The manner in which the discontinuous solutions have to be combined together to simulate long cuts (those that implies two or more nodes), is provided by the so-called cracking node method.

An application for the simulation of corneal surgery (radial keratotomy) is developed and analysed. The cornea is modelled as linear elastic and has been subjected to different patterns of radial incisions. Two different ways of approaching the solution (the PRB and the SRB formulations) are provided, both depending on whether the computational vademecum is particularised for a load position or not before its projection. Reported errors were low, and computation times were within the specified limits for haptic devices.

Second, the real-time simulation of tearing of soft tissues is also developed. This is achieved by computing the PGD vademecum of a parametric equation based on the theory of continuum damage mechanics. In order to simplify the computation of the damage fields, and to reduce the number of parameters, POD model order reduction techniques are incorporated into the vademecum. The fundamental aspect of this approach consists in implementing the vademecum within the framework of an incremental explicit formulation, as a sort of time integrator, i. e. a feedback system that routes back the last computed outputs as initial conditions with which to compute the new inputs.

This method is applied for simulating, in real-time, the removal of visceral adipose tissue from the walls of the gallbladder. Both fat and gallbladder are modelled as linear elastic, although considering different material parameters for each of them. Excellent results are obtained meeting all expectations: haptic real-time execution is achieved, low levels of error are generated, and a realistic immersive feeling is perceived.

In short, the use of strategies based on PGD computational vademecums, in combination with other supporting techniques, has proven to be an appealing methodology to address the real-time simulation of surgical interactions, and paves the way towards the development of future generations of simulators, whose complexity will be focused on physiological, micro-anatomical and biochemical aspects.

5.2 Thesis contributions

The work that has been carried out in the context of this thesis has provided two main contributions in the fields of computational biomechanics and numerical simulation.

On the one hand, it has been proven that PGD computational vademecums can be efficiently used as reduced order bases *à la* POD. The combination of these reduced bases with XFEM techniques allows the real-time simulation of the process of cutting. On the other hand, it has also been proven that computational vademecums can be conveniently coupled with POD techniques to reduce multi-parametric systems of great dimensions, such as those obtained by the equations of continuum damage mechanics. These vademecums can be implemented, in the context of an explicit incremental formulation, to simulate the tearing of soft tissues in real-time.

However, other research works (not necessarily related to the real-time simulation of surgery) have been carried out during the doctoral period of the author to prove the suitability of PGD in certain fields of applied mathematics and engineering. These works encompass the solution of problems with high-order differential operators [185], the development of applications for augmented learning [186], and the detection of contact between deformable solids [187].

5.3 Future work

The design of full second-generation computer-based surgical simulators is a task that can be as unbounded as the living organs they simulate. This thesis has shown that the simulation of the most common invasive surgical interactions can be achieved using PGD. However, there are matters that remain open for further research.

For instance, with regard to cutting, the extreme cases in which cutting leads to dissection have not been studied. Such a study did not need to be carried out in the example of the corneal surgery, since the developed simulator is intended for training surgeons in the particular intervention of radial keratotomy, where no dissection is possible. Nevertheless, these kind of studies may be interesting to make the method exportable to other organs or surgical procedures.

Concerning tearing, the implemented method has been developed using a simplified model of damage, which has been very useful to verify its correct operation. Now that this verification has been positively conducted, a complete implementation of the damage model (which includes the damage field $\boldsymbol{h}$ into the parametrisation of the computational vademecum) should be obtained. Aside from this, a more detailed treatment of the resection process would also be interesting to analyse.

In general, future full developments of these simulators may demand the implementation of several interactions simultaneously; for example, cutting, tearing, and suturing tissues in the same session, and not as separate actions as they are at present. It may be convenient to devise some kind of strategy to handle these demands.

As a final remark, although the evolution of computer-based surgical simulators was established by Satava, twenty years ago, as a sequence of five generations (see § 1.1.3.2), this prediction may not be fulfilled if clinical practice demands other needs. For example, it seems much more relevant that surgeons train and plan surgical procedures using real *patient avatars* (i. e. computational patient-specific models of organs) [25], rather than using extremely detailed but generic fifth-generation models.

5.4 Publications

As a result of the research conducted throughout the doctoral period, several articles have been published in journals, and in proceedings of national and international conferences, as well as posters and book chapters. These publications are detailed below.

5.4.1 Journal publications

1. Quesada, C., González, D., Alfaro, I., Cueto, E., & Chinesta, F. (2016). Computational vademecums for real-time simulation of surgical cutting in haptic environments. *International Journal for Numerical Methods in Engineering.*

2. Quesada, C., González, D., Alfaro, I., Cueto, E., Huerta, A., & Chinesta, F. (2015). Real-time simulation techniques for augmented learning in science and engineering. *The Visual Computer*, 1–15.

3. Quesada, C., Xu, G., González, D., Alfaro, I., Leygue, A., Visonneau, M., ... & Chinesta, F. (2015). Un método de descomposición propia generalizada para operadores diferenciales de alto orden. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 31(3), 188–197.

4. González, D., Alfaro, I., Quesada, C., Cueto, E., & Chinesta, F. (2015). Computational vademecums for the real-time simulation of haptic collision between nonlinear solids. Computer Methods in Applied Mechanics and Engineering, 283, 210-223.

A fifth article about the tearing of soft tissues described in chapter 4 (Quesada, C., Alfaro, I., González, D., & Chinesta, F., Cueto, E. Haptic simulation of tissue tearing during surgery) is pending publication.

5.4.2 Conference papers

5. Carlos Quesada, Iciar Alfaro, David Gonzalez, Elias Cueto, & Francisco Chinesta (2016). Model order reduction of initial value problems. In *12th World Congresses on Computational Mechanics (WCCM XII).* Seoul, South Korea

6. Carlos Quesada, David González, Icíar Alfaro, Elías Cueto, & Francisco Chinesta (2015). Real-time simulation of surgical cutting in haptic environments using computational vademecums. In *Congress on Numerical Methods in Engineering (CMN 2015).* Lisbon, Portugal.

7. Carlos Quesada, David González, Icíar Alfaro, Elías Cueto, & Francisco Chinesta (2015). Real-time simulation of surgical cutting in haptic environments using computational vademecums. In *2nd Joint Thematic Workshop CSMA-SEMNI*. Biarritz, France.

8. Carlos Quesada, Icíar Alfaro, David González, Elías Cueto, & Francisco Chinesta (2015). Real-time simulation of surgical cutting in haptic environments using computational vademecums. In *IV Jornada de Jóvenes Investigadores I3A*. Zaragoza, Spain.

9. C. Quesada, D. González, I. Alfaro, E. Cueto, & F. Chinesta (2014). Simulación del corte quirúrgico en tiempo real mediante PGD. In *XXXII Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB 2014)*. Barcelona, Spain.

10. C. Quesada, D. González, I. Alfaro, E. Cueto, & F. Chinesta (2014). Real-time simulation of surgical cutting using PGD. In *11th World Congress on Computational Mechanics (WCCM XI)*. Barcelona, Spain

11. C. Quesada, I. Alfaro, D. González, & E. Cueto (2013). A new generation of real-time simulation techniques. In *II Jornada de Jóvenes Investigadores I3A*. Zaragoza, Spain.

5.4.3 Poster communications

12. C. Quesada, D. González, I. Alfaro, & E. Cueto (2013). PGD approximations for high-order problems. In *2nd International Workshop on Reduced Basis, POD and PGD Model Reduction Techniques*. Blois, France.

5.4.4 Book chapters

13. Gonzalez, D., Alfaro, I., Quesada, C., Cueto, E., & Chinesta, F. (2015). Vademecums for Real-Time Computational Surgery. In Computational Biomechanics for Medicine (pp. 3-12). Springer International Publishing.

14. Quesada, C., Alfaro, I., González, D., Cueto, E., & Chinesta, F. (2014, October). PGD-Based Model Reduction for Surgery Simulation: Solid Dynamics and Contact Detection. In *International Symposium on Biomedical Simulation* (pp. 193–202). Springer International Publishing.

Appendices

Appendix A

Conclusiones

A.1 Observaciones finales

En esta tesis, se han desarrollado dos estrategias novedosas para la simulación en tiempo real de dos interacciones muy comunes en la sala de operaciones, como son el corte y el rasgado (o desgarro) de tejidos blandos. Estas estrategias se basan en el uso de técnicas de reducción de modelos, principalmente la Descomposición Propia Generalizada (PGD), que se ha utilizado ampliamente en su forma de vademécum computacional, esto es, una solución general a un problema paramétrico de altas dimensiones que se puede evaluar a gran velocidad.

Los simuladores han demostrado ser muy beneficiosos tanto para el aprendizaje y desarrollo de habilidades quirúrgicas como para su transferencia a la práctica clínica. Los simuladores quirúrgicos tradicionales, como animales vivos o cadáveres humanos, proporcionan un ambiente de trabajo realista. Sin embargo, las desventajas que presentan (altos costes, mantenimiento complejo, acceso a instalaciones especiales, cuestiones ético-culturales, etc.) convierten a los simuladores quirúrgicos por ordenador en una alternativa atractiva, la cual ha suscitado un gran interés en los últimos años.

Sin embargo, la simulación quirúrgica por ordenador todavía se encuentra en una fase inicial. Durante los primeros años de su desarrollo, los simuladores no cumplían con el nivel mínimo de realismo por dos razones fundamentales: la potencia de cálculo era muy limitada y faltaban por desarrollar algoritmos con cierta base física. Por el contrario, durante la última década, el desarrollo de métodos numéricos basados en mecánica computacional ha permitido la simulación en tiempo real de deformaciones utilizando leyes constitutivas simples para modelar los órganos. Asimismo, otras interacciones quirúrgicas invasivas, tales como el corte y el rasgado de tejidos blandos, no se habían tratado adecuadamente.

Los vademécums computacionales proporcionados por la PGD han demostrado ser, por sí mismos, una herramienta muy eficiente para la simulación de deformaciones de materiales constitutivos lineales y no lineales, pero no se habían utilizado hasta ahora para simular otras interacciones quirúrgicas. En esta tesis, PGD aparece en combinación con otros métodos numéricos y técnicas de reducción de modelos para conseguir la simulación en tiempo real del corte y el rasgado de tejidos blandos.

En primer lugar, se estudia un método para la simulación del corte quirúrgico de tejidos blandos. Es válido para entornos hápticos, los cuales requieren tiempos de respuesta de 500–1 000 Hz, y combina vademécums computacionales obtenidos con PGD con técnicas XFEM en un marco de trabajo multiescala. La simulación realista en tiempo real del corte quirúrgico es una fuente importante de dificultades, ya que exige modificar la geometría y la topología del dominio y de su malla asociada sin penalizar el tiempo de cálculo.

La idea clave para lograr este objetivo consiste en precalcular, en una etapa *off-line*, el vademécum de un órgano para cada posible localización de una carga aplicada (obteniéndose así una solución continua). Además, todos los desplazamientos generados por el corte de cada nodo de la superficie accesible del modelo, para cualquier posible orientación de los cortes, también se calculan *off-line* utilizando técnicas XFEM (obteniéndose así un conjunto de soluciones discontinuas). A continuación, en la etapa interactiva *on-line*, la solución continua se utiliza como una base reducida, que se enriquece con las soluciones discontinuas necesarias, para obtener la solución del modelo cortado. Desde un punto de vista práctico, la fase *on-line* sólo implica el cálculo de un sistema de ecuaciones matricial sencillo que se puede resolver muy rápido. La manera en que las soluciones discontinuas se combinan para simular cortes largos (aquellos que implican dos o más nodos), viene dada por el denominado *cracking node method*.

Se ha desarrollado y analizado una aplicación para la simulación de cirugía corneal (en particular, la intervención conocida como queratotomía radial). La córnea se ha modelado como un material elástico lineal y se ha sometido a diferentes patrones de incisiones radiales. Se proporcionan dos enfoques diferentes para abordar la solución (las formulaciones PRB y SRB), en función de si el vademécum computacional se particulariza para una posición de carga antes de su proyección o no. Los errores obtenidos han sido bajos y los tiempos de cálculo se encontraban dentro de los límites especificados para dispositivos hápticos.

En segundo lugar, también se ha desarrollado la simulación en tiempo real del rasgado de tejidos blandos. Esto se consigue mediante el cálculo de un vademécum obtenido con PGD para una ecuación paramétrica basada en la teoría de mecánica de daño continuo. Para simplificar el cálculo de los campos de daño, y con el fin de reducir el número de parámetros, se han incorporado al vademécum técnicas POD de reducción de modelos. El aspecto fundamental de este enfoque consiste en la aplicación del vademécum en el marco de una formulación explícita incremental como si de una especie de integrador temporal se tratase. Es decir, como un sistema realimentado en el que las salidas son las condiciones iniciales con las que se calculan las nuevas entradas.

Este método se aplica para simular, en tiempo real, la extirpación de tejido adiposo visceral de las paredes de la vesícula biliar. Tanto la grasa como la vesícula biliar se han modelado como un material elástico lineal, aunque teniendo en cuenta diferentes parámetros de los materiales para cada uno de ellos. Se han obtenido excelentes resultados que cumplen con todas las expectativas: se ha conseguido la ejecución háptica en tiempo real, se han generado bajos niveles de error y las sensaciones percibidas son realistas.

En resumen, el uso de estrategias basadas en vademécums computacionales basados en PGD, en combinación con otras técnicas de apoyo, ha demostrado ser un método atractivo para hacer frente a la simulación en tiempo real de interacciones quirúrgicas, y allana el camino hacia el desarrollo de futuras generaciones de simuladores, cuya complejidad se centrará en aspectos fisiológicos, microanatómicos y bioquímicos.

A.2 Contribuciones de la tesis

El trabajo que se ha llevado a cabo en el contexto de esta tesis ha proporcionado dos contribuciones principales en los campos de la biomecánica computacional y la simulación numérica.

Por un lado, se ha demostrado que los vademecums computacionales basados en PGD se pueden utilizar eficientemente como bases de orden reducido «a la POD». La combinación de estas bases reducidas con técnicas XFEM permite la simulación en tiempo real del proceso de corte. Por otra parte, también se ha demostrado que los vademécums computacionales se pueden acoplar convenientemente con técnicas POD para reducir sistemas multiparamétricos de grandes dimensiones, como los obtenidos por las ecuaciones de la mecánica de daño continuo. Estos vademécums se pueden implementar, en el contexto de una formulación incremental explícita, para simular el rasgado de tejidos blandos en tiempo real.

Sin embargo, también se han llevado a cabo, durante el período de doctorado del autor, otros trabajos de investigación (no necesariamente relacionados con la simulación quirúrgica en tiempo real) para probar la idoneidad de PGD en ciertos campos de matemática aplicada e ingeniería. Estos trabajos abarcan la solución de problemas con operadores diferenciales de alto orden [185], el desarrollo de aplicaciones para el aprendizaje aumentado [186], y la detección de contacto entre sólidos deformables [187].

A.3 Trabajo futuro

El diseño de simuladores quirúrgicos completos de segunda generación es una tarea muy ambiciosa que puede ser tan ilimitada como los órganos vivos que se simulan. Esta tesis ha demostrado que PGD permite conseguir la simulación de las interacciones quirúrgicas invasivas más comunes. Sin embargo, todavía queda trabajo abierto para futuras investigaciones.

Por ejemplo, con respecto al corte, los casos extremos en los que éste lleva a la disección completa del tejido no se han estudiado. Tal estudio no ha sido necesario en el ejemplo de la cirugía corneal, ya que el simulador desarrollado está destinado a entrenar a cirujanos en la intervención particular de la queratotomía radial, donde la disección no es posible. Sin embargo, este tipo de estudios puede ser interesante para que el método sea exportable a otros órganos o procedimientos quirúrgicos.

En cuanto al rasgado, el método implementado se ha desarrollado utilizando un modelo simplificado de daño, que ha sido muy útil para verificar su correcto funcionamiento. Una vez que la verificación es positiva, se debe obtener una implementación completa del modelo de daño que incluya el campo de daño $\boldsymbol{h}$ en la parametrización del vademécum computacional. Además, también sería interesante analizar con más detalle el proceso de resección.

En general, futuros desarrollos completos de este tipo de simuladores pueden requerir la implementación de varias interacciones al mismo tiempo; por ejemplo, cortar, rasgar y suturar tejidos en una misma sesión, y no como acciones separadas, que es como se hace en la actualidad. Podría ser conveniente idear algún tipo de estrategia para gestionar estos casos.

Como última observación, aunque la evolución de los simuladores quirúrgicos por ordenador fue establecida por Satava hace veinte años como una secuencia de cinco generaciones (véase § 1.1.3.2), esta predicción puede no llegar a cumplirse si la práctica clínica tiene otras necesidades. Por ejemplo, parece mucho más relevante que los cirujanos entrenen y planifiquen los procedimientos quirúrgicos usando *avatares* reales del paciente (esto es, modelos de órganos por ordenador específicos para cada paciente) [25], en lugar de utilizar un modelo de quinta generación extremadamente detallado pero genérico.

A.4 Publicaciones

Como resultado de la investigación realizada a lo largo del período doctoral, se han publicado varios artículos y capítulos de libros, y se han realizado varias contribuciones en congresos nacionales e internacionales, tanto orales como en formato póster. Todas ellas se detallan a continuación.

A.4.1 Revistas

1. Quesada, C., González, D., Alfaro, I., Cueto, E., & Chinesta, F. (2016). Computational vademecums for real?time simulation of surgical cutting in haptic environments. *International Journal for Numerical Methods in Engineering*.
2. Quesada, C., González, D., Alfaro, I., Cueto, E., Huerta, A., & Chinesta, F. (2015). Real-time simulation techniques for augmented learning in science and engineering. *The Visual Computer*, 1–15.
3. Quesada, C., Xu, G., González, D., Alfaro, I., Leygue, A., Visonneau, M., ... & Chinesta, F. (2015). Un método de descomposición propia generalizada para operadores diferenciales de alto orden. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 31(3), 188–197.
4. González, D., Alfaro, I., Quesada, C., Cueto, E., & Chinesta, F. (2015). Computational vademecums for the real-time simulation of haptic collision between nonlinear solids. Computer Methods in Applied Mechanics and Engineering, 283, 210-223.

Está pendiente de publicación un quinto artículo sobre el rasgado de tejidos blandos descrito en el capítulo 4 de esta tesis (Quesada, C., Alfaro, I., González, D., & Chinesta, F., Cueto, E. Haptic simulation of tissue tearing during surgery).

A.4.2 Congresos

5. Carlos Quesada, Iciar Alfaro, David Gonzalez, Elias Cueto, & Francisco Chinesta (2016). Model order reduction of initial value problems. In *12th*

World Congresses on Computational Mechanics (WCCM XII). Seoul, South Korea

6. Carlos Quesada, David González, Icíar Alfaro, Elías Cueto, & Francisco Chinesta (2015). Real-time simulation of surgical cutting in haptic environments using computational vademecums. In *Congress on Numerical Methods in Engineering (CMN 2015)*. Lisbon, Portugal.

7. Carlos Quesada, David González, Icíar Alfaro, Elías Cueto, & Francisco Chinesta (2015). Real-time simulation of surgical cutting in haptic environments using computational vademecums. In *2nd Joint Thematic Workshop CSMA-SEMNI*. Biarritz, France.

8. Carlos Quesada, Icíar Alfaro, David González, Elías Cueto, & Francisco Chinesta (2015). Real-time simulation of surgical cutting in haptic environments using computational vademecums. In *IV Jornada de Jóvenes Investigadores I3A*. Zaragoza, Spain.

9. C. Quesada, D. González, I. Alfaro, E. Cueto, & F. Chinesta (2014). Simulación del corte quirúrgico en tiempo real mediante PGD. In *XXXII Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB 2014)*. Barcelona, Spain.

10. C. Quesada, D. González, I. Alfaro, E. Cueto, & F. Chinesta (2014). Real-time simulation of surgical cutting using PGD. In *11th World Congress on Computational Mechanics (WCCM XI)*. Barcelona, Spain

11. C. Quesada, I. Alfaro, D. González, & E. Cueto (2013). A new generation of real-time simulation techniques. In *II Jornada de Jóvenes Investigadores I3A*. Zaragoza, Spain.

A.4.3 Comunicaciones póster

12. C. Quesada, D. González, I. Alfaro, & E. Cueto (2013). PGD approximations for high-order problems. In *2nd International Workshop on Reduced Basis, POD and PGD Model Reduction Techniques*. Blois, France.

A.4.4 Capítulos de libros

13. Gonzalez, D., Alfaro, I., Quesada, C., Cueto, E., & Chinesta, F. (2015). Vademecums for Real-Time Computational Surgery. In Computational Biomechanics for Medicine (pp. 3-12). Springer International Publishing.

14. Quesada, C., Alfaro, I., González, D., Cueto, E., & Chinesta, F. (2014, October). PGD-Based Model Reduction for Surgery Simulation: Solid Dynamics and Contact Detection. In *International Symposium on Biomedical Simulation* (pp. 193–202). Springer International Publishing.

Bibliography

Bibliography

[1] John A Aucar, Nicholas R Groch, Scott A Troxel, and Steve W Eubanks. A review of surgical simulation with attention to validation methodology. *Surgical Laparoscopy Endoscopy & Percutaneous Techniques*, 15(2):82–89, 2005.

[2] Lana Sturm et al. Surgical simulation for training: Skills transfer to the operating room. ASERNIP-S Report No. 61, ASERNIP-S, Adelaide, South Australia, July 2007.

[3] Leanne M Sutherland, Philippa F Middleton, Adrian Anthony, Jeffrey Hamdorf, Patrick Cregan, David Scott, and Guy J Maddern. Surgical simulation: a systematic review. *Annals of surgery*, 243(3):291–300, 2006.

[4] Shaun Shi Yan Tan and Sudip K Sarker. Simulation in surgery: a review. *Scottish medical journal*, 56(2):104–109, 2011.

[5] Jacob R. Peschman and Jon C. Gould. *Success in Academic Surgery: Developing a Career in Surgical Education*, chapter Opportunities in Simulation Centers, pages 29–36. Springer London, London, 2013.

[6] Amina A. Bouhelal, Hitendra R. H. Patel, and Bijendra Patel. *Simulation Training in Laparoscopy and Robotic Surgery*, chapter Value of Virtual Reality in Medical Education, pages 39–48. Springer London, London, 2012.

[7] Ian Choy and Allan Okrainec. Simulation in surgery: perfecting the practice. *Surgical Clinics of North America*, 90(3):457–473, 2010.

[8] AG Gallagher and O Traynor. Simulation in surgery: opportunity or threat? *Irish journal of medical science*, 177(4):283–287, 2008.

[9] Scott T Rehrig, Kinga Powers, and Daniel B Jones. Integrating simulation in surgery as a teaching tool and credentialing standard. *Journal of Gastrointestinal Surgery*, 12(2):222–233, 2008.

[10] Ryan Owens and Jeffrey M Taekman. Virtual reality, haptic simulators, and virtual environments. In *The Comprehensive Textbook of Healthcare Simulation*, pages 233–253. Springer, 2013.

[11] Justin M Albani and David I Lee. Virtual reality-assisted robotic surgery simulation. *Journal of Endourology*, 21(3):285–287, 2007.

[12] Hoshang Kolivand, Bazli Tomi, Najib Zamri, and Mohd Shahrizal Sunar. *Medical Imaging Technology: Reviews and Computational Applications*, chapter Virtual Surgery, Applications and Limitations, pages 169–195. Springer Singapore, Singapore, 2015.

[13] Anthony G. Gallagher and Gerald C. O'Sullivan. *Fundamentals of Surgical Simulation: Principles and Practice*, chapter Simulations for Procedural Training, pages 39–66. Springer London, London, 2012.

[14] Gerald M. Fried. Fls assessment of competency using simulated laparoscopic tasks. *Journal of Gastrointestinal Surgery*, 12(2):210–212, 2007.

[15] Hervé Delingette and Nicholas Ayache. Soft tissue modeling for surgery simulation. *Handbook of Numerical Analysis*, 2004.

[16] Chad Epps, Marjorie Lee White, and Nancy Tofil. *The Comprehensive Textbook of Healthcare Simulation*, chapter Mannequin Based Simulators, pages 209–232. Springer New York, New York, NY, 2013.

[17] Brian Dunkin, GL Adrales, K Apelgren, and JD Mellinger. Surgical simulation: a current review. *Surgical endoscopy*, 21(3):357–366, 2007.

[18] Anthony G Gallagher, E Matt Ritter, Howard Champion, Gerald Higgins, Marvin P Fried, Gerald Moses, C Daniel Smith, and Richard M Satava. Virtual reality simulation for the operating room: proficiency-based training as a paradigm shift in surgical skills training. *Annals of surgery*, 241(2):364–372, 2005.

[19] SR Dawe, GN Pena, JA Windsor, JAJL Broeders, PC Cregan, PJ Hewett, and GJ Maddern. Systematic review of skills transfer after surgical simulation-based training. *British Journal of Surgery*, 101(9):1063–1076, 2014.

[20] Susan R Dawe, John A Windsor, Joris AJL Broeders, Patrick C Cregan, Peter J Hewett, and Guy J Maddern. A systematic review of surgical skills transfer after simulation-based training: laparoscopic cholecystectomy and endoscopy. *Annals of surgery*, 259(2):236–248, 2014.

[21] Richard M. Satava. Medical virtual reality. The current status of the future. *Studies in health technology and informatics*, 29:100–106, 1996.

[22] Marc Garbey, Barbara L Bass, and S Berceli. Multiscale mechanobiology modeling for surgery assessment. *Acta Mechanica Sinica*, 28(4):1186–1202, 2012.

[23] Sarah Graham, Russell H. Taylor, and Michael Vannier. *Needs Assessment for Computer-Integrated Surgery Systems*, pages 931–939. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[24] Russell H Taylor. *Computer-integrated surgery: technology and clinical applications.* Mit Press, 1996.

[25] David González, Elías Cueto, and Francisco Chinesta. Computational patient avatars for surgery planning. *Annals of biomedical engineering*, 44(1):35–45, 2016.

[26] Adam Wittek, Nicole M Grosland, Grand Roman Joldes, Vincent Magnotta, and Karol Miller. From finite element meshes to clouds of points: A review of methods for generation of computational biomechanics models for patient-specific applications. *Annals of biomedical engineering*, 44(1):3–15, 2016.

[27] Elías Cueto and Francisco Chinesta. Real time simulation for computational surgery: a review. *Advanced Modeling and Simulation in Engineering Sciences*, 1(1):11, 2014.

[28] Stephen D Laycock and AM Day. Recent developments and applications of haptic devices. In *Computer Graphics Forum*, volume 22, pages 117–132. Wiley Online Library, 2003.

[29] Yoseph Bar-Cohen. Haptic devices for virtual reality, telepresence, and human-assistive robotics. *Biologically Inspired Intelligent Robots*, 73, 2003.

[30] Christer Ericson. *Real-time collision detection.* CRC Press, 2004.

[31] Scott L Delp, Peter Loan, Cagatay Basdogan, and Joseph M Rosen. Surgical simulation: An emerging technology for training in emergency medicine. *Presence: Teleoperators and Virtual Environments*, 6(2):147–159, 1997.

[32] Sarthak Misra, K Ramesh, and Allison Okamura. Modeling of tool-tissue interactions for computer-based surgical simulation: a literature review. *Presence*, 17(5):463–491, 2008.

[33] Adam I Levine, Samuel DeMaria Jr, Andrew D Schwartz, and Alan J Sim. *The comprehensive textbook of healthcare simulation.* Springer Science & Business Media, 2013.

[34] Yi-Je Lim, Dhanannjay Deo, Tejinder P Singh, Daniel B Jones, and Suvranu De. In situ measurement and modeling of biomechanical response of human cadaveric soft tissues for physics-based surgical simulation. *Surgical endoscopy*, 23(6):1298–1307, 2009.

[35] Richard M Satava. Historical review of surgical simulation—a personal perspective. *World journal of surgery*, 32(2):141–148, 2008.

[36] U. Meier, O. López, C. Monserrat, M. C. Juan, and M. Alcañiz. Real-time deformable models for surgery simulation: A survey. *Computer Methods and Programs in Biomedicine*, 77(3):183–197, 2005.

[37] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *SIGGRAPH Comput. Graph.*, 21(4):205–214, August 1987.

[38] Sarah FF Gibson and Brian Mirtich. A survey of deformable modeling in computer graphics. Technical report, Mitsubishi Electric Research Laboratories, 1997.

[39] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[40] Lenka Jerábková. *Interactive cutting of finite elements based deformable objects in virtual environments.* PhD thesis, Mathematik, Informatik und Naturwissenschaften der Rheinisch-Westfälischen Technischen Hochschule Aachen, 2007.

[41] Ziyun Li. *Haptic Dissection of Deformable Objects using Extended Finite Element Method.* PhD thesis, School of Electrical Engineering and Computer Science, Faculty of Engineering, University of Ottawa, 2014.

[42] Jernej Barbič and Doug L James. Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM transactions on graphics (TOG)*, volume 24, pages 982–990. ACM, 2005.

[43] Igor Peterlík, Jiri Filipovic, and Ludek Matyska. *Haptic interaction with complex models based on precomputations.* INTECH Open Access Publisher, 2010.

[44] Gaizka San Vicente. *Designing deformable models of soft tissue for virtual surgery planning and simulation using the Mass-Spring Model*. PhD thesis, School of Engineering, University of Navarra, 2011.

[45] Jiří Filipovič. *Algorithms and Methods for Haptic Interaction with Deformable Objects*. PhD thesis, Faculty of Informatics, Masaryk University, 2012.

[46] Ashley Horton. *A meshless method for computer integrated surgery*. PhD thesis, School of Mechanical and Chemical Engineering, University of Western Australia, 2015.

[47] Wu Jichuan. *Fast Physics-Based Simulation of Vascular Surgery*. PhD thesis, Department of Mechanical Engineering, National University of Singapore, 2015.

[48] Aruni Nisansala, Maheshya Weerasinghe, GKA Dias, Damitha Sandaruwan, and Nihal Kodikara. Soft tissue modeling techniques in surgery simulation. *International Journal of Computer and Information Technology*, 4(5), 9 2015.

[49] 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.

[50] Yan Zhuang and John Canny. Real-time and physically realistic simulation of global deformation. In *ACM SIGGRAPH*, volume 99, page 270, 1999.

[51] Yan Zhuang and John Canny. Real-time global deformations. In *Proc. 4th International Workshop on Algorithmic Foundations of Robotics*, volume 127, 2000.

[52] Guillaume Picinbono, Herve Delingette, and Nicholas Ayache. Nonlinear and anisotropic elastic soft tissue models for medical simulation. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1370–1375. IEEE, 2001.

[53] Hualiang Zhong, Mark P Wachowiak, and Terry M Peters. A real time finite element based tissue simulation method incorporating nonlinear elastic behavior. *Computer methods in biomechanics and biomedical engineering*, 8(3):177–189, 2005.

[54] Hualiang Zhong and Terry Peters. A real time hyperelastic tissue model. *Computer methods in biomechanics and biomedical engineering*, 10(3):185–193, 2007.

[55] Xunlei Wu, Michael S. Downes, Tolga Goktekin, and Frank Tendick. Adaptive Nonlinear Finite Elements for Deformable Body Simulation Using Dynamic Progressive Meshes. *Computer Graphics Forum*, 2001.

[56] Gilles Debunne, Mathieu Desbrun, Alan Barr, and Marie-Paule Cani. *Interactive multiresolution animation of deformable models.* Springer, 1999.

[57] Lenka Jeřábková, Torsten Kuhlen, Timm P Wolter, and Norbert Pallua. A voxel based multiresolution technique for soft tissue deformation. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 158–161. ACM, 2004.

[58] Matthieu Nesme, François Faure, and Yohan Payan. Hierarchical multi-resolution finite element model for soft body simulation. In *Biomedical Simulation*, pages 40–47. Springer, 2006.

[59] Wen Wu, Jian Sun, and Pheng Ann Heng. A hybrid condensed finite element model for interactive 3d soft tissue cutting. *Studies in health technology and informatics*, 94:401–403, 2002.

[60] Alessandro Faraci, Fernando Bello, and Ara Darzi. Soft tissue deformation using a hierarchical finite element model. *Studies in health technology and informatics*, 98:92–98, 2003.

[61] Hyun K Kim, David W Rattner, and Mandayam A Srinivasan. The role of simulation fidelity in laparoscopic surgical training. In *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003*, pages 1–8. Springer, 2003.

[62] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, pages 239–246. Canadian Human-Computer Communications Society, 2004.

[63] Matthieu Nesme, Paul G Kry, Lenka Jeřábková, and François Faure. Preserving topology and elasticity for embedded deformable models. *ACM Transactions on Graphics (TOG)*, 28(3):52, 2009.

[64] Anderson Maciel, Tansel Halic, Zhonghua Lu, Luciana P Nedel, and Suvranu De. Using the physx engine for physics-based virtual surgery with force feedback. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 5(3):341–353, 2009.

[65] 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.

[66] Carlos A Felippa. A systematic approach to the element-independent corotational dynamics of finite elements. Technical Report CU-CAS-00-03, Center for Aerospace Structures, College of Engineering, University of Colorado, January 2000.

[67] Zeike A Taylor, Mario Cheng, and Sébastien Ourselin. Real-time nonlinear finite element analysis for surgical simulation using graphics processing units. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2007*, pages 701–708. Springer, 2007.

[68] 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.

[69] 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.

[70] Grand Roman Joldes, Adam Wittek, and Karol Miller. Suite of finite element algorithms for accurate computation of soft tissue deformation for surgical simulation. *Medical Image Analysis*, 13(6):912–919, 2009.

[71] Dhanannjay Deo and Suvranu De. PhyNeSS: A physics-driven neural networks-based surgery simulation system with force feedback. *Proceedings - 3rd Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, World Haptics 2009*, pages 30–34, 2009.

[72] I.T. Jolliffe. *Principal component analysis.* Springer-Verlag New York, 2nd edition, 2002.

[73] Michel Loeve. Fonctions aléatoires du second ordre. In *Processus stochastiques et mouvement Brownien.* Gauthier-Villars, 1965.

[74] Zeike A Taylor, Sébastien Ourselin, and Stuart Crozie. A reduced order finite element algorithm for surgical simulation. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 239–242. IEEE, 2010.

[75] Zeike A Taylor, Stuart Crozier, and Sébastien Ourselin. A reduced order explicit dynamic finite element algorithm for surgical simulation. *Medical Imaging, IEEE Transactions on*, 30(9):1713–1721, 2011.

[76] 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.

[77] F Dogan and M. Serdar Celebi. Real-time deformation simulation of non-linear viscoelastic soft tissues. *Simulation*, 87(3):179–187, 2011.

[78] Annika Radermacher and Stefanie Reese. Proper orthogonal decomposition-based model reduction for non-linear biomechanical analysis. *International Journal of Materials Engineering Innovation*, 4(2):149–165, 2013.

[79] Bruno Cochelin, Noureddine Damil, and Michel Potier-Ferry. The asymptotic-numerical method: an efficient perturbation technique for non-linear structural mechanics. *Revue européenne des éléments finis*, 3(2):281–297, 1994.

[80] B Cochelin, N Damil, and M Potier-Ferry. Asymptotic–numerical methods and pade approximants for non-linear elastic structures. *International journal for numerical methods in engineering*, 37(7):1187–1213, 1994.

[81] Siamak Niroomandi, Iciar Alfaro, Elias Cueto, and Francisco Chinesta. Model order reduction for hyperelastic materials. *International Journal for Numerical Methods in Engineering*, 81(9):1180, 2010.

[82] Siamak Niroomandi, Icíar Alfaro, Elías Cueto, and Francisco Chinesta. Accounting for large deformations in real-time simulations of soft tissues based on reduced-order models. *Computer Methods and Programs in Biomedicine*, 105(1):1–12, 2012.

[83] 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.

[84] 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 modeling of complex fluids. *Journal of Non-Newtonian Fluid Mechanics*, 139(3):153–176, 2006.

[85] 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.

[86] 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.

[87] Francisco Chinesta, Amine Ammar, and Elías Cueto. Recent Advances and New Challenges in the Use of the Proper Generalized Decomposition for Solving Multidimensional Models. *Archives of Computational Methods in Engineering*, 17(4):327–350, 2010.

[88] Francisco Chinesta, Pierre Ladeveze, and Elías Cueto. A Short Review on Model Order Reduction Based on Proper Generalized Decomposition. *Archives of Computational Methods in Engineering*, 18(4):395–404, October 2011.

[89] 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.

[90] 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.

[91] Hervé Delingette. Biquadratic and quadratic springs for modeling st venant kirchhoff materials. In Fernando Bello and Philip J. Edwards, editors, *ISBMS*, volume 5104 of *Lecture Notes in Computer Science*, pages 40–48. Springer, 2008.

[92] S De and KJ Bathe. The method of finite spheres. *Computational Mechanics*, 25(4):329–345, 2000.

[93] S De, J Kim, and MA Srinivasan. A meshless numerical technique for physically based real time medical. *Medicine Meets Virtual Reality 2001: Outer Space, Inner Space, Virtual Space*, 81:113, 2001.

[94] Suvranu De, Jung Kim, Yi-Je Lim, and Mandayam A Srinivasan. The point collocation-based method of finite spheres (pcmfs) for real time surgery simulation. *Computers & structures*, 83(17):1515–1525, 2005.

[95] Yi Je Lim and Suvranu De. Real time simulation of nonlinear tissue response in virtual surgery using the point collocation-based method of finite spheres. *Computer Methods in Applied Mechanics and Engineering*, 196(31-32):3011–3024, 2007.

[96] Suleiman Banihani, Timon Rabczuk, and Thakir Almomani. Pod for real-time simulation of hyperelastic soft biological tissue using the point collocation method of finite spheres. *Mathematical Problems in Engineering*, 2013, 2013.

[97] 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.

[98] Y-J Lim and Suvranu De. Nonlinear tissue response modeling for physically realistic virtual surgery using paff. In *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics Conference*, pages 479–480. IEEE, 2005.

[99] Sarah F Frisken-Gibson. Using linked volumes to model object collisions, deformation, cutting, carving, and joining. *Visualization and Computer Graphics, IEEE Transactions on*, 5(4):333–348, 1999.

[100] 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.

[101] 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.

[102] 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.

[103] Gerrit Voß, James K Hahn, Wolfgang Müller, and Rob Lindeman. Virtual cutting of anatomical structures. *Studies in health technology and informatics*, pages 381–383, 1999.

[104] Daniel Bielser and Markus H Gross. Interactive simulation of surgical cuts. In *Computer Graphics and Applications, 2000. Proceedings. The Eighth Pacific Conference on*, pages 116–442. IEEE, 2000.

[105] Andrew B Mor and Takeo Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2000*, pages 598–607. Springer, 2000.

[106] Fabio Ganovelli, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A multiresolution model for soft objects supporting interactive cuts and lacerations. In *Computer Graphics Forum*, volume 19, pages 271–281. Citeseer, 2000.

[107] Daniel Bielser, Pascal Glardon, Matthias Teschner, and Markus Gross. A state machine for real-time cutting of tetrahedral meshes. *Graphical Models*, 66(6):398–417, 2004.

[108] Martin Seiler, Denis Steinemann, Jonas Spillmann, and Matthias Harders. Robust interactive cutting based on an adaptive octree simulation mesh. *The Visual Computer*, 27(6-8):519–529, 2011.

[109] Han-Wen Nienhuys and A Frank van der Stappen. Combining finite element deformation with cutting for surgery simulations. In *EuroGraphics short presentations*, pages 43–52. Citeseer, 2000.

[110] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 141–151, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[111] Alex Lindblad and George Turkiyyah. A physically-based framework for real-time haptic cutting and interaction with 3d continuum models. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 421–429. ACM, 2007.

[112] Martin Wicke, Mario Botsch, and Markus Gross. A finite element method on convex polyhedra. In *Computer Graphics Forum*, volume 26, pages 355–364. Wiley Online Library, 2007.

[113] Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. Polyhedral finite elements using harmonic basis functions.

In *Computer Graphics Forum*, volume 27, pages 1521–1529. Wiley Online Library, 2008.

[114] Han-Wen Nienhuys and A Frank van der Stappen. A surgery simulation supporting cuts and finite element deformation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001*, pages 145–152. Springer, 2001.

[115] Denis Steinemann, Matthias Harders, Markus Gross, and Gabor Szekely. Hybrid cutting of deformable solids. In *Virtual Reality Conference, 2006*, pages 35–42. IEEE, 2006.

[116] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. In *ACM SIGGRAPH 2005 Courses*, page 4. ACM, 2005.

[117] Eftychios Sifakis, Kevin G Der, and Ronald Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 73–80. Eurographics Association, 2007.

[118] Ted Belytschko and Tom Black. Elastic crack growth in finite elements with minimal remeshing. *International journal for numerical methods in engineering*, 45(5):601–620, 1999.

[119] Lara M Vigneron, Jacques G Verly, and Simon K Warfield. Modelling surgical cuts, retractions, and resections via extended finite element method. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2004*, pages 311–318. Springer, 2004.

[120] Lenka Jeřábková and Torsten Kuhlen. Stable cutting of deformable objects in virtual environments using xfem. *IEEE Computer Graphics and Applications*, 2:61–71, 2009.

[121] Ziyun Li, Ling Jin, Jochen Lang, and Emil Petriu. Dissection simulation of deformable objects using the extended finite element method. In *Haptic, Audio and Visual Environments and Games (HAVE), 2014 IEEE International Symposium on*, pages 1–6. IEEE, 2014.

[122] Hui Zhang, Shahram Payandeh, and John Dill. On cutting and dissection of virtual deformable objects. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3908–3913. IEEE, 2004.

[123] Denis Steinemann, Miguel A Otaduy, and Markus Gross. Fast arbitrary splitting of deforming objects. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 63–72. Eurographics Association, 2006.

[124] Nico Pietroni, Fabio Ganovelli, Paolo Cignoni, and Roberto Scopigno. Splitting cubes: a fast and robust technique for virtual cutting. *The Visual Computer*, 25(3):227–239, 2009.

[125] Xia Jin, Grand Roman Joldes, Karol Miller, King H Yang, and Adam Wittek. Meshless algorithm for soft tissue cutting in surgical simulation. *Computer methods in biomechanics and biomedical engineering*, 17(7):800–811, 2014.

[126] François Boux de Casson and Christian Laugier. Simulating 2d tearing phenomena for interactive medical surgery simulators. In *ca*, page 9. IEEE, 2000.

[127] Roger Webster, Joseph Sassani, Rod Shenk, Matt Harris, Jesse Gerber, Aaron Benson, John Blumenstock, Chad Billman, and Randy Haluck. Simulating the continuous curvilinear capsulorhexis procedure during cataract surgery on the eyesi system. *Stud Health Technol Inform*, 111:592–595, 2005.

[128] 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.

[129] Kumar Vemaganti, Grand R Joldes, Karol Miller, and Adam Wittek. Total lagrangian explicit dynamics-based simulation of tissue tearing. In *Computational Biomechanics for Medicine*, pages 63–72. Springer, 2011.

[130] Wenguang Li. Damage models for soft tissues: a survey. *Journal of Medical and Biological Engineering*, pages 1–23, 2016.

[131] Alex J Lindblad, George M Turkiyyah, Ganesh Sankaranarayanan, Suzanne J Weghorst, and Daniel Berg. Two-handed next generation suturing simulator. *Studies in health technology and informatics*, 98:215–220, 2003.

[132] Fuhan Shi and Shahram Payandeh. On suturing simulation with haptic feedback. In *International Conference on Human Haptic Sensing and Touch Enabled Computer Applications*, pages 599–608. Springer, 2008.

[133] Jeffrey Berkley, George Turkiyyah, Daniel Berg, Mark Ganter, and Suzanne Weghorst. Real-time finite element modeling for surgery simulation: An application to virtual suturing. *IEEE Transactions on visualization and computer graphics*, 10(3):314–325, 2004.

[134] Kup-Sze Choi, Sze-Ho Chan, and Wai-Man Pang. Virtual suturing simulation based on commodity physics engine for medical learning. *Journal of medical systems*, 36(3):1781–1793, 2012.

[135] PENG Su, YANG Yang, LEIYU Zhang, and LONG Huang. Biomechanical simulation of needle insertion into cornea based on distortion energy failure criterion. *Acta Bioeng. Biomech*, 18(1):1–30, 2016.

[136] Shan Jiang and Xingji Wang. Mechanics-based interactive modeling for medical flexible needle insertion in consideration of nonlinear factors. *Journal of Computational and Nonlinear Dynamics*, 11(1):011004, 2016.

[137] Nathan Wilson, Kenneth Wang, Robert W Dutton, and Charles Taylor. A software framework for creating patient specific geometric models from medical imaging data for simulation based medical planning of vascular surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 449–456. Springer, 2001.

[138] Lee-Ying Wu. *Bleeding Effects for Flinders Endoscopic Sinus Surgery Simulator*. PhD thesis, Flinders University–Adelaide, Australia, 2013.

[139] Mohamed Guiatni, Vincent Riboulet, Christian Duriez, Abderrahmane Kheddar, and Stéphane Cotin. A combined force and thermal feedback interface for minimally invasive procedures simulation. *Ieee/Asme Transactions On Mechatronics*, 18(3):1170–1181, 2013.

[140] Thomas E Turner, Santiago Schnell, and Kevin Burrage. Stochastic approaches for modelling in vivo reactions. *Computational biology and chemistry*, 28(3):165–178, 2004.

[141] Amine Ammar, Elías Cueto, and Francisco Chinesta. Reduction of the chemical master equation for gene regulatory networks using proper generalized decompositions. *International Journal for Numerical Methods in Biomedical Engineering*, 28(9):960–973, 2012.

[142] Jose V Aguado, Antonio Huerta, Francisco Chinesta, Adrien Leygue, and Elias Cueto. A fully-separated PGD algorithm for nonlinear problems. In *11th World Congress on Computational Mechanics (WCCM XI)*, 2014.

[143] Wilhelmus HA Schilders, Henk A Van der Vorst, and Joost Rommes. *Model order reduction: theory, research aspects and applications*, volume 13. Springer, 2008.

[144] David Ryckelynck, Francisco Chinesta, E Cueto, and Amine Ammar. On the a priori model reduction: Overview and recent developments. *Archives of Computational methods in Engineering*, 13(1):91–128, 2006.

[145] Yvon Maday and Einar M Rønquist. A reduced-basis element method. *Journal of scientific computing*, 17(1-4):447–459, 2002.

[146] Jens Lohne Eftang. *Reduced basis methods for parametrized partial differential equations*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2011.

[147] José Vicente Aguado. *Advanced strategies for the separated formulation of problems in the Proper Generalized Decomposition framework*. PhD thesis, École Centrale de Nantes, 2015.

[148] Antoine Dumon, Cyrille Allery, and Amine Ammar. Proper general decomposition (PGD) for the resolution of Navier–Stokes equations. *Journal of Computational Physics*, 230(4):1387–1407, 2011.

[149] 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.

[150] Amine Ammar and Francisco Chinesta. Circumventing curse of dimensionality in the solution of highly multidimensional models encountered in quantum mechanics using meshfree finite sums decomposition. In *Meshfree Methods for Partial Differential Equations IV*, pages 1–17. Springer, 2008.

[151] F Chinesta, P Ladevèze, A Ammar, E Cueto, and A Nouy. Proper generalized decomposition in extreme simulations: towards a change of paradigm in computational mechanics? *IACM Expressions*, 26(09):2–7, 2009.

[152] David González, Amine Ammar, Francisco Chinesta, and Elías Cueto. Recent advances on the use of separated representations. *International Journal for Numerical Methods in Engineering*, 81(5):637, 2010.

[153] Etienne Pruliere, Francisco Chinesta, and Amine Ammar. On the deterministic solution of multidimensional parametric models using the proper generalized decomposition. *Mathematics and Computers in Simulation*, 81(4):791–810, 2010.

[154] Francisco Chinesta, Amine Ammar, Adrien Leygue, and Roland Keunings. An overview of the proper generalized decomposition with applications in computational rheology. *Journal of Non-Newtonian Fluid Mechanics*, 166(11):578–592, 2011.

[155] Francisco Chinesta and Elías Cueto. *PGD-based modeling of materials, structures and processes*. Springer, 2014.

[156] Francisco Chinesta and Pierre Ladevèze. Separated representations and PGD-based model reduction, 2014.

[157] 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.

[158] C. Bernoulli. *Vademecum des Mechanikers*. Cotta, 1836.

[159] Thomas Lloyd David Croft. *Proper generalised decompositions: theory and applications*. PhD thesis, School of Mathematics, Cardiff University, 2015.

[160] 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.

[161] Teeranoot Chanthasopeephan, Jaydev P Desai, and Alan CW Lau. Modeling soft-tissue deformation prior to cutting for surgical simulation: finite element analysis and study of cutting parameters. *IEEE transactions on biomedical engineering*, 54(3):349–359, 2007.

[162] Hadrien Courtecuisse, Hoeryong Jung, Jérémie Allard, Christian Duriez, Doo Yong Lee, and Stéphane Cotin. Gpu-based real-time soft tissue deformation with cutting and haptic feedback. *Progress in biophysics and molecular biology*, 103(2):159–168, 2010.

[163] He-xiang Wang, Ai-min Hao, Liu Xue-mei, et al. Real-time cutting method for soft tissue based on tled algorithm. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 3, pages V3–393. IEEE, 2010.

[164] GJ van Zwieten, EH van Brummelen, KG van der Zee, MA Gutiérrez, and RF Hanssen. Discontinuities without discontinuity: the weakly-enforced slip method. *Computer Methods in Applied Mechanics and Engineering*, 271:144–166, 2014.

[165] Jeong-Hoon Song and Ted Belytschko. Cracking node method for dynamic fracture with finite elements. *International Journal for Numerical Methods in Engineering*, 77(3):360–385, 2009.

[166] Claude Le Bris, Tony Lelievre, and Yvon Maday. Results and questions on a nonlinear approximation approach for solving high-dimensional partial differential equations. *Constructive Approximation*, 30(3):621–651, 2009.

[167] 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*, 2016.

[168] Natarajan Sukumar and Ted Belytschko. Arbitrary branched and intersecting cracks with the extended finite element method. *Int. J. Numer. Meth. Eng*, 48:1741–1760, 2000.

[169] J Fish. The s-version of the finite element method. *Computers & Structures*, 43(3):539–547, 1992.

[170] Jernej Barbič and Doug James. Time-critical distributed contact for 6-dof haptic rendering of adaptively sampled reduced deformable models. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 171–180. Eurographics Association, 2007.

[171] 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.

[172] Anna Pandolfi and Gerhard A Holzapfel. Three-dimensional modeling and computational analysis of the human cornea considering distributed collagen fibril orientations. *Journal of biomechanical engineering*, 130(6):061006, 2008.

[173] V Alastrué, B Calvo, E Pena, and M Doblaré. Biomechanical modeling of refractive corneal surgery. *Journal of biomechanical engineering*, 128(1):150–160, 2006.

[174] Elena Lanchares, Begoña Calvo, José A Cristóbal, and Manuel Doblaré. Finite element simulation of arcuates for astigmatism correction. *Journal of biomechanics*, 41(4):797–805, 2008.

[175] 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.

[176] J. Lemaitre. *A Course on Damage Mechanics*. Springer Berlin Heidelberg, 1996.

[177] G.Z. Voyiadjis and P.I. Kattan. *Damage Mechanics*. Mechanical Engineering. CRC Press, 2005.

[178] J Lemaitre and JL Chaboche. *Mécanique des matériaux solides*. Dunod, Paris, 1985.

[179] 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.

[180] Javier Bonet and Richard D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, Cambridge, 2008.

[181] 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, 2015.

[182] 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 their relationships to the structure and composition of the extracellular matrix. *American Journal of Physiology-Endocrinology and Metabolism*, 305(12):E1427–E1435, 2013.

[183] 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.

[184] Amit Gefen and Einat Haberman. Viscoelastic properties of ovine adipose tissue covering the gluteus muscles. *Journal of biomechanical engineering*, 129(6):924–930, 2007.

[185] Carlos Quesada, Guangtao Xu, David González, Icıar Alfaro, Adrien Leygue, Michel Visonneau, Elıas Cueto, and Francesco Chinesta. Un método de descomposición propia generalizada para operadores diferenciales de alto orden. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 31(3):188–197, 2015.

[186] C Quesada, D González, I Alfaro, E Cueto, A Huerta, and F Chinesta. Real-time simulation techniques for augmented learning in science and engineering. *The Visual Computer*, pages 1–15, 2015.

[187] Carlos Quesada, Icíar Alfaro, David González, Elías Cueto, and Francisco Chinesta. Pgd-based model reduction for surgery simulation: Solid dynamics and contact detection. In *International Symposium on Biomedical Simulation*, pages 193–202. Springer, 2014.