NTNU – Trondheim
Norwegian University of
Science and Technology

Simulation of crack propagation using isogeometric analysis applied with NURBS and LR B-splines

Oda Kulleseid Nilsen

Master of Science in Physics and Mathematics
Submission date: June 2012
Supervisor: Trond Kvamsdal, MATH
Co-supervisor: Kjetil André Johannessen, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

This report features the isogeometric finite element method applied to the elastodynamic problem in a brittle medium with a potential for cracking. Griffith's theory for fracturing is used. The development of the model is outlined, complete with the Euler-Lagrange equations. The cracking is described with a phase field supplemented with a history field, contrary to the usual way of building the crack directly into the geometry by modification of the basis, facilitating the use of isogeometric analysis even with simplistic basis functions such as Non-Uniform Rational B-Splines (NURBS). The introduction of the crack-phase field results in non-linearity in the coupled problem. The problem is semi-discretized, upon which the spatial sub-problem is treated with isogeometric analysis. The numerical time-stepping solution routine is built around the Newton-Raphson method, but includes both pre-conditioning and correctors and is known as the predictor/multi-corrector time integration scheme. The Jacobian of the semi-discretized system (needed for the Newton-Raphson iteration) is developed analytically. In addition to the numerical tests with NURBS as our basis, we will also test the method with Locally Refined B-splines (LR B-Splines), ensuring better resolution along the crack path. The LR B-spline represents an alternative to the more commonly used T-Spline.

Sammendrag

Rapporten inneholder isogeometrisk endelig elementmetode anvendt på det elastodynamiske problemet i et sprøtt medium med potensial for sprekkdannelse. Griffiths teori for sprekkdannelse er benyttet. Utledningen av modellen er gitt, komplett med Euler-Lagrangelikningene. Oppsprekkingen er beskrevet ved et fasefelt og et historiefelt, i motsettning til den mest benyttede metoden hvor man i stedet bygger sprekken direkte inn i geometrien ved å modifisere basis. Dette fasiliterer bruk av selv en så enkel basis som ikke-uniform rasjonell b-spline (NURBS). Introduksjonen av sprekkfasefeltet resulterer i ikkelinearitet i det koplede problemet. Problemet blir først semidiskretisert, hvorpå det romlige underproblemet blir behandlet med isogeometrisk analyse. Den numeriske løsningen i tid er bygget rundt Newton-Raphson metoden, men inkluderer både prekondisjonering og korrektorer. Jakobianten til det semidiskretiserte systemet blir utviklet analytisk (denne behøves til Newton-Raphson iterasjonen). I tillegg til testene utført med NURBS som basis er det også utført en utprøving av metoden med lokalt forfinede b-splines (LR B-spline) benyttet som basis. Dette gjør at man kan spesifisere bedre oppløsning eksklusivt langs sprekklinjen. LR B-spline representerer et alternativ til den mer brukte T-spline.

Preface

It is assumed throughout that the reader is familiar with the Finite Element Method (FEM), as this is the building brick upon which isogeometric analysis is built. For those not familiar with Non-Uniform Rational B-Splines (NURBS) and its use within isogeometric analysis, we recommend starting off with the relatively thorough introduction included in the appendices. Although there is given an introduction on NURBS in the appendices there will not be given a proper introduction to the LR B-spline as the subject is outside the scope of this report, instead I refer to [7, 12].

Notice that throughout the report I assume the convention that scalars and tensors (as well as scalar and tensor functions) that belong to a set is numbered in **boldface**. I do this to clarify for myself that the subscript indicates that it is a part of a set as opposed to an entry of a tensor (this distinction is of course somewhat artificial, but very helpful none the less).

The following report is the Master thesis with course code TMA4910 Numerical Mathematics, written in the spring semester of 2012.

The thesis is written under the supervision of associate professor Trond Kvamsdal and PhD fellow Kjetil André Johannessen. The idea for this thesis was theirs, and they encouraged me with much enthusiasm to keep on when I was about to abandon the ship for an easier way out. Kjetil kindly supplied me with all the code and framework needed to generate and refine LR B-spline meshes, and helped with the installations and any small questions I had, for which I am very grateful.

Contents

List of Figures

List of Tables

1. Introduction

The term *isogeometric analysis* was coined by its inventor Tom Hughes et al. in [11]. Cottrell, Hughes et al. wrote the up until recently only text book on the subject [6]. Isogeometric analysis is in its concepts similar to the finite element method (FEM). The key difference lies within the geometry, and the basis used. In isogeometric analysis the geometry is always expressed using the same basis that will be used to approximate the solution. This is what we refer to as the "isogeometric concept". The idea itself is very simplistic and should be easy to understand, to predict its consequences we must look a bit further.

The main motivation behind isogeometric analysis is to spend less time on the mesh and model generation and free up time for analysis. By using the same basis in the geometry and numerical solution we can avoid the often costly step of discretizing (meshing) the original design. Furthermore, we avoid introducing any errors that would follow by discretizing the design. In certain applications this can be of great importance, i.e. boundary layer effects will be sensitive to the geometry of the boundary.

To model the fracturing process we will use Griffith's theory of crack propagation superimposed onto the theory of linear elasticity. Griffith's theory conditions the formation/prolongation of a crack upon a critical energy release rate. At the backdrop of everything we do lies a variational approach to this fracturing process, thoroughly described in by Bourdin et al. in [4], see also [5].

To account for existing fractures in the material the most intuitive approach is to model the fracture by including it directly into the geometry or the basis used. One early attempt was the embedded discontinuity method yielding a discontinuous fracture surface [9, 21]. Today the most common approach is possibly the extended finite element method allowing for a smooth fracture [2, 14]. This method has also been used successfully with isogeometric analysis [15, 10].

Another opportunity is to instead model the fracture by a damage field,

a relatively new approach stemming from [4]. Although the concept is harder to grasp the implementation is straight forward. For an excellent introduction to the damage field we will use (later dubbed the crack phase field) see Miehe, Hofacker and Welschinger [16, 18]. To obtain a set of governing equations we will minimize a energy functional by use of the Euler-Lagrange equations, following the footsteps of Borden, Verhoosel, Scott, Hughes and Landis in [3].

The first attempts at isogeometric analysis were performed with NURBS as the choice of basis [11]. (By selecting weights equal to unity they degenerate to B-Splines.) This choice of basis does not facilitate local refinement, as dictated by the tensor-product like structure of any NURBS mesh. T-Splines are another commonly used choice of basis that allow for local refinement [20, 19, 1]. However, implementation of effective refinement strategies is not straight forward. Dörfel et al. [8] have shown that existence of a worst case scenario where the local refinement turns global. A newcomer to the game is LR B-splines by Dokken et al. [7], see also [12]. In this thesis we will undertake isogeometric analysis on crack propagation with both NURBS and LR B-splines.

2. Modelling the fracturing process

2.1. Fracturing described by a phase field

The most intuitive approach to modelling a crack, Γ, is to track its progress and build it into the geometry as it evolves. Although the concept is simple enough the implementation is all the more challenging. An alternative approach is to model the crack with a phase field $c(\mathbf{x}, t)$, that at any given point takes on a value between 0 and 1, nil indicating a crack and one indicating undamaged material. As we force the damage field to yield a sharper and sharper crack topology, with decreasing areas containing intermediate values, this becomes a viable option.

The tricky part comes when we want to formulate an elasticity problem on the cracking domain. The work integral will contain an integral over the crack face. This is eloquently solved by introducing a crack surface density function $\gamma(c, \nabla c)$, that takes in the phase field and its spatial derivative as parameters.

$$\gamma(c, \nabla c) = \frac{1}{4e}(c-1)^2 + e|\nabla c|^2, \tag{2.1}$$

where we have $c(\mathbf{x}, t) = 0$ at $\mathbf{x} \in \Gamma(t)$. e is a model parameter that controls the amount of smearing of the crack (we use e instead of the more common ϵ to avoid any confusion with the strain). The underlying idea is that running e towards zero the area integral over the crack density function $\gamma(c, \nabla c)$ will converge towards the line integral tracing the crack surface. For details refer to [16].

Observe that we have yet to obtain the equation governing the crack phase field, this will be undertaken in section 2.3.4.

2.2. Displacement field

We choose to describe the displacements of our bounded domain/body of interest in terms of a separate vector-field $\mathbf{u}(\mathbf{x}, t)$.

The strains will then be defined the following way:

$$\epsilon_{ij} = u_{(i,j)} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{2.2}$$

often written in matrix notation in terms of the symmetric nabla-operator, ∇_s

$$\boldsymbol{\epsilon} = \nabla_s \mathbf{u} = \frac{1}{2}\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right). \tag{2.3}$$

2.3. Multivariate Euler-Lagrange equations

2.3.1. Elastic energy

By the standard theory of elasticity for isotropic solids we can represent the elastic energy density as

$$\psi_0(\boldsymbol{\epsilon}, c) = \sum_i \sum_j \left[\frac{\lambda}{2}\epsilon_{ii}\epsilon_{jj} + \mu\epsilon_{ij}\epsilon_{ji}\right] \tag{2.4}$$

$$= \frac{\lambda}{2}\mathrm{tr}(\boldsymbol{\epsilon})^2 + \mu \cdot \mathrm{tr}(\boldsymbol{\epsilon}^2), \tag{2.5}$$

where λ is Lamé's first parameter and μ is the shear modulus (also called Lamé's second parameter). But we must not forget to include changes in the elastic energy due to cracking as material stiffness will develop in the failure zone. We will separate between tensile and compressive loading, as only the first is expected to yield further cracking. The first step is to perform a splitting of the elastic energy in the base case:

$$\psi_0(\epsilon) = \underbrace{\psi^+(\boldsymbol{\epsilon})}_{\text{tensile}} + \underbrace{\psi^-(\boldsymbol{\epsilon})}_{\text{compressive}}. \tag{2.6}$$

To perform this splitting we must first identify the principal strains $\epsilon_{\mathbf{a}}$ by performing a spectral decomposition of the strain, $\boldsymbol{\epsilon}$:

$$\boldsymbol{\epsilon} = \sum_a \epsilon_{\mathbf{a}} \cdot \mathbf{n}_{\mathbf{a}} \otimes \mathbf{n}_{\mathbf{a}}, \tag{2.7}$$

where $\{\mathbf{n}_{\mathbf{a}}\}$ form an orthonormal basis, with each $\mathbf{n}_{\mathbf{a}}$ being identified as a direction of principal strain. We may then perform a splitting of the strain into a "positive" and a "negative" part

$$\boldsymbol{\epsilon}_+(\boldsymbol{\epsilon}) := \sum_a \frac{\epsilon_{\mathbf{a}} + |\epsilon_{\mathbf{a}}|}{2} \mathbf{n}_{\mathbf{a}} \otimes \mathbf{n}_{\mathbf{a}}, \tag{2.8a}$$

$$\boldsymbol{\epsilon}_-(\boldsymbol{\epsilon}) := \sum_a \frac{\epsilon_{\mathbf{a}} - |\epsilon_{\mathbf{a}}|}{2} \mathbf{n}_{\mathbf{a}} \otimes \mathbf{n}_{\mathbf{a}}, \tag{2.8b}$$

such that $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}_+ + \boldsymbol{\epsilon}_-$. Using these definitions we may form the tensile and compressive components of the elastic energy

$$\psi^+(\boldsymbol{\epsilon}) := \frac{\lambda}{2} \left(\frac{\mathrm{tr}(\boldsymbol{\epsilon}) + |\mathrm{tr}(\boldsymbol{\epsilon})|}{2} \right)^2 + \mu \cdot \mathrm{tr}(\boldsymbol{\epsilon}_+^2), \tag{2.9a}$$

$$\psi^-(\boldsymbol{\epsilon}) := \frac{\lambda}{2} \left(\frac{\mathrm{tr}(\boldsymbol{\epsilon}) - |\mathrm{tr}(\boldsymbol{\epsilon})|}{2} \right)^2 + \mu \cdot \mathrm{tr}(\boldsymbol{\epsilon}_-^2). \tag{2.9b}$$

Allow the first term to be driven towards zero in the case it is in the materials failure zone (as modelled by the phase-field). We want $\psi(\boldsymbol{\epsilon}, c) = \psi_0(\boldsymbol{\epsilon})$ in the undamaged material (where $c = 1$) and will therefore follow Borden [3].

$$\psi(\boldsymbol{\epsilon}, c) = [(1-k)c^2 + k]\psi^+(\boldsymbol{\epsilon}) + \psi^-(\boldsymbol{\epsilon}), \tag{2.10}$$

with artificial parameter $k << 1$ to enhance numerical stability. (We will not make use of this and choose $k = 0$.)

2.3.2. Fracture energy

By Griffith's theory on fracture the fracture energy depends on the size of the crack surface. Therefore the energy/work consumed to create the

diffusive crack topology might be expressed as a product of the Griffith-type critical energy release rate and the crack density function

$$G_c \cdot \gamma(c, \nabla c), \tag{2.11}$$

where G_c is the critical fracture energy density

$$\mathcal{G}_c = \frac{[\text{ energy }]}{[\text{ fracture surface }]}.$$

2.3.3. Kinetic energy

To describe a dynamic system we must include the kinetic energy

$$\frac{1}{2}\rho\dot{\mathbf{u}}^2, \tag{2.12}$$

where ρ is the density of the material and $\dot{\mathbf{u}} = \partial\mathbf{u}/\partial t$.

2.3.4. The work integral

We combine the contributions in a work integral, where we integrate over both the full domain and our designated time interval. (In the literature the integration over time is often left implicit.)

$$\begin{aligned}
\mathcal{I} &= \int\int_\Omega \overbrace{\mathcal{L}(\mathbf{x}, t, \mathbf{u}, \dot{\mathbf{u}}, c, \nabla c)}^{\text{the Lagrangian}} \,\mathrm{d}\mathbf{x}\,\mathrm{dt} \\
&= \int\int_\Omega \underbrace{\frac{1}{2}\rho\dot{\mathbf{u}}\dot{\mathbf{u}}}_{\text{kinetic energy}} - \underbrace{[(1-k)c^2 + k]\psi^+(\epsilon(\mathbf{u})) - \psi^-(\epsilon(\mathbf{u}))}_{\text{elastic energy}} \,\mathrm{d}\mathbf{x}\,\mathrm{dt} \\
&\quad - \int\int_\Omega \underbrace{\mathcal{G}_c \underbrace{\left[\frac{(c-1)^2}{4e} + e\sum_i \left(\frac{\partial c}{\partial x_i}\right)^2\right]}_{=\gamma(c,\nabla c)}}_{\text{fracture energy}} \,\mathrm{d}\mathbf{x}\,\mathrm{dt}.
\end{aligned}$$

We need the following definitions to proceed:

$$\dot{\mathbf{u}} := \frac{\partial \mathbf{u}}{\partial t} \tag{2.13a}$$

$$\dot{c} := \frac{\partial c}{\partial t} \tag{2.13b}$$

$$\mathbf{u}_{x_i} := \frac{\partial \mathbf{u}}{\partial x_i} \tag{2.13c}$$

$$c_{x_i} := \frac{\partial c}{\partial x_i}. \tag{2.13d}$$

The Euler-Lagrange functions minimize the work integral. We write out the Euler-Lagrange equations for multiple functions; $\mathbf{u}(\mathbf{x}, t)$ and $c(\mathbf{x}, t)$, and multiple variables; $\mathbf{x}$ and t.

$$\underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{u}}}_{=0} - \sum_i \frac{\partial}{\partial x_i}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{u}_{x_i}}\right) - \frac{\partial}{\partial t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{u}}}\right) = \mathbf{0}, \tag{2.14a}$$

$$\frac{\partial \mathcal{L}}{\partial c} - \sum_i \frac{\partial}{\partial x_i}\left(\frac{\partial \mathcal{L}}{\partial c_{x_i}}\right) - \underbrace{\frac{\partial}{\partial t}\left(\frac{\partial \mathcal{L}}{\partial \dot{c}}\right)}_{=0} = 0. \tag{2.14b}$$

Insertion of the Lagrangian, $\mathcal{L}$, and some rearranging yields the coupled system

$$\sum_j \frac{\partial \sigma_{ij}}{\partial x_j} = \rho \ddot{u}_i \quad \forall i, \tag{2.15a}$$

$$\left(\frac{4e(1-k)\psi^+}{\mathcal{G}_c} + 1\right) c - 4e^2 \sum_j \frac{\partial^2 c}{\partial x_j^2} = 1. \tag{2.15b}$$

where we used the the following definition

$$\sigma_{ij} := \frac{\partial \psi}{\partial \epsilon_{ij}} = [(1-k)c^2 + k]\frac{\partial \psi^+}{\partial \epsilon_{ij}} + \frac{\partial \psi^-}{\partial \epsilon_{ij}}. \tag{2.16}$$

To calculate the derivatives we introduce the short notation $\langle x \rangle_+ := (x+|x|)/2$ for the ramp function and $\langle x \rangle_- := (x-|x|)/2$ for its counterpart. Let "$\mathbf{I}$" denote the second order identity tensor. We have that

$$\frac{\partial \psi^+}{\partial \boldsymbol{\epsilon}} = \lambda \langle \mathrm{tr}(\boldsymbol{\epsilon}) \rangle_+ \mathbf{I} + 2\mu \boldsymbol{\epsilon}_+, \tag{2.17a}$$

$$\frac{\partial \psi^-}{\partial \boldsymbol{\epsilon}} = \lambda \langle \mathrm{tr}(\boldsymbol{\epsilon}) \rangle_- \mathbf{I} + 2\mu \boldsymbol{\epsilon}_-. \tag{2.17b}$$

The first term of each expression is easily calculated while the second term is substantially more involved.

2.4. Strain-History field

We must somehow ensure $\Gamma(t) \subset \Gamma(t + \Delta t)$. We make the observation that the tensile elastic energy ψ^+ clearly drives the evolution of the crack as all other factors in equation (2.15b) describing the crack-density c are constants. The "ad hoc" solution: Replace ψ^+ by its maximum through history, resulting in the history field $\mathcal{H}(t, \mathbf{x})$ [16].

The equations that together enforce "maximum through history" are

$$\psi^+ - \mathcal{H} \leq 0, \tag{2.18a}$$

$$\dot{\mathcal{H}} \geq 0, \tag{2.18b}$$

$$\dot{\mathcal{H}}(\psi^+ - \mathcal{H}) = 0. \tag{2.18c}$$

(Numerically we will make a simplifying assumption.)

The central idea of an algorithmic decoupling of the coupled equations (2.15a) and (2.15b) bases on an approximation of the current history field $\mathcal{H}_{n+1}$ on the displacement at the previous time step t_n [16].

2.5. Strong form formulation

As before we have the equations of motion

$$\sum_j \frac{\partial \sigma_{ij}}{\partial x_j} = \rho \ddot{u}_i \quad \forall i, \tag{2.19a}$$

$$\left(\frac{4e(1-k)\mathcal{H}}{\mathcal{G}_c} + 1 \right) c - 4e^2 \sum_j \frac{\partial^2 c}{\partial x_j^2} = 1. \tag{2.19b}$$

We must also specify sufficient boundary conditions:

$$u_i = g_i \text{ on } \partial\Omega_g \quad \forall i, \tag{2.19c}$$

$$\sum_j \sigma_{ij} n_j = h_i \text{ on } \partial\Omega_h \quad \forall i, \tag{2.19d}$$

$$\frac{\partial c}{\partial x_i} n_i = 0 \text{ on } \partial\Omega \quad \forall i, \tag{2.19e}$$

where n_j are the components of the outward normal vector at the boundary. It is also possible to have Neumann boundary conditions in one direction of the displacement field and Dirichlet boundary conditions in another but we will not make use of this opportunity in our experiments. To start off the time iteration scheme we will also need a specification of the initial conditions:

$$\mathbf{u}(\mathbf{x}, t = 0) = \mathbf{u}_0(\mathbf{x}) \quad \mathbf{x} \in \Omega, \tag{2.19f}$$

$$\dot{\mathbf{u}}(\mathbf{x}, t = 0) = \mathbf{v}_0(\mathbf{x}) \quad \mathbf{x} \in \Omega, \tag{2.19g}$$

$$c(\mathbf{x}, t = 0) = c_0(\mathbf{x}) \quad \mathbf{x} \in \Omega, \tag{2.19h}$$

where any pre-existing cracks may either be modelled by $c_0(\mathbf{x})$ or directly in the geometry.

2.6. Weak form formulation

To obtain the weak form formulation of our problem we multiply the equations with arbitrary test functions $\mathbf{w}$ and q respectively, both subject to certain integrability conditions, then integrating over the domain of interest. We also perform partial integrations, and we arrive at two coupled equations. The weak form formulation of eq. (2.19a) amounts to:

$$\rho \int_\Omega \ddot{\mathbf{u}} \cdot \mathbf{w} \, \mathrm{d}\mathbf{x} + \int_\Omega \nabla \mathbf{w} : \left(\boldsymbol{\sigma}^T \right) \, \mathrm{d}\mathbf{x} = \int_{\partial\Omega_h} \mathbf{h} \cdot \mathbf{w} \, \mathrm{d}\mathbf{s}, \tag{2.20a}$$

where "$\cdot$" is the usual dot product while ":" is the double dot product, when taken between two second order tensors A and B it is defined as $A : B = \sum_i \sum_j A_{ij} B_{ji}$.

Balance of angular momentum dictates that $\boldsymbol{\sigma}^T = \boldsymbol{\sigma}$, so we may rewrite the second integral of eq. (2.20a) in terms of the symmetric nabla-operator

if so preferred. The weak form formulation of eq. (2.19b) is:

$$\int_\Omega \left(\frac{4e(1-k)\mathcal{H}}{\mathcal{G}_c} + 1 \right) cq \, \mathrm{d}\mathbf{x} + 4e^2 \int_\Omega \nabla c \cdot \nabla q \, \mathrm{d}\mathbf{x} = \int_\Omega q \, \mathrm{d}\mathbf{x}. \tag{2.20b}$$

The boundary integral stemming from the partial integration is zeroed out by the Neumann boundary condition, given in equation (2.19e) and we say that this boundary condition is enforced weakly.

2.7. Reformulation for the linear case by Voigt notation

Unfortunately equation (2.20a) will in general be non-linear, due to the splitting of the elastic energy in a compressive and tensile part. However, whenever there is no crack present in the crack phase field ($c = 1$ everywhere) we still enjoy linearity, and we may use this to do benchmarking on the time integrators and so on. The following reformulation of the problem is also used whenever one opt out of the damage field approach and instead tackle the crack tracking problem directly, thus avoiding non-linearity in the displacement field. To exploit our software maximally we wish to tailor our problem description such that it eventually[1] can be stated as the general matrix-equation $\mathbf{Au} = \mathbf{b}$, $\mathbf{A}$ being a matrix and $\mathbf{u}$ and $\mathbf{b}$ vectors. The problem we face is the presence of the double dot product, indicating an expression with double summation signs whereas the matrix equation only allows for a single summation sign. A workaround is found by introducing Voigt notation for the strains $\boldsymbol{\epsilon}$ and stresses $\boldsymbol{\sigma}$ both second order tensors in the usual notation, in two spatial dimensions they are:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix}, \tag{2.21}$$

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{xx} & \epsilon_{xy} \\ \epsilon_{yx} & \epsilon_{yy} \end{bmatrix}. \tag{2.22}$$

[1] That is; after the introduction of a suitable basis and a choice of a numerical integration procedure as well as a time integration method, then finally performing a Galerkin projection.

Instead we will use the Voigt notation vector form:

$$\tilde{\boldsymbol{\sigma}} := \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}, \tag{2.23}$$

$$\tilde{\boldsymbol{\epsilon}} := \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ 2\epsilon_{xy} \end{bmatrix}. \tag{2.24}$$

Notice that the strain is defined differently (any shear strains is scaled by two). This ensures that $\boldsymbol{\epsilon} : \boldsymbol{\sigma} = \tilde{\boldsymbol{\epsilon}} \cdot \tilde{\boldsymbol{\sigma}}$ (as $\boldsymbol{\epsilon}$ and $\boldsymbol{\sigma}$ are both symmetric due to the definition of strain and balance of angular momentum respectively).

Now we may reformulate equation (2.20a) step by step:

$$\rho \int_\Omega \ddot{\mathbf{u}} \cdot \mathbf{w} \, \mathrm{d}\mathbf{x} + \int_\Omega \nabla_s \mathbf{w} : \boldsymbol{\sigma}(\boldsymbol{\epsilon}(\mathbf{u})) \, \mathrm{d}\mathbf{x} = \int_{\partial\Omega_h} \mathbf{h} \cdot \mathbf{w} \, \mathrm{ds}, \tag{2.25}$$

$$\rho \int_\Omega \ddot{\mathbf{u}} \cdot \mathbf{w} \, \mathrm{d}\mathbf{x} + \int_\Omega \boldsymbol{\epsilon}(\mathbf{w}) : \boldsymbol{\sigma}(\boldsymbol{\epsilon}(\mathbf{u})) \, \mathrm{d}\mathbf{x} = \int_{\partial\Omega_h} \mathbf{h} \cdot \mathbf{w} \, \mathrm{ds}, \tag{2.26}$$

$$\rho \int_\Omega \ddot{\mathbf{u}} \cdot \mathbf{w} \, \mathrm{d}\mathbf{x} + \int_\Omega \tilde{\boldsymbol{\epsilon}}(\mathbf{w}) \cdot \tilde{\boldsymbol{\sigma}}(\tilde{\boldsymbol{\epsilon}}(\mathbf{u})) \, \mathrm{d}\mathbf{x} = \int_{\partial\Omega_h} \mathbf{h} \cdot \mathbf{w} \, \mathrm{ds}, \tag{2.27}$$

$$\rho \int_\Omega \ddot{\mathbf{u}} \cdot \mathbf{w} \, \mathrm{d}\mathbf{x} + \int_\Omega \mathbb{D}\mathbf{w} \cdot \mathbb{C}\mathbb{D}\mathbf{u} \, \mathrm{d}\mathbf{x} = \int_{\partial\Omega_h} \mathbf{h} \cdot \mathbf{w} \, \mathrm{ds}. \tag{2.28}$$

where by eq. (2.2) and (2.5) we have that

$$\mathbb{D} := \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix}, \tag{2.29}$$

$$\mathbb{C} := \begin{bmatrix} 2\mu + \lambda & \lambda & 0 \\ \lambda & 2\mu + \lambda & 0 \\ 0 & 0 & \mu \end{bmatrix}. \tag{2.30}$$

3. A computational model

3.1. Numerical model for the crack phase-field

As seen earlier the weak form formulation of the governing equation of the crack is

$$\int_\Omega \left(\frac{4e(1-k)\mathcal{H}}{\mathcal{G}_c} + 1 \right) cq \, \mathrm{d}\mathbf{x} + 4e^2 \int_\Omega \nabla c \cdot \nabla q \, \mathrm{d}\mathbf{x} = \int_\Omega q \, \mathrm{d}\mathbf{x}, \quad \forall q \in H^1(\Omega), \tag{3.1}$$

where $H^1(\Omega) = \{q | \int_\Omega q^2 dA, \int_\Omega q_x^2 dA, \int_\Omega q_y^2 dA < \infty\}$. The only unknown at this point of calculation will be the crack density function $c(\mathbf{x})$ as we assume the history function $\mathcal{H}(t, \mathbf{x})$ to be known. In the usual fashion (within the finite element community) we can write this equation on the form $d(c, q) = z(q)$, where $d(\cdot, \cdot)$ is seen to be a bilinear form while $z(\cdot)$ is linear. We let X_h be the space spanned by our choice of basis functions. To determine the numerical solution, $c_h \in X_h = \mathrm{span}(R_{\mathbf{1}}, ..., R_{\mathbf{n_{np}}})$, we perform a spatial Galerkin projection. This is akin to asking that

$$d(c_h, q) = z(q), \forall q \in X_h. \tag{3.2}$$

By expanding c_h in terms of our basis functions and using the bi-linearity of $d(\cdot, \cdot)$ we get

$$\sum_{A=1}^{n_{np}} c_{\mathbf{A}} d(R_{\mathbf{A}}, q) = z(q), \quad \forall q \in X_h, \tag{3.3}$$

where $c_{\mathbf{A}}$ is the control variables that describes the numerical solution such that $c_h = \sum_{\mathbf{A}=1}^{n_{np}} c_{\mathbf{A}} R_{\mathbf{A}}(\vec{\xi})$. $R_{\mathbf{A}}(\vec{\xi})$ is just the basis function numbered A while n_{np} is the total number of basis functions and $\{c_{\mathbf{A}}\}$ are the control variables that we shall solve for. Since $q \in X_h$ is arbitrary, $z(\cdot)$ is linear

and $d(\cdot,\cdot)$ is bi-linear this is equivalent to

$$\sum_{A=1}^{n_{np}} c_{\mathbf{A}} d(R_{\mathbf{A}}, R_{\mathbf{B}}) = z(R_{\mathbf{B}}), \quad B = 1, ..., n_{np}. \tag{3.4}$$

This can again be written in matrix form

$$\mathbf{Dc} = \mathbf{z}. \tag{3.5}$$

where entry (i, j) of matrix $\mathbf{D}$ take on the value $d(R_{\mathbf{i}}, R_{\mathbf{j}})$, while entry i of vector $\mathbf{z}$ is just $z(R_{\mathbf{i}})$. Vector $\mathbf{c}$ is our solution vector, each entry i corresponds to the control variable $c_{\mathbf{i}}$.

We recall that the Neumann boundary condition, equation (2.19e), is enforced weakly through equation (3.1), so we need not make any further steps to enforce this particular boundary condition.

Notice that in the event that the history field is not updated, the crack phase field will remain unchanged, so we may save ourselves for the calculation.

3.2. Numerical model for the non-linear elastodynamics problem

The spatial part of the solution will be found by the isogeometric finite element method. We continue from the weak form formulation

$$\rho \int_{\Omega} \ddot{\mathbf{u}} \cdot \mathbf{w} \, \mathrm{d}\mathbf{x} + \int_{\Omega} \tilde{\boldsymbol{\epsilon}}(\mathbf{w}) \cdot \tilde{\boldsymbol{\sigma}}(\tilde{\boldsymbol{\epsilon}}(\mathbf{u})) \, \mathrm{d}\mathbf{x} = \int_{\partial\Omega_h} \mathbf{h} \cdot \mathbf{w} \, \mathrm{d}\mathbf{s}, \;\; \forall \mathbf{w} \in (H^1(\Omega))^2. \tag{3.6}$$

This is can be written in the usual form $m(\ddot{\mathbf{u}}, \mathbf{w}) + a(\mathbf{u}, \mathbf{w}) = l(\mathbf{w})$. We see that both $a(\mathbf{u}, \cdot)$ and $l(\cdot)$ are linear forms while $m(\cdot, \cdot)$ is bilinear. We let Y_h be the (spatial) space spanned by our choice of basis functions. We let $\mathbf{u}_h \in Y_h$ denote the numerical solution. As before we perform a spatial Galerkin projection to obtain an expression for $\mathbf{u}_h$:

$$m(\ddot{\mathbf{u}}_h, \mathbf{w}) + a(\mathbf{u}_h, \mathbf{w}) = l(\mathbf{w}), \;\; \forall \mathbf{w} \in Y_h. \tag{3.7}$$

We will use finite difference formulas to discretize $\mathbf{u}_h$ and $\dot{\mathbf{u}}_h$ in terms of the second derivative $\ddot{\mathbf{u}}_h$, thus obtaining a system of ordinary differential equations (ODEs), with one ODE for each time step. This is called a semi-discretization. Starting from scratch one might chose a finite differences scheme, plug it into the equation and perform Newton-Raphson iterations (as the problem is non-linear we must solve it iteratively). But we have an added bit of information we should exploit; we will know the zeroth, first and second order time derivatives of the solutions at the previous time step. This will be used to craft a better initial guess for the Newton-Raphson iteration, and it will also be used to correct the Newton-Raphson result in each iteration (this would be the multi-corrector). A much used algorithm with these features is the predictor/multi-corrector algorithm. We use the zero acceleration predictor. From here on we will drop the subscript h (indicating a numerical solution), and instead let the subscript be an iteration counter within each time step. In the description of the predictor-multi-corrector algorithm, equations (3.8a)-(3.8p), the sign "=" will be the assignment operator, not the equality sign.

Predictor:

$$q = 0 \tag{3.8a}$$

$$\mathbf{u}_q^{n+1} = \mathbf{u}^n + \Delta t \dot{\mathbf{u}}^n + \frac{(\Delta t)^2}{2}(1-2\beta)\ddot{\mathbf{u}}^n \tag{3.8b}$$

$$\dot{\mathbf{u}}_q^{n+1} = \dot{\mathbf{u}}^n + \Delta t(1-\gamma)\ddot{\mathbf{u}}^n \tag{3.8c}$$

$$\ddot{\mathbf{u}}_q^{n+1} = 0 \tag{3.8d}$$

Multi-corrector:

$$\mathbf{u}_q^{n+\alpha_f} = (1-\alpha_f)\mathbf{u}^n + \alpha_f \mathbf{u}_q^{n+1} \tag{3.8e}$$

$$\dot{\mathbf{u}}_q^{n+\alpha_f} = (1-\alpha_f)\dot{\mathbf{u}}^n + \alpha_f \dot{\mathbf{u}}_q^{n+1} \tag{3.8f}$$

$$\ddot{\mathbf{u}}_q^{n+\alpha_m} = (1-\alpha_m)\ddot{\mathbf{u}}^n + \alpha_m \ddot{\mathbf{u}}_q^{n+1} \tag{3.8g}$$

$$\mathbf{J}\Delta\ddot{\mathbf{u}} = -\mathbf{f}(\mathbf{u}_q^{n+\alpha_f}, \ddot{\mathbf{u}}_q^{n+\alpha_m}) \tag{3.8h}$$

$$\mathbf{u}_{q+1}^{n+1} = \mathbf{u}_q^{n+1} + \beta(\Delta t)^2 \Delta\ddot{\mathbf{u}} \tag{3.8i}$$

$$\dot{\mathbf{u}}_{q+1}^{n+1} = \dot{\mathbf{u}}_q^{n+1} + \gamma \Delta t \Delta\ddot{\mathbf{u}} \tag{3.8j}$$

$$\ddot{\mathbf{u}}_{q+1}^{n+1} = \ddot{\mathbf{u}}_q^{n+1} + \Delta\ddot{\mathbf{u}} \tag{3.8k}$$

$$q = q + 1 \tag{3.8l}$$

Equation (3.8h) is a single Newton-Raphson iteration where $\mathbf{f}(\cdot,\cdot) = \mathbf{0}$ are the equations for the displacement field and $\mathbf{J}$ its Jacobian, the definition will be given in equation (3.20). The multi-corrector stage can be iterated as many times as needed for convergence. Note however that before we repeat the multi-corrector we should update the history field $\mathcal{H}$ with the newest solution and solve for the crack phase field. After we have reached convergence (and updated the history and crack phase fields) we perform the assignments

$$\mathbf{u}^{n+1} = \mathbf{u}_q^{n+1} \tag{3.8m}$$

$$\dot{\mathbf{u}}^{n+1} = \dot{\mathbf{u}}_q^{n+1} \tag{3.8n}$$

$$\ddot{\mathbf{u}}^{n+1} = \ddot{\mathbf{u}}_q^{n+1} \tag{3.8o}$$

$$n = n + 1 \tag{3.8p}$$

and return to the Predictor if we wish to advance another time step.

As the predictor/multi-corrector algorithm above suggests we will evaluate the displacement field at the "predicted solutions" $\mathbf{u}_q^{n+\alpha_f}$ and $\ddot{\mathbf{u}}_q^{n+\alpha_m}$.

$$m(\ddot{\mathbf{u}}_q^{n+\alpha_m}, \mathbf{w}) + a(\mathbf{u}_q^{n+\alpha_f}, \mathbf{w}) = l(\mathbf{w}), \quad \forall \mathbf{w} \in Y_h. \tag{3.9}$$

The test function $\mathbf{w} \in Y_h$ is arbitrary and may be expressed as a linear combination of the basis functions $\mathbf{R_B}, B = 1, ..., n_{np}$. This combined with the fact that functionals a, m and l are linear in $\mathbf{w}$ reduce our problem to a finite set of (non-linear) equations

$$m(\ddot{\mathbf{u}}_q^{n+\alpha_m}, \mathbf{R_B}) + a(\mathbf{u}_q^{n+\alpha_f}, \mathbf{R_B}) = l(\mathbf{R_B}), \quad B = 1, ..., n_{np}. \tag{3.10}$$

We may express $\mathbf{u}_q^{n+\alpha_f}$ and $\ddot{\mathbf{u}}_q^{n+\alpha_m}$ in terms of their respective corresponding control variables[1] $\tilde{u}_{\mathbf{A}}^{n+\alpha_m}$ and $\ddot{\tilde{u}}_{\mathbf{A}}^{n+\alpha_m}$ and basis functions, $\mathbf{R_A}$;

$$\mathbf{u}_q^{n+\alpha_f} = \sum_{A=1}^{n_{np}} \tilde{u}_{\mathbf{A}}^{n+\alpha_f} \mathbf{R_A}, \tag{3.11}$$

$$\ddot{\mathbf{u}}_q^{n+\alpha_m} = \sum_{A=1}^{n_{np}} \ddot{\tilde{u}}_{\mathbf{A}}^{n+\alpha_m} \mathbf{R_A}. \tag{3.12}$$

[1] We suppress the dependency on q for the control variables for enhanced readability. Recall that q is just counting how many passes there have been at the multi-corrector stage within the current time step.

By substituting this and at the same time, exploiting the bi-linearity of $m(\cdot,\cdot)$ we get

$$\sum_{A=1}^{n_{np}} \left(\ddot{\tilde{u}}_{\mathbf{A}}^{n+1} m(\mathbf{R_A}, \mathbf{R_B}) \right) + a\left(\left[\sum_{A=1}^{n_{np}} \tilde{u}_{\mathbf{A}}^{n+1} \mathbf{R_A} \right], \mathbf{R_B} \right) = l(\mathbf{R_B}), \quad B = 1, ..., n_{np}. \tag{3.13}$$

We use this to define the vector-valued function $\mathbf{f}(\cdot,\cdot)$ by its entries:

$$\begin{aligned} &(\mathbf{f}(\mathbf{u}_q^{n+\alpha_f}, \ddot{\mathbf{u}}_q^{n+\alpha_m}))_B := \\ &\sum_{A=1}^{n_{np}} \left(\ddot{\tilde{u}}_{\mathbf{A}}^{n+1} m(\mathbf{R_A}, \mathbf{R_B}) \right) + a\left(\left[\sum_{A=1}^{n_{np}} \tilde{u}_{\mathbf{A}}^{n+1} \mathbf{R_A} \right], \mathbf{R_B} \right) - l(\mathbf{R_B}), \\ &B = 1, ..., n_{np}. \end{aligned} \tag{3.14}$$

Resulting is a non-linear multivariate system of equations;

$$\mathbf{f}(\mathbf{u}_q^{n+\alpha_f}, \ddot{\mathbf{u}}_q^{n+\alpha_m}) = \mathbf{0}. \tag{3.15}$$

To solve eq. (3.15) for the control variables (numerically) we will use the Newton-Raphson method. For the convergence criteria it suffices to use the Euclidean norm on the evaluated function $\mathbf{f}(\mathbf{u}_q^{n+\alpha_f}, \ddot{\mathbf{u}}_q^{n+\alpha_m})$, as each entry entails integration over the spatial domain Ω. The Newton-Raphson method calls for a procedure to calculate the Jacobian, $\mathbf{J}$ of the system. Our chosen algorithm demand that we develop the Jacobian with respect to the temporal second derivative, so that we may solve for the change in acceleration rather than the change in displacement. (This technicality only yields differing constants, the main structures of the Jacobian would remain the same if we developed it with respect to the displacement instead.)

We wish to obtain an expression for $\mathbf{u}_q^{n+\alpha_f}$ where the dependency on $\ddot{\mathbf{u}}_q^{n+1}$ is explicit. By taking into account both of the equations (3.8d) and (3.8k) we realize that we may write the following equivalent expression for equation (3.8i):

$$\mathbf{u}_{q+1}^{n+1} = \mathbf{u}_q^{n+1} + \beta(\Delta t)^2 \ddot{\mathbf{u}}_{q+1}^{n+1}. \tag{3.16}$$

Inserting this into equation (3.8e) we get the desired expression for $\mathbf{u}_q^{n+\alpha_f}$:

$$\mathbf{u}_q^{n+\alpha_f} = (1-\alpha_f)\mathbf{u}^n + \alpha_f \mathbf{u}_q^{n+1} \tag{3.17}$$

$$= (1-\alpha_f)\mathbf{u}^n + \alpha_f \mathbf{u}_{q-1}^{n+1} + \alpha_f \beta (\Delta t)^2 \ddot{\mathbf{u}}_q^{n+1}. \tag{3.18}$$

We already have an expression for $\ddot{\mathbf{u}}_q^{n+\alpha_m}$ where the dependency on $\ddot{\mathbf{u}}_q^{n+1}$ is explicit in equation (3.8g) but it is repeated here for convenience:

$$\ddot{\mathbf{u}}_q^{n+\alpha_m} = (1-\alpha_m)\ddot{\mathbf{u}}^n + \alpha_m \ddot{\mathbf{u}}_q^{n+1}. \tag{3.19}$$

We will begin to develop the Jacobian:

$$\begin{aligned}
(\mathbf{J})_{ij} &:= \frac{\partial}{\partial \ddot{\tilde{u}}_{\mathbf{i}}^{n+1}} (\mathbf{f}(\mathbf{u}_q^{n+\alpha_f}, \ddot{\mathbf{u}}_q^{n+\alpha_m}))_j \\
&= \frac{\partial}{\partial \ddot{\tilde{u}}_{\mathbf{i}}^{n+1}} \left[m(\sum_{A=1}^{n_{np}} \ddot{\tilde{u}}_{\mathbf{A}}^{n+\alpha_m} \mathbf{R}_{\mathbf{A}}, \mathbf{R}_{\mathbf{j}}) \right] + \frac{\partial}{\partial \ddot{\tilde{u}}_{\mathbf{i}}} \left[a\left(\mathbf{u}_q^{n+\alpha_f}, \mathbf{R}_{\mathbf{j}}\right) \right] \\
&= \alpha_m \frac{\partial}{\partial \ddot{\tilde{u}}_{\mathbf{i}}^{n+1}} \left[\sum_{A=1}^{n_{np}} \ddot{\tilde{u}}_{\mathbf{A}}^{\alpha} \cdot m(\mathbf{R}_{\mathbf{A}}, \mathbf{R}_{\mathbf{j}}) \right] \\
&\qquad + \frac{\partial}{\partial \ddot{\tilde{u}}_{\mathbf{i}}^{n+1}} \int_\Omega \boldsymbol{\epsilon}(\mathbf{R}_{\mathbf{j}}) : \boldsymbol{\sigma}\left(\boldsymbol{\epsilon}\left(\mathbf{u}_q^{n+\alpha_f}\right)\right) \, \mathrm{d}\mathbf{x} \\
&= \alpha_m \cdot m(\mathbf{R}_{\mathbf{i}}, \mathbf{R}_{\mathbf{j}}) + \int_\Omega \boldsymbol{\epsilon}(\mathbf{R}_{\mathbf{j}}) : \Big(\underbrace{\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}}}_{\text{4th order tensor}} : \overbrace{\frac{\partial \boldsymbol{\epsilon}}{\partial \ddot{\tilde{u}}_{\mathbf{i}}^{n+1}}\Big|_{\mathbf{u}_q^{n+\alpha_f}}}^{\text{2nd order tensor}} \Big) \, \mathrm{d}\mathbf{x} \\
&= \alpha_m \cdot m(\mathbf{R}_{\mathbf{i}}, \mathbf{R}_{\mathbf{j}}) \\
&\qquad + \alpha_f \beta (\Delta t)^2 \cdot \int_\Omega \boldsymbol{\epsilon}(\mathbf{R}_{\mathbf{j}}) : \Big(\underbrace{\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}}}_{\text{4th order tensor}} : \overbrace{\frac{\partial \boldsymbol{\epsilon}}{\partial \ddot{\tilde{u}}_{\mathbf{i}}^{n+1}}\Big|_{\ddot{\mathbf{u}}_q^{n+1}}}^{\text{2nd order tensor}} \Big) \, \mathrm{d}\mathbf{x},
\end{aligned} \tag{3.20}$$

where the double dot product between a 4^{th} order tensor $\mathbb{A}$ and a 2^{nd} order tensor $\mathbf{A}$ is defined the following way: $(\mathbb{A} : \mathbf{A})_{ij} = \sum_k \sum_j \mathbb{A}_{ijkl} \mathbf{A}_{kl}$ and $(\mathbf{A} : \mathbb{A})_{ij} = \sum_k \sum_j \mathbf{A}_{kl} \mathbb{A}_{klij}$. We already know that the mass matrix

stemming from the first term will be a sparse matrix since the support of the basis functions is limited. Furthermore we can see that $\boldsymbol{\epsilon}(\mathbf{R_j})$ may be non-zero only where $\mathbf{R_j}$ has support. Similarly we will show that $\partial\boldsymbol{\epsilon}/\partial\tilde{u}_{\mathbf{i}}^{n+1}$ may be non-zero only where $\mathbf{R_i}$ has support, refer to equation (3.21). From this we may infer that the Jacobian $\mathbf{J}$ will be a sparse matrix! (A very useful feature as it is.)

We will first calculate $\partial\boldsymbol{\epsilon}/\partial\tilde{u}_{\mathbf{k}}^{n+1}$. To avoid confusion recall that the physical interpretation of indicing is not common across all variables and notice that k is fixed. Vector $\mathbf{e_i}$ is the i-th unit vector.

$$\begin{aligned}\left(\left.\frac{\partial\boldsymbol{\epsilon}}{\partial\tilde{u}_{\mathbf{k}}^{n+1}}\right|_{\ddot{\mathbf{u}}_q^{n+1}}\right)_{ij} &= \frac{1}{2}\frac{\partial}{\partial\tilde{u}_{\mathbf{k}}^{n+1}}\left(\frac{\partial(\mathbf{u}_q^{n+1})_j}{\partial x_i}+\frac{\partial(\mathbf{u}_q^{n+1})_i}{\partial x_j}\right)\\ &= \frac{1}{2}\frac{\partial}{\partial\tilde{u}_{\mathbf{k}}^{n+1}}\left(\mathbf{e_j}\cdot\sum_{A=1}^{n_{np}}\tilde{u}_{\mathbf{A}}^{n+1}\frac{\partial\mathbf{R_A}}{\partial x_i}+\mathbf{e_i}\cdot\sum_{A=1}^{n_{np}}\tilde{u}_{\mathbf{A}}^{n+1}\frac{\partial\mathbf{R_A}}{\partial x_j}\right)\\ &= \frac{1}{2}\left(\mathbf{e_j}\cdot\frac{\partial\mathbf{R_k}}{\partial x_i}+\mathbf{e_i}\cdot\frac{\partial\mathbf{R_k}}{\partial x_j}\right).\end{aligned} \tag{3.21}$$

The tensor will of course have the same dimension and size as $\boldsymbol{\epsilon}$. Over the next two pages we will step by step obtain an expression for the fourth order tensor $\partial\boldsymbol{\sigma}/\partial\boldsymbol{\epsilon}$. We start by expanding it,

$$\begin{aligned}\frac{\partial\boldsymbol{\sigma}}{\partial\boldsymbol{\epsilon}} &= \frac{\partial}{\partial\boldsymbol{\epsilon}}\left(\left[(1-k)c^2+k\right]\frac{\partial\psi^+}{\partial\boldsymbol{\epsilon}}+\frac{\partial\psi^-}{\partial\boldsymbol{\epsilon}}\right)\\ &= \left[(1-k)c^2+k\right]\frac{\partial}{\partial\boldsymbol{\epsilon}}\left(\lambda\langle\mathrm{tr}(\boldsymbol{\epsilon})\rangle_+\mathbf{I}+2\mu\boldsymbol{\epsilon}_+\right)+\frac{\partial}{\partial\boldsymbol{\epsilon}}\left(\lambda\langle\mathrm{tr}(\boldsymbol{\epsilon})\rangle_-\mathbf{I}+2\mu\boldsymbol{\epsilon}_-\right).\end{aligned} \tag{3.22}$$

We will develop the individual components. Use the chain rule and obtain

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\epsilon}}\left(\langle\mathrm{tr}(\boldsymbol{\epsilon})\rangle_+\mathbf{I}\right) &= H(\mathrm{tr}(\boldsymbol{\epsilon}))\frac{\partial}{\partial\boldsymbol{\epsilon}}(\mathrm{tr}(\boldsymbol{\epsilon})\mathbf{I})\\ &= H(\mathrm{tr}(\boldsymbol{\epsilon}))\frac{\partial}{\partial\boldsymbol{\epsilon}}\begin{bmatrix}\sum_i\epsilon_{ii} & 0 & \cdots\\ 0 & \sum_i\epsilon_{ii} & \cdots\\ \vdots & \vdots & \ddots\end{bmatrix}.\end{aligned} \tag{3.23}$$

Now, looking at the elements of the fourth order tensor on the right hand side we see that we must have

$$\left(\frac{\partial}{\partial \boldsymbol{\epsilon}}\left(\mathrm{tr}(\boldsymbol{\epsilon})\mathbf{I}\right)\right)_{ijkl} := \frac{\partial\left(\mathrm{tr}(\boldsymbol{\epsilon})\mathbf{I}\right)_{ij}}{\partial \epsilon_{kl}} = \begin{cases} 0 & \text{if } i \neq j \\ 0 & \text{if } k \neq l \\ 1 & \text{if } i = j \text{ and } k = l \end{cases} \tag{3.24}$$

This tensor resembles the fourth order identity tensor $\mathbb{I}$, but it carries a few extra non-zero entries. We failed to recover its usual name/symbol in the literature, so we will just name this tensor $\mathbb{J}$ for the remainder of the text.

$$\frac{\partial}{\partial \boldsymbol{\epsilon}}\left(\langle \mathrm{tr}(\boldsymbol{\epsilon})\rangle_{+}\mathbf{I}\right) = H(\mathrm{tr}(\boldsymbol{\epsilon}))\mathbb{J}, \tag{3.25a}$$

and completely analogous we get

$$\frac{\partial}{\partial \boldsymbol{\epsilon}}\left(\langle \mathrm{tr}(\boldsymbol{\epsilon})\rangle_{-}\mathbf{I}\right) = H(-\mathrm{tr}(\boldsymbol{\epsilon}))\mathbb{J}. \tag{3.25b}$$

It still remains to calculate $\frac{\partial \boldsymbol{\epsilon}_{+}}{\partial \boldsymbol{\epsilon}} = \frac{\partial}{\partial \boldsymbol{\epsilon}}\left(\sum_a \langle \epsilon_{\mathbf{a}}\rangle_{+}\mathbf{n}_{\mathbf{a}} \otimes \mathbf{n}_{\mathbf{a}}\right)$ and its negative counterpart. First we introduce the short notation $\mathbf{M}_{\mathbf{a}} := \mathbf{n}_{\mathbf{a}} \otimes \mathbf{n}_{\mathbf{a}}$. We define the fourth order tensor $\mathbb{G}_{\mathbf{ab}}$ by its coordinates as $(\mathbb{G}_{\mathbf{ab}})_{ijkl} := (\mathbf{M}_{\mathbf{a}})_{ik}(\mathbf{M}_{\mathbf{b}})_{jl} + (\mathbf{M}_{\mathbf{a}})_{il}(\mathbf{M}_{\mathbf{b}})_{jk}$. The formula for differentiation of the separate components of the spectral decomposition can be found in [17] and reads as follows

$$\frac{\partial \epsilon_{\mathbf{a}}}{\partial \boldsymbol{\epsilon}} = \mathbf{M}_{\mathbf{a}}, \tag{3.26a}$$

$$\frac{\partial \mathbf{M}_{\mathbf{a}}}{\partial \boldsymbol{\epsilon}} = \sum_{b \neq a} \frac{1}{2(\epsilon_{\mathbf{a}} - \epsilon_{\mathbf{b}})}(\mathbb{G}_{\mathbf{ab}} + \mathbb{G}_{\mathbf{ba}}). \tag{3.26b}$$

With this in mind we will develop $\frac{\partial \boldsymbol{\epsilon}_{+}}{\partial \boldsymbol{\epsilon}}$ by its individual entries:

$$
\begin{aligned}
\left(\frac{\partial \boldsymbol{\epsilon}_+}{\partial \boldsymbol{\epsilon}}\right)_{ijkl} &= \frac{\partial (\boldsymbol{\epsilon}_+)_{ij}}{\partial \epsilon_{kl}} \\
&= \frac{\partial}{\partial \epsilon_{kl}} \left(\sum_a \langle \epsilon_{\mathbf{a}} \rangle_+ (\mathbf{M}_{\mathbf{a}})_{ij} \right) \\
&= \sum_a \left[\frac{\partial \langle \epsilon_{\mathbf{a}} \rangle_+}{\partial \epsilon_{kl}} (\mathbf{M}_{\mathbf{a}})_{ij} + \langle \epsilon_{\mathbf{a}} \rangle_+ \frac{\partial (\mathbf{M}_{\mathbf{a}})_{ij}}{\partial \epsilon_{kl}} \right] \\
&= \sum_a H(\epsilon_{\mathbf{a}}) \cdot (\mathbf{M}_{\mathbf{a}})_{kl} (\mathbf{M}_{\mathbf{a}})_{ij} \\
&\qquad + \sum_a \sum_{b \neq a} \frac{\langle \epsilon_{\mathbf{a}} \rangle_+}{2(\epsilon_{\mathbf{a}} - \epsilon_{\mathbf{b}})} \left[(\mathbb{G}_{\mathbf{ab}})_{ijkl} + (\mathbb{G}_{\mathbf{ba}})_{ijkl} \right].
\end{aligned} \tag{3.27}
$$

The result for $\frac{\partial \boldsymbol{\epsilon}_-}{\partial \boldsymbol{\epsilon}}$ is obtained similarly. Let us define the fourth order tensor $\mathbb{Q}_{\mathbf{a}}$ by its coordinates: $(\mathbb{Q}_{\mathbf{a}})_{ijkl} := (\mathbf{M}_{\mathbf{a}})_{kl} (\mathbf{M}_{\mathbf{a}})_{ij}$. We are now ready to finish the formula we sought after:

$$
\begin{aligned}
\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} &= \Big[(1-k)c^2 + k \Big] \left[\lambda \frac{\partial}{\partial \boldsymbol{\epsilon}} \left(\langle \mathrm{tr}(\boldsymbol{\epsilon}) \rangle_+ \mathbf{I} \right) + 2\mu \frac{\partial \boldsymbol{\epsilon}_+}{\partial \boldsymbol{\epsilon}} \right] \\
&\quad + \left[\lambda \frac{\partial}{\partial \boldsymbol{\epsilon}} \left(\langle \mathrm{tr}(\boldsymbol{\epsilon}) \rangle_- \mathbf{I} \right) + 2\mu \frac{\partial \boldsymbol{\epsilon}_-}{\partial \boldsymbol{\epsilon}} \right] \\
&= \Big[(1-k)c^2 + k \Big] \lambda H(\mathrm{tr}(\boldsymbol{\epsilon})) \mathbb{J} \\
&\quad + \Big[(1-k)c^2 + k \Big] \left[2\mu \sum_a H(\epsilon_{\mathbf{a}}) \mathbb{Q}_{\mathbf{a}} \right] \\
&\quad + \Big[(1-k)c^2 + k \Big] \left[2\mu \sum_a \sum_{b \neq a} \frac{\langle \epsilon_{\mathbf{a}} \rangle_+}{2(\epsilon_{\mathbf{a}} - \epsilon_{\mathbf{b}})} \left[\mathbb{G}_{\mathbf{ab}} + \mathbb{G}_{\mathbf{ba}} \right] \right] \\
&\quad + \lambda H(-\mathrm{tr}(\boldsymbol{\epsilon})) \mathbb{J} \\
&\quad + 2\mu \sum_a \left(H(-\epsilon_{\mathbf{a}}) \mathbb{Q}_{\mathbf{a}} + \sum_{b \neq a} \frac{\langle \epsilon_{\mathbf{a}} \rangle_-}{2(\epsilon_{\mathbf{a}} - \epsilon_{\mathbf{b}})} \left[\mathbb{G}_{\mathbf{ab}} + \mathbb{G}_{\mathbf{ba}} \right] \right).
\end{aligned} \tag{3.28}
$$

Although this fourth order tensor looks rather intimidating it is small enough that we can easily work with it, in three dimensions it has $3^4 = 81$ entries and in two spatial dimensions it has no more than $2^4 = 16$ entries.

We have now completed all the formulas necessary to assemble the Jacobian, $\mathbf{J}$, as given in equation (3.20).

3.3. Numerical model for the history field

As mentioned in section 2.4 the idea behind the algorithmic decoupling of the coupled equations (2.15a) and (2.15b) bases on an approximation of the current history field $\mathcal{H}_{n+1}$ on the displacement at the previous time step t_n [16].

We will use the following scheme:

$$\mathcal{H}^h(t_{n+1}, \mathbf{x}) = \begin{cases} \mathcal{H}^h(t_n, \mathbf{x}) & \text{if } \mathcal{H}^h(t_n, \mathbf{x}) \geq \psi^+(t_n, \mathbf{x}) \\ \psi^+(t_n, \mathbf{x}) & \text{if } \mathcal{H}^h(t_n, \mathbf{x}) \leq \psi^+(t_n, \mathbf{x}) \end{cases} \tag{3.29}$$

In practice this strategy means that we keep a list with a corresponding history field value for each evaluation point used in the numerical integration procedure of our choice (in our case the quadrature points). This approach is obviously very cumbersome as it doesn't immediately allow for local refinement at each new time step; instead we must refine a priori. If we were to refine at each time step we would either need a procedure for solution transfer or another method to calculate the history field all together.

3.3.1. Implementing the pre-existing crack in the initial conditions

As mentioned earlier we may model a pre-existing crack indirectly via the initial condition $c(\mathbf{x}, t_0) = c_0(\mathbf{x})$. The alternative is to model it directly with the geometry: In regards to the need for local refinement along all of the crack as described in the crack phase field to enhance the sharpness of the crack phase field this is a very good option.

To make sure that the induced crack doesn't disappear after the first time step we must include it in the history field. As the crack phase field can be generated directly from the history field, we might perform this task backwards and start with an initial history field and use this to generate the initial condition for the crack phase field. Let C be the highest value for the initial crack in the crack phase field, we choose $C = 10^{-3}$. Now, let the metric $d(\mathbf{x}, \Gamma)$ measure the distance between the chosen crack topology

$\Gamma(\mathbf{x})$ and the point $\mathbf{x}$. By [3] we may then use the following initial history field

$$\mathcal{H}^h(t_0, \mathbf{x}) = \begin{cases} \frac{\mathcal{G}_c}{4e}\left(\frac{1}{C} - 1\right)\left(1 - \frac{d(\mathbf{x},\Gamma)}{e}\right) & \text{if } d(\mathbf{x}, \Gamma) \leq e, \\ 0 & \text{otherwise.} \end{cases} \tag{3.30}$$

The initial condition for the crack phase field may now be obtained by following the procedure given in section 3.1.

4. Numerical example

4.1. Crack branching

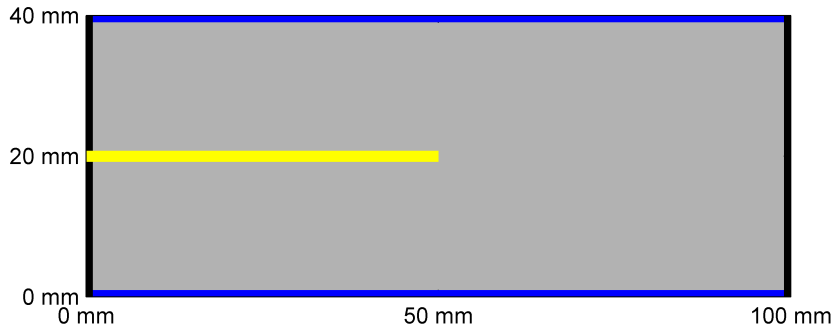


Figure 4.1.: Domain of interest: a 40x100 mm plate. Along the top and bottom (blue) boundary there is applied a traction force in the outward facing direction. Along the left and right (black) boundary there is a zero traction condition. The initial conditions will include a crack along the yellow line, induced by the initial history field.

4.1.1. Set-up: Parameters, initial and boundary conditions

The problem parameters is set equal to the suggested values in [3] to ease the comparison of results. Material density is set to $\rho = 2450$ kg/m^3 and critical fracture energy density is set to $\mathcal{G}_c = 3$ J/m^2. Furthermore we set Young's modulus to $E = 32$ GPa and Poissons's ratio to $\nu = 0.2$.

Converted to Lamé constants in the case of plane strain this equals: $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} = \frac{6.4}{0.72} \cdot 10^9$ Pa $\approx 8.89 \cdot 10^9$ Pa and $\mu = \frac{E}{2(1+\nu)} = \frac{32}{2.4} \cdot 10^9$ Pa $\approx 1.33 \cdot 10^{10}$ Pa.

The domain is a rectangle with width 100 mm and height 40 mm. Initially there is a crack induced by the prescribed history field, running horizontally from the left-hand end of the rectangle to the midpoint of the rectangle, see figure (4.1). The initial displacements and its first derivative is chosen to be zero everywhere. As for the boundary conditions there is a zero-traction condition along the left- and right-hand side (this amounts to $\mathbf{h} = \mathbf{0}$). An outward pointing traction load of $\sigma = 1$ MPa is applied at the top and bottom edges, $\mathbf{h} = [0, 10^6]^T$. (This implies that the whole boundary is governed by Neumann boundary conditions.)

The parameters for the time stepping scheme is chosen by the generalized-α method, again in line with [3]. The spectral radius is set to $\rho_\infty = 0.5$. The parameters are prescribed as follows:

$$\alpha_f = \frac{1}{\rho_\infty + 1}, \tag{4.1}$$

$$\alpha_m = \frac{2 - \rho_\infty}{\rho_\infty + 1}, \tag{4.2}$$

$$\beta = \frac{1}{4}(1 + \alpha_m - \alpha_f)^2, \tag{4.3}$$

$$\gamma = \frac{1}{2} + \alpha_m - \alpha_f. \tag{4.4}$$

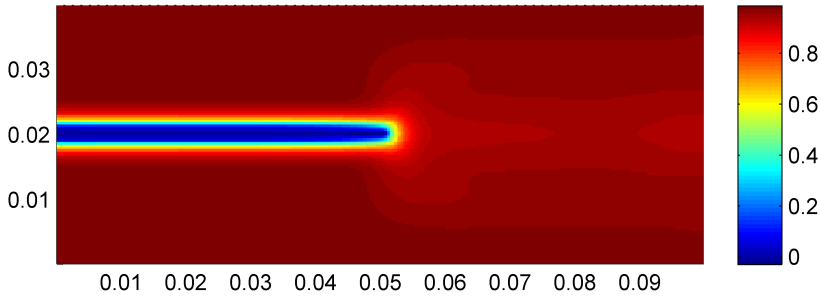
The time step is set $\Delta t = h/v_R$, where h is the mesh size parameter (the smallest element have roughly area h^2) and $v_R = 2125$ m/s is the Rayleigh wave speed given in [3].

4.1.2. Results with NURBS

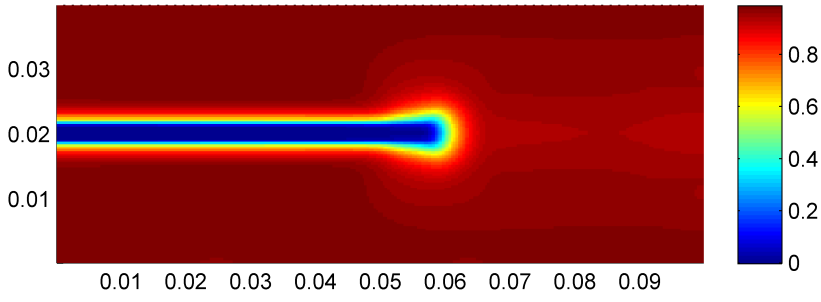
The following results were obtained using only one pass at the multi-corrector stage at each time step (performing only one Newton-Raphson iteration plus the successive corrector at each time step). Compared to the results obtained in Borden et al. [3] we observe a significant qualitative resemblance. The relative widening of the crack is due to a higher value assigned to the crack phase field smearing parameter e. Remember that we must have $e > h/2$ to capture the fracture correctly, so the sharpness

of the crack is in turn limited by the computational capacity we possess. The time it took for the crack to propagate throughout the domain is seen to be higher on our grid, this is in line with the trend seen in previous results that indicate that a finer grid will result in faster crack propagation [3].

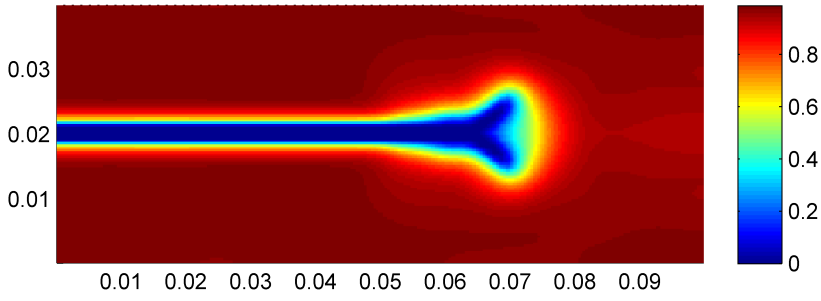
We found that we could only obtain satisfactory results on a reasonably refined grid. At coarse grids we found that the crack took on a pitchfork-shape rather than the desired Y-shape, and at even coarser grids we saw that the majority of the domain came to be either cracked or damaged material. Given that we expect substantial changes in topology over small areas this shouldn't come as a surprise; we need a certain amount of basis functions to capture the fracture. Despite this we will make a note of the fact that it is remarkable that the phase-field approach allows us to make use of NURBS (with one only patch), given that the mesh we operate on is structured much like a tensor product. The method was also tested with basis degrees two and three to satisfactory results. Figures (4.2a)-(4.2c) and (4.3a)-(4.3c) shows a time series of the propagating fracture including branching.



(a) $T = 40dt = 1.98 \cdot 10^{-5}$s.

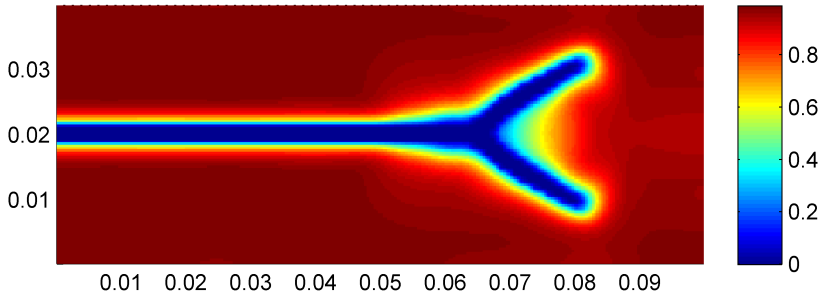


(b) $T = 80dt = 3.96 \cdot 10^{-5}$s.

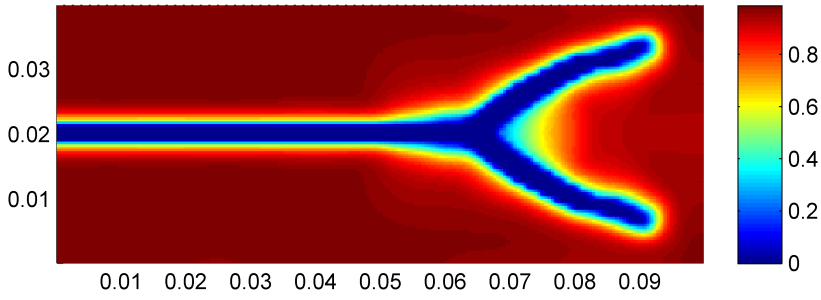


(c) $T = 120dt = 5.94 \cdot 10^{-5}$s.

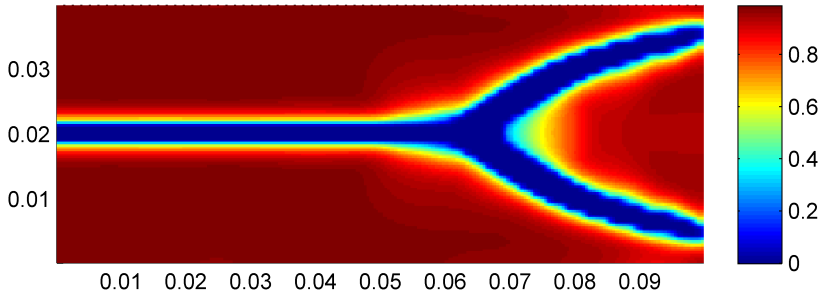
Figure 4.2.: Crack phase field for basis degrees one. With NURBS as basis and uniform elements. Degrees of freedom, $ndof = 3744$ and $h = 1.05 \cdot 10^{-3}$m while $dt = 4.95 \cdot 10^{-7}$s for all three figures above. The crack phase field smearing parameter is set to $e = 5.2 \cdot 10^{-4}$m.



(a) $T = 160dt = 7.93 \cdot 10^{-5}$s.



(b) $T = 200dt = 9.91 \cdot 10^{-5}$s.



(c) $T = 240dt = 1.19 \cdot 10^{-4}$s.

Figure 4.3.: Crack phase field for basis degrees one. With NURBS as basis and uniform elements. Degrees of freedom, $ndof = 3744$ and $h = 1.05 \cdot 10^{-3}$m while $dt = 4.95 \cdot 10^{-7}$s for all three figures above. The crack phase field smearing parameter is set to $e = 5.2 \cdot 10^{-4}$m.

4.1.3. Results with LR B-splines

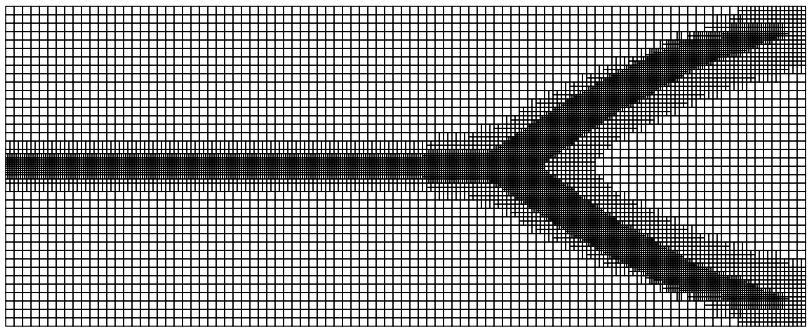
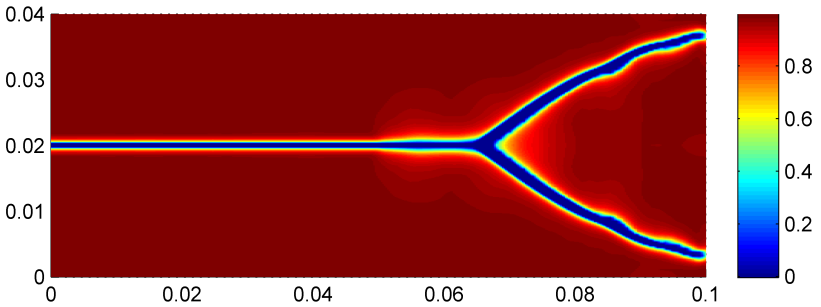
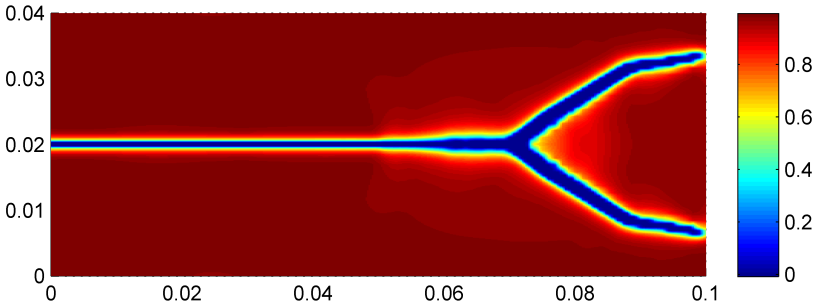


Figure 4.4.: The domain as partitioned by the LR B-spline approach with local refinement centred along the crack path. Corresponding to mesh no. 3 in table (4.1). Notice that the refinements isn't quite centred on top of each other, this is due to our greedy strategy with regard to the crack phase field smearing parameter, e, as we choose to lessen the smearing of the crack phase field at each successive refinement.

Our choice of history field without easy access to solution transfer means that we must refine a priori. To refine a priori we solve the problem on a uniform mesh, mark elements for refinement, refine and repeat as many times as we deem necessary. Although we don't access any analytical solution for comparison it is evident from similar problems that we might expect the error to dominate at the crack tip [13]. The crack tip will of course propagate along the crack path so we would want to refine this area (as we refine a priori). We should also consider the fact that both the time step, dt, and the crack phase field parameter, e, depends upon the mesh size. To be able to capture the crack topology correctly and at the same time minimize e we will therefore refine as uniformly as we manage along the complete crack path, this includes refinement along the pre-existing crack path of the initial conditions. The LR B-spline consists of B-splines but is not kept in the usual tensor product-structured mesh,



(a) Parameters as in table (4.1), particularly $e = 1.75 \cdot 10^{-4}$m



(b) Parameters as in table (4.1), with the exception that $e = 2.5 \cdot 10^{-4}$m

Figure 4.5.: Crack phase field using LR B-splines as basis. Plots of the fully formed crack for two different values of the crack phase field smearing parameter e.

instead there is allowed for individual refinement of any element. As with T-splines the local refinement will sometimes propagate into neighbouring elements beyond the original intention. To us this effect was only a minor inconvenience, hardly noticeable at all.

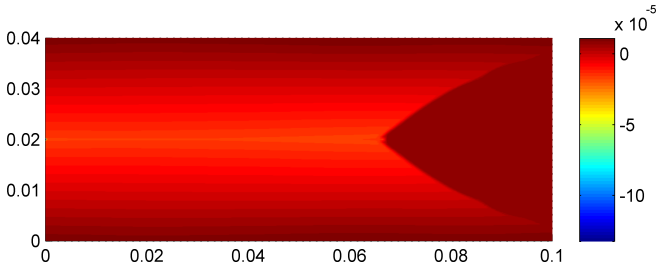
Our example features a twice refined grid, for details see table (4.1). The crack phase field is shown in figure (4.5a), while the displacement field is illustrated in figure (4.6) where we notice that the displacements depict as we would expect them to intuitively. By examining the displacements in the y-direction over the horizontal section of the crack in figure (4.6b) we observe that the crack is in actuality much sharper than one would expect

from looking at just the crack phase field alone. This is due to the fact that $c < 1$ represents damaged material, but only as c goes towards zero do we have a fully formed crack (as we chose $C = 10^{-3}$ in section 3.3.1 we assume $c \leq C$ represents a fully formed crack in practice).

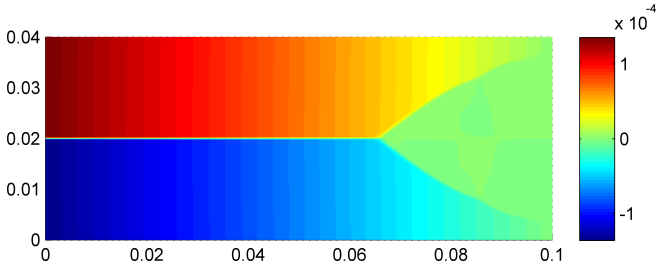
A second test was performed using all the parameters (except for those of the grid) equal to those of Borden et al. [3], see figure (4.5b). The results are similar qualitatively, in particular we see that the kink in the crack trajectory near the edge of the domain is reproduced. The crack in figure (4.5b) was also created using the mesh as depicted in figure (4.4). We notice that the crack in figure (4.5b) isn't fully contained in the twice refined area, although it is contained in the once refined area. The chosen crack phase field smearing parameter is approximately equal to half of the mesh size parameter h_{min} of the once refined mesh and therefore this doesn't cause us any problems.

No.	Deg. of freedom $ndof$	Time step $dt/[s]$	Mesh size $h_{min}/[m]$	Smearing $e/[m]$	Basis deg. p	Threshold θ
1	3744	$4.95 \cdot 10^{-7}$	$1.05 \cdot 10^{-3}$	$5.26 \cdot 10^{-3}$	1	0.5
2	6146	$2.48 \cdot 10^{-7}$	$5.26 \cdot 10^{-3}$	$2.63 \cdot 10^{-4}$	1	0.5
3	11 100	$1.24 \cdot 10^{-7}$	$2.63 \cdot 10^{-4}$	$1.75 \cdot 10^{-4}$	1	-

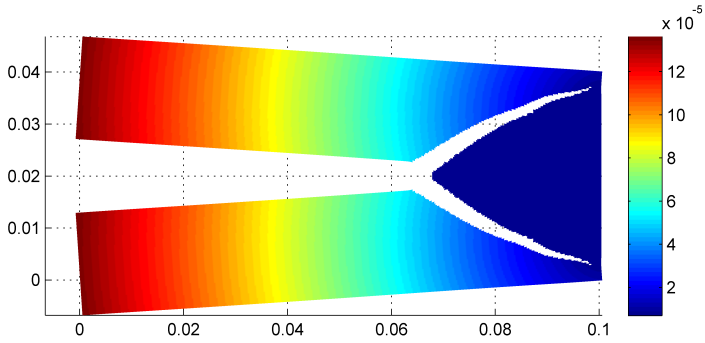
Table 4.1.: This table displays our choices for the successive refinements. The degrees of freedom is calculated for the crack phase field, for the corresponding displacement field the degrees of freedom will be twice that number. dt is the time step, h_{min} the shortest side wall of any one element, e is the model parameter of the crack phase field that controls the smearing of the phase field and the threshold value is used to decide which elements to mark for the next refinement. If the crack phase field is found to be below the threshold value in any given element, it is marked for refinement, and will be inserted a centred cross in the subsequent refinement process.



(a) Displacements in x-direction, measured in meters.



(b) Displacements in y-direction, measured in meters.



(c) The cracked plate. The displacements are magnified with a factor of 50 and the area where $c < 10^{-3}$ is taken out. It is coloured with the euclidean norm of the displacements, measured in meters (not magnified).

Figure 4.6.: Plots of the displacement field calculated on the locally refined domain. We refer to row no. 3 in table (4.1) for specifications.

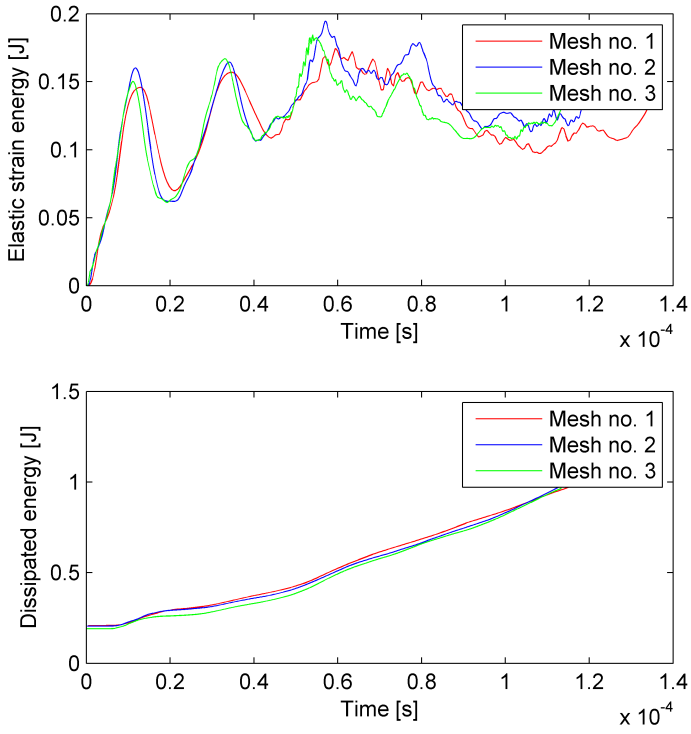


Figure 4.7.: Energy plots. We observe that the energies are slightly overestimated compared to previous results [3]. This conforms to the observation that energies are overestimated in coarser grids.

We will attempt to reproduce the energy plots of [3]. The elastic strain energy is defined as

$$\mathcal{E}_{\text{elastic}} := \int_\Omega \{[(1-k)c^2 + k]\psi^+(\boldsymbol{\epsilon}) + \psi^-(\boldsymbol{\epsilon})\}\, \mathrm{d}\mathbf{x} \tag{4.5a}$$

The dissipated energy is defined as

$$\mathcal{E}_{\text{dissipated}} := \int_\Omega G_c \left[\frac{(c-1)^2}{4e} + e\left(\frac{\partial c}{\partial \mathbf{x}} \cdot \frac{\partial c}{\partial \mathbf{x}}\right)\right] \mathrm{d}\mathbf{x} \tag{4.5b}$$

The results are plotted in figure (4.7). The shape of the plots are as expected. Notice that we don't experience convergence towards the refined grid solution, this because of the greedy strategy with respect to the model smearing parameter; by altering a model parameter (of the actual mathematical model, we do not refer to the computational model) at each step we cannot expect this. By choosing this strategy however we get a confirmation that the shape of the crack phase field is similar to that in [3]. As the very idea of a damage field is to force the smearing towards zero as the grid is refined this is would also be the reasonable approach in applications.

4.2. Remarks on the implementation

The programs used[1] to generate the results are implemented using MATLAB, but a high speed computing language such as C++ would probably be a more suitable choice. As in ordinary FEM the assembly of the matrices may be performed in parallel. We used the Parallel Computing Toolbox, specifically using the single program multiple data option (the `spmd` statement), useful because we accessed a computational server with multiple central processing units. This is what allowed us to get reasonable run times despite our choice of programming language. Another possibility would be to use a suitable graphics processing unit for the parallelization. We also obtained significant memory preservation by exploiting the sparse structure of our matrices (for this we used the built-in `spalloc` function).

The finer details of implementing isogeometric analysis is left out and those who do not wish to work them out for themselves may refer to the excellent book by Cottrell et al. that features the necessary pseudo code to get started on the implementation [6].

[1] Mesh generation, including generation of connectivity lists are provided externally.

5. Concluding remarks

In this thesis we explore a recently developed model on dynamic fracturing. The literature is scarce and no traditional benchmark cases exists as of now. To ease the comparison of results we attempted to reproduce (some of) the plots of [3].

The model is outlined with some extra detail for enhanced clarity, including the set-up of the multivariate Euler-Lagrange equations. The Jacobian of the semi-discretized system is developed analytically. It is not quite clear to us how the Jacobian have been treated previously and it might well be that a numerical approximation to calculate the fourth order tensor that is the derivative of the stresses with respect to the strains would suffice. The formula for the Jacobian makes it easy to see that it will be a sparse matrix, a very fortunate property that is exploited in the implementation to save on the memory requirements.

To tackle the problem of non-reversibility of fracturing our model is enriched with a history field describing the maximal tensile energy density. By use of a simplifying assumption on the history field we gain algorithmic decoupling of the crack and displacements. However we lack a good strategy to deal with the history field, as of now the lack of a strategy for solution transfer prohibits us from performing successive refinements during the course of one simulation, instead we must refine a priori, a very cumbersome task. While the introduction of solution transfer of some sort would mitigate the reruns required to perform local refinement, we would also need to assess the effects it would have on the solution quality.

The selected numerical solution procedure is implemented and found to work satisfactory. In particular our tests confirm that we get the expected results with both NURBS and LR B-splines. To obtain reasonable results the grid must not be too coarse, this result is intuitive given that the smearing parameter of the crack phase field is limited downwards by the mesh size. It is the first time the model is tested with LR B-Splines, and the results seem promising. We performed successful local refinement and

in particular we observed the propagation effects of the refinement to be minimal.

Appendices

A. Introduction to NURBS

A.1. Isogeometric analysis and NURBS type basis functions

When performing isogeometric analysis we may use Non-Uniform Rational B-splines (NURBS) as our basis. One great advantage is that they can represent conic sections exactly, example in figure (A.4). Furthermore the tensor-product like structure makes them supremely easy to work with. To assemble neighbour and connectivity lists have never been easier. What might seem surprising is that NURBS of higher polynomial order will be supported in more elements (also called knot spans) than those of a lower order. Fortunately the total number of non-zero basis functions at any given point will not increase compared to FEM, so the desired sparse matrix system is retained as you can see in figure (A.1). The most significant drawback of using NURBS in isogeometric analysis is the fact that it doesn't allow for true local refinement; any refinement will propagate throughout the domain.

To perform isogeometric analysis with a NURBS basis we leave behind the notion of interpolated nodes and allow for some slack in our new control points (see figure (A.3) for an example). Maybe even more confusing we redefine elements to be equal to the spans of the knots in parameter space, found in the always non-decreasing knot vectors that describe our basis: We partition the domain into elements by the grid decided by the knots in parameter space. The parameter space as decided by the knot vectors is always a line/rectangle/cube depending on the dimension of parameter space, and it is projected onto our geometry. A schematic illustration is given in figure (A.5).

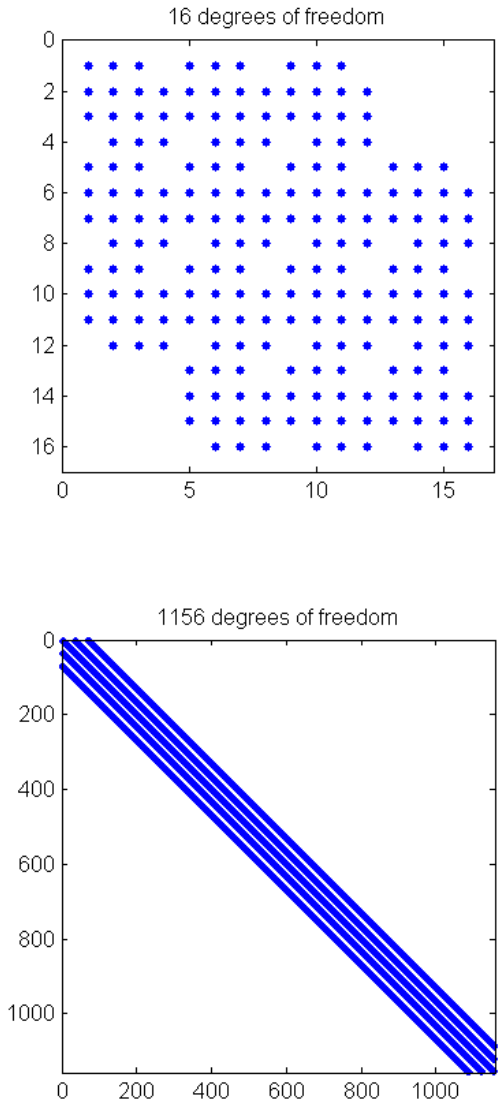


Figure A.1.: Sparse matrix $\mathbf{A}$ of the discretized system $\mathbf{Au} = \mathbf{f}$ for the Poisson equation (using isogeometric analysis). White is zero-entries, blue represent non-zero matrix entries.

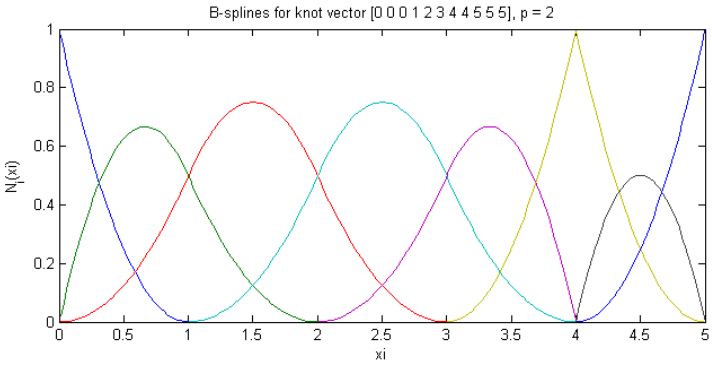


Figure A.2.: B-splines. Notice how the end-points are interpolated, and also the reduced continuity at the repeated knot $\xi = 4$. Also notice that the set of basis functions sum to one over the whole interval.

A.2.1. Knot Vectors and corresponding B-splines

Definition 1. *An open knot vector* $\Xi = [\xi_1, ..., \xi_{n+p+1}]$ *is a non-decreasing vector with elements in parameter space* $\xi \in \mathbb{R}$. *The vector has* $p + n + 1$ *entries where* p *is the polynomial order of the basis and* n *is the total number of basis functions in the* ξ*-direction. The first and last entries of the open knot vector are repeated* $p + 1$ *times.*

We will only ever work with open knot vectors (characterised by $p + 1$ repeated knots in the ends), and will therefore refer to them simply as knot vectors. Notice that the indices of the knot vector is often called index space.

The founding pillar of the NURBS-function is the B-spline, which is defined in the following way:

Definition 2 (Cox-de Boor recursion formula)**.** *B-splines are defined recursively starting at polynomial order* $p = 0$ *:*

$$N_{i,p=0}(\xi) = \begin{cases} 1 & \textit{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \textit{otherwise.} \end{cases} \tag{A.1}$$

For $p > 0$ B-splines are defined recursively by

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \tag{A.2}$$

Furthermore: Whenever the denominator in one of the terms is equals zero the term in question is defined to take on the value zero.

The B-splines of order zero will be piecewise constants, B-splines of order one will be piecewise linear functions (hat functions) etc. For each time p increases by one the support of the B-spline spans one more element (knot span). This can be seen directly of the recursion formula. An implication is that every time p is increased by one the number of basis functions is reduced by one. The B-splines are linearly independent.

Another property of the B-splines is their partition of unity, that is the sum of the B-splines at any given point within the specified geometry will always sum to one, a well known concept from FEM. Unlike the basis functions most commonly used in FEM however, B-splines always remains positive, turning into an advantage when we begin to invert matrices.

A.2.2. Derivatives

The formula for the B-spline derivative of order greater than or equal to one can similarly to the B-spline itself be written as a recursion formula [6]:

$$\begin{aligned} \frac{d^k}{d\xi^k} N_{i,p}(\xi) = & \frac{p}{\xi_{i+p} - \xi_i} \left(\frac{d^{k-1}}{d\xi^{k-1}} N_{i,p-1}(\xi) \right) \\ & - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} \left(\frac{d^{k-1}}{d\xi^{k-1}} N_{i+1,p-1}(\xi) \right) \end{aligned} \tag{A.3}$$

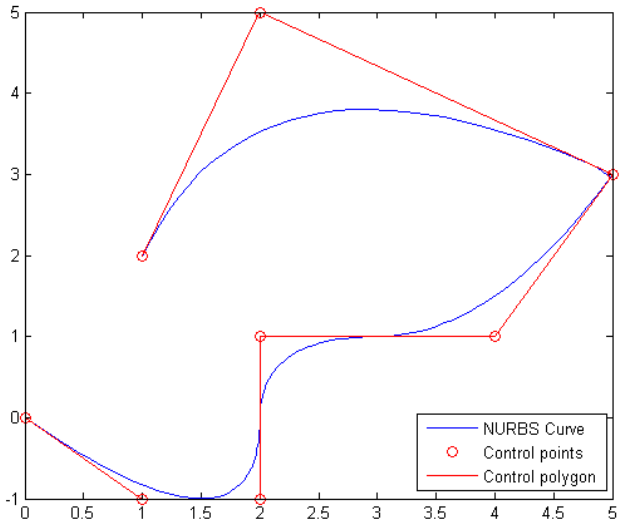


Figure A.3.: With the same knot vector as in figure (A.2) and control points where all weights are set equal to one. Because all weights are of value one it is a B-spline curve as well as a NURBS-curve.

One parametric dimension

In one parametric dimension there is assigned one control point $\mathbf{B}_i \in \mathbb{R}^d$ (arbitrary dimension d) for every basis function $i = 1, ..., n$. Every control point is in addition assigned a positive weight w_i. This weight w_i will also appear in the NURBS basis functions. (The weighs carry a nice geometrical interpretation [6], but for our purposes the stated mechanical interpretation suffices.)

Definition 3. *The NURBS basis function in one parametric dimension is defined by*

$$R_i^p(\xi) = \frac{N_{i,p}(\xi)w_i}{\sum_{\hat{i}=1}^{n} N_{\hat{i},p}(\xi)w_{\hat{i}}}$$

The $N_{i,p}$ are the B-splines defined by the Cox-de Boor recursion formula.

Summing over each NURBS basis function multiplied with its corresponding control point, $\mathbf{B}_i$ we obtain a NURBS geometry $\mathbf{C}(\xi)$:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} R_i^p(\xi)\mathbf{B}_i$$

Notice that the set of control points $\{\mathbf{B}_i\}_{i=1}^n$ will not usually be exactly interpolated by $\mathbf{C}(\xi)$, see figure (A.3) for an example of this. All basis functions have continuity C^{p-m_i} at the knots, where m_i is the number of times the knot appear in the knot vector (inside the elements the continuity is of course infinite). This implies that the endpoints will be interpolated (because of the open knot vector structure) and also that we should repeat p times any knot in the knot vector corresponding to a control point we demand interpolated.

Multiple parametric dimensions

When we try to imagine the geometries we are challenged by two different specifications: The dimension, d, of the control points, $\mathbf{B}_i$ decides which euclidean space, $\mathbb{R}^d$ the geometry will be depicted in. On the other hand the parametric dimension (how many parameters we must specify) decides the dimension of the object, if it is a curve, a surface or a solid.

We will start with a two dimensional parameter space, such that we get a curve, and therefore two separate knot vectors $\Xi = [\xi_1, ..., \xi_{n+p+1}]$ and $\mathcal{H} = [\eta_1, ..., \eta_{m+q+1}]$, where n and m are the number of basis functions in the ξ and η directions respectively while p and q are the respective orders. One often refer to the space of all possible combinations of the knot vector entries as the index space.

A control net $\mathbf{B}_{i,j}$ with corresponding weights $w_{i,j}$ must also be specified. (I like to think of the control net as a crude approximation of our geometry.) The univariate B-spline basis functions corresponding to the knot vectors Ξ and $\mathcal{H}$ are denoted $N_{i,p}(\xi)$ for $i = 1, ..., n$ and $M_{j,q}(\eta)$ for $j = 1, ..., m$ respectively. We now define the tensor product structured NURBS basis functions:

Definition 4. *NURBS basis functions in parameter space of two dimensions are defined by*

$$R_{i,j}^{p,q}(\xi,\eta) = \frac{N_{i,p}(\xi)M_{j,q}(\eta)w_{i,j}}{\sum_{\hat{j}=1}^{m}\sum_{\hat{i}=1}^{n} N_{\hat{i},p}(\xi)M_{\hat{j},q}(\eta)w_{\hat{i},\hat{j}}}, \qquad i = 1,...,n \qquad j = 1,...,m.$$

The open knot vectors only guarantee that we will interpolate the corner control points. The boundary points will only be "interpolated in one direction in parameter space". This should make us realize that if we want to interpolate a particular control point, we must make sure to repeat the corresponding knots in each individual knot vector p and q times respectively.

The surface is naturally constructed as a sum over the control points multiplied with the corresponding NURBS function as before. The generalization from two to three dimensions in parameter space (and further) is completely analogous to the generalization from one to two dimensions we just did here. As the parametric dimensions grow the notation becomes tedious, so for any parametric dimension we will often just write $R(\vec{\xi})$, suppressing the remaining parameters and indices. The denominator of a NURBS basis function is often written just $W(\vec{\xi})$.

By basis function we will from now onwards understand the NURBS basis function.

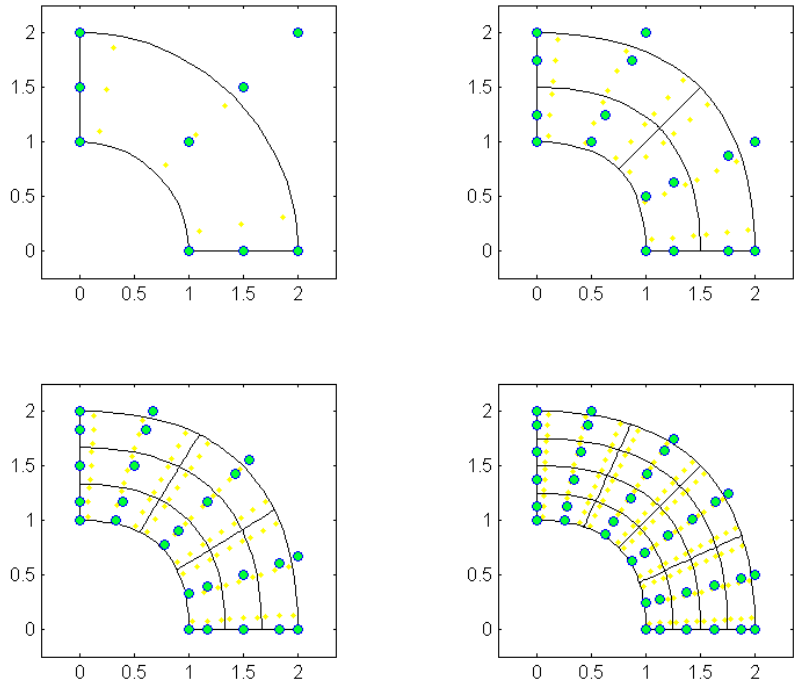


Figure A.4.: Mesh refinement of a quartered torus/doughnut by knot insertion for $n = m = 3, 4, 5, 6$ and $p = q = 2$. The elements, or knot spans, are framed in black lines. Control points are depicted in green. In addition the Gaussian quadrature points (used for numerical integration) are depicted in yellow.

A NURBS-geometry can be refined without changing the geometry. That is; the elements (knot spans) can be shrunk which also results in the control polygon closing in on the underlying geometry all while the NURBS-geometry is left exact. The control polygon is the polygon as specified by the control points, that would be $\{\mathbf{B}_{i,j}\}_{i,j=1}^{n,m}$ in two-dimensional parameter space, see figure (A.3) for a one-dimensional example. Further-

more the continuity of the basis is preserved, recall that the basis for the geometry is the same as for the numerical solution space and therefore it is clearly an important feature when we attempt to model higher derivatives.

We start off with only one parametric dimension and with the initial knot vector $\Xi = [\xi_1, ..., \xi_{n+p+1}]$. We also have a set of n control points, and we organize them by transposing the vector of their column vectors (forming a matrix) $B = [\vec{B}_1, ..., \vec{B}_n]^T$. The corresponding weights are stored in a column vector $w = [w_1, ..., w_n]^T$. In practice the weights will usually be stored as the last entry of each control point in the control point array. The first step is to formulate a new refined knot vector $\bar{\Xi}$ such that $\Xi \subset \bar{\Xi}$. The next step is to calculate the new control points $\bar{B}$ and their weights $\bar{w}$ by the formula

$$\bar{B} = \mathbf{T}^p B \tag{A.4}$$

$$\bar{w} = \mathbf{T}^p w \tag{A.5}$$

where the matrix $\mathbf{T}^p$ is defined recursively by

$$T_{ij}^{s+1} = \frac{\bar{\xi}_{i+s} - \xi_j}{\xi_{j+s} - \xi_j} T_{ij}^s + \frac{\xi_{j+s+1} - \bar{\xi}_{i+s}}{\xi_{j+s+1} - \xi_{j+1}} T_{ij+1}^s \text{ for } s = 0, 1, ..., p-1 \tag{A.6}$$

with base case

$$T_{ij}^0 = \begin{cases} 1 & \bar{\xi}_i \in [\xi_j, \xi_{j+1}) \\ 0 & \text{otherwise.} \end{cases} \tag{A.7}$$

When we operate with more than one dimension in parameter space we can make it easy for ourselves and refine the mesh in only one parametric direction at a time and also refining only one curve of the control polygon at a time. That is: First we refine Ξ. We calculate the new control points first row by row in index space (as defined by the indices of the knot vectors). Note that while there are $m + q + 1$ rows in index space there are only m rows of control points in two parametric dimensions, whereas in three parametric dimensions we would have $(m + q + 1)(l + r + 1)$ rows in index space but only ml rows of control points. Then we refine $\mathcal{H}$, and

calculate the new control points column by column. And so on in the case that we have additional parameters. Of course we could just as well refine all rows at once, then all columns. Because we can apply the knot insertion algorithm separately to each direction in parameter space it is therefore not necessary to generalize it to higher dimensions. (To clear any confusion regarding the relationship between parameter and index space take a look at figure (A.5).)

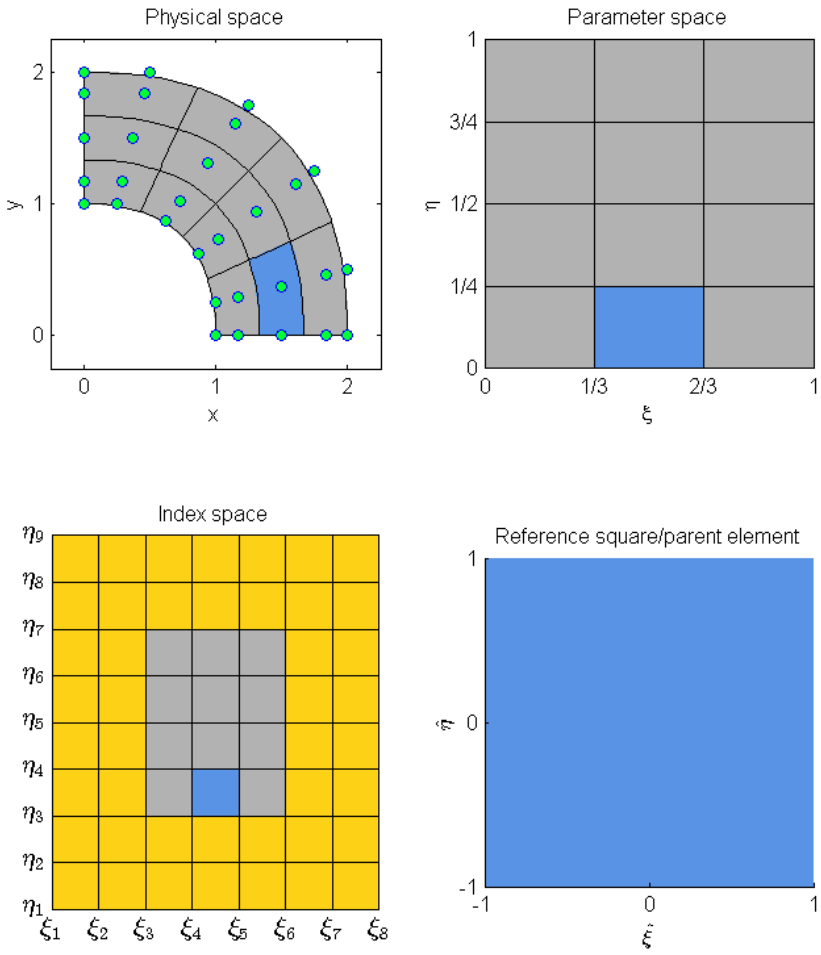


Figure A.5.: Single patch NURBS-geometry. Control points in green. The basis functions are quadratic in both directions. The knot vectors are $\Xi = [\xi_1, ..., \xi_8] = [0, 0, 0, 1/3, 2/3, 1, 1, 1]$ and $\mathcal{H} = [\eta_1, ..., \eta_9] = [0, 0, 0, 1/4, 1/2, 3/4, 1, 1, 1]$. Notice that the zero measure knot spans are coloured yellow and visible only in index space. When we integrate over element number two, coloured blue, we perform the integration on the parent element, map it onto the knot span in parameter space and finally onto the physical element. We use the notion of index space to determine/visualize the support of basis functions.

Bibliography

[1] Y. Bazilevs, V. M. Calo, J.A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199:229–263, 2010.

[2] T. Belytschko, N. Moës, S. Usui, and C. Parimi. Arbitrary discontinuities in finite elements. *International Journal for Numerical Methods in Engineering*, 50:993–1013, 2001.

[3] M. J. Borden, C.V. Verhoosel, M. A. Scott, T. J. R. Hughes, and C. M. Landis. A phase-field description of dynamic brittle fracture, 2011.

[4] B. Bourdin, G. A. Francfort, and J. J. Marigo. The variational approach to fracture. *J Elasticity*, 91:5–148, 2008.

[5] B. Bourdin, C. J. Larsen, and C. L. Richardson. A time-discrete model for dynamic fracture based on crack regularization. *J Fract*, 168:133–143, 2009.

[6] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis*. John Wiley & Sons, Ltd, 2009.

[7] T. Dokken, T. Lyche, and K. F. Pettersen. Locally refinable splines over box-partitions. Submitted to Computer Aided Geometric Design, 2012.

[8] M. R. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199:264–275, 2010.

[9] E. N. Dvorkin, A. M. Cuiti no, and G. Gioia. Finite elements with displacement interpolated embedded localization lines insensitive to

mesh size and distortions. *International Journal for Numerical Methods in Engineering*, 30:541–564, 1990.

[10] S. S. Ghorashi, N. Valizadeh, and S. Mohammadi. Extended isogeometric analysis for simulation of stationary and propagating cracks. *Int. J. Numer. Meth. Engng*, 2011.

[11] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2004.

[12] K. A. Johannessen, T. Kvamsdal, and T. Dokken. Isogeometric analysis using LR B-splines. Submitted to Computer Methods in Applied Mechanics and Engineering, 2012.

[13] T. Kvamsdal and K. M. Okstad. Error estimation based on superconvergent patch recovery using atatically admissible stress fields. *Int. J. Numer. Meth. Engng.*, 42:443–472, 1997.

[14] J. Lasry, Y. Renard J. Pommier, and M. Salaun. eXtended Finite Element Methods for thin cracked plates with Kirchoff-Love theory. *Computer Methods in Applied Mechanics and Engineering*, 2009.

[15] E. De Luycker, D. J. Benson, T. Belytschko, Y. Bazilevs, and M. C. Hsu. X-FEM in isogeometric analysis for linear fracture mechanics. *Int. J. Numer. Meth. Engng*, 87:541–565, 2011.

[16] C. Miehe, M. Hofacker, and F. Welschinger. A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits. *Computer Methods in Applied Mechanics and Engineering*, 199:2765–2778, 2010.

[17] C. Miehe and M. Lambrecht. Algorithms for computation of stresses and elasticity moduli in terms of Seth-Hill's family of generalized strain tensors. *Communications in numerical methods in engineering*, 17:337–353, 2001.

[18] C. Miehe, F. Welschinger, and M. Hofacker. Thermodynamically consistent phase-field models of fracture: Variational principles

and multi-field FE implementations. *Int. J. Numer. Meth. Engng*, 83:1273–1311, 2010.

[19] M. A. Scott, X. Li, T. W. Sederberg, and T. J. R. Hughes. Local refinement of analysis-suitable T-splines. *Computer Methods in Applied Mechanics and Engineering*, 213-216:206–222, 2012.

[20] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM Transactions on Graphics*, 22:477–484, 2003.

[21] J. C. Simo and F. Armero. Improved versions of assumed enhanced strain tri-linear elements for 3D finite deformation problems. *Computer Methods in Applied Mechanics and Engineering*, 110:359–386, 1993.