Universitat Politècnica de Catalunya
Programa de Doctorat en Enginyeria Civil

Departament de Matemàtica Aplicada III

Phase Field Approach to Fracture: Massive Parallelization and Crack Identification

by Vahid Ziaei-Rad

Doctoral Thesis
Advisors: Yongxing Shen, Marino Arroyo Balaguer

Barcelona, May 2016

Abstract

Phase Field Approach to Fracture: Massive Parallelization and Crack Identification

Vahid Ziaei-Rad

The phase field method has proven to be an important tool in computational fracture mechanics in that it does not require complicated crack tracking and is able to predict crack nucleation and branching. However, the computational cost of such a method is high due to a small regularization length parameter, which in turns restricts the maximum element size that can be used in a finite element mesh. In this work, we developed a massively parallel algorithm on the graphical processing unit (GPU) to alleviate this difficulty in the case of dynamic brittle fracture. In particular, we adopted the standard finite element method on an unstructured mesh combined with second order explicit integrators.

As the explicit methods fit nicely with the GPU paradigm especially in terms of thread and memory hierarchy, we solve an elastodynamic problem when the phase field update is based on a gradient flow, so that a fully explicit implementation is feasible. To ensure stability, we designed a time adaptivity strategy to account for the decreasing critical time step during the evolution of the fields.

We demonstrated the performance of the GPU-implemented phase field models by means of representative numerical examples, with which we studied the effect of the artificial viscosity, an artificial parameter to be input, and compared the crack path branching predictions from three popular phase field models. Moreover, we verified the method with convergence studies and performed a scalability study to demonstrate the desired linear scaling of the program in terms of the wall time per physical time as a function of the number of degrees of freedom.

One of the main ideas of the phase field method is to employ a smeared representation of discrete cracks. However, in some applications it is still convenient to have the explicit crack path available, or even to develop a mechanism to introduce crack paths to partially replace a smeared crack propagation model.

Hydraulic fracturing is one example for which a crack identification scheme may be useful. This is a technique used in the oil and gas industry, where fractures are propagated by high-pressure liquid inside them. With a phase field (or damage) approach, it becomes challenging how to impose the pressure loading on the rock mass exerted by the fracturing fluid, since most phase field approaches to date were developed for traction-free crack faces.

In this work, we present a variational method to identify the crack path from phase field approaches to fracture. The method is proven to be successful not only for a simple curved crack but also for multiple and branched cracks. The algorithm employs the non-maximum suppression technique, a procedure borrowed from the image processing field, to detect a bounding area which covers the ridge of the phase field profile. After that, it is continued with the step to determine a cubic spline to represent the crack path and to improve it via a constrained optimization process. To demonstrate the performance of our method, we provide the results with three sets of representative examples. The developed algorithm can be combined with one on crack opening, for more elaborate interpretation of phase field simulations. This is the topic of the next part of the work.

In this dissertation, we also provide a variational way to calculate the crack opening from phase field approaches to fracture. We also demonstrate the performance of our method with three sets of representative examples, and verify the results with a proper benchmark.

Having the crack geometry available from a phase field approach can provide more elaborate interpretation of the phase field simulations. It may also offer a possibility of developing less expensive numerical schemes for a fluid-driven crack propagation of impermeable solids. This will be the topic of our future work.

Acknowledgments

I have not reached where I am now merely by my own efforts. Not only because of the interdisciplinary essence of my work, which requires a variety of knowledge and skills from different fields, but also because working together with other people has boosted my individual abilities and has given me unique experiences.

First and foremost, I acknowledge the help of my advisor, Prof. Yongxing Shen, for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, immense knowledge, and kindness.

I want to express my gratitude to the faculty of LaCàN and Department of Applied Mathematics III, particularly Prof. Marino Arroyo Balaguer for his great support during the completion of this work. I am also truly thankful to Prof. Antonio Rodríguez-Ferran.

I would like to express my deep gratitude to Dr. Pablo Mata Almonacid for his gracious hospitality during our visit to Centro de la investigación en Ecosistemas de Patagonia (CIEP), Chile.

During this research work, I stayed one year at University of Michigan-Shanghai Jiao Tong University Joint Institute (UM-SJTU JI), Shanghai. I am indebted to all people there, especially Cheng Cheng, Li Shen, Jiahao Jiang, Can Wu, and Yihuan Li.

I express my thankfulness for the indispensable assistance of administrative staff of LaCàN, Imma Rius and Susanna Algarra, department of applied mathematics III, Maria Angels Huguet and Carme Lòpez, and school of civil engineering (CAMINS), Maria Jose Cueto and Silvia Aranda Gimo. I also offer thanks to LaCàN IT support, David Ortin.

I want to thank the support of the Marie Curie Career Integration Grant (European Commission) FP7-PEOPLE-2011-CIG-CompHydraulFrac and the National Science Foundation of China, grant number 11402146.

I cannot find words to express my gratitude to my parents, for their love and unconditional support throughout my life.

This dissertation is dedicated to my parents.

Contents

List of Figures

List of Tables

Outline

- In Chapter 1, we present and analyze an explicit algorithm to solve for a rate-dependent formulation of the phase field modeling of brittle fracture. The formulation is based on dynamic force balance coupled with a gradient-type phase field evolution law due to [Miehe et al., 2010b]. We adopt the standard finite element method on unstructured meshes and second-order explicit time integrators for the implementation.

- In Chapter 2, we develop a massively parallel program to implement our explicit algorithm to solve for a rate-dependent formulation of the phase field modeling of brittle fracture on NVIDIA GPUs using the Compute Unified Device Architecture (CUDA).

- In Chapter 3, we present three representative sets of numerical examples to demonstrate the performance of the proposed algorithm for the phase field fracture models. An NVIDIA Kepler K20M GPU and two Intel Xeon E5-2670 (8 Cores, 2.6GHz, 20MB Cache, 8.0GT) CPUs were used in the calculations. The three sets of examples include a tension test, a shear test, and a symmetric bending test.

- In Chapter 4, we build a crack identification scheme by taking advantage of the variational structure of the phase field approach, although the scheme can also be applied to a solution from a damage model. The methodology is built on the so called *equivalent phase field* of a given curve, which is the generalization from the analytic phase field solution from the potential energy functional in a special case.

- In Chapter 5, we offer a variational approach to calculate the crack opening from a smeared crack representation.
- In Chapter 6 we made some conclusions and future work.
- Each of these chapters is partially or fully based on the following publications:
    - Vahid Ziaei-Rad and Yongxing Shen, Massive parallelization of the phase field formulation for crack propagation with time adaptivity, *Submitted.*
      Chapter 1, 2, 3
    - Vahid Ziaei-Rad, Li Shen, Jiahao Jiang, and Yongxing Shen, Identifying the crack path for the phase field approach to fracture with non-maximum suppression, *Submitted.*
      Chapter 4
    - Vahid Ziaei-Rad, Cheng Cheng, and Yongxing Shen, A variationally consistent method for identifying the crack opening from a phase field solution, *In preparation.*
      Chapter 5

Chapter 1

Phase field formulation for crack propagation with time adaptivity

1.1 Introduction

Understanding and predicting fracture and failure in materials is important in engineering designs. Computational modeling is a way to study the fracture phenomena, especially in cases in which experiments are impractical or too expensive. As a consequence, a wide variety of fracture models have been developed. In the case of brittle fracture, many of such models were developed based on Griffith's theory in which crack nucleation and propagation are determined by a critical value of the energy release rate. Many numerical approaches have been constructed with the standard finite element methods (FEMs) [Rangarajan et al., 2015] or the extended finite element method (XFEM) [Moës et al., 1999, Shen and Lew, 2010a,b, 2014] in conjunction with Griffith-type fracture models, such as the approach presented in [Xu et al., 2014]. These approaches explicitly represent cracks as discontinuities. They require: (1) either always adjusting the mesh in traditional FEMs [Rangarajan et al., 2015] or introducing enrichments like the XFEM; and (2) extra input to predict crack nucleation and branching.

Based on energy minimization, the variational theory of fracture was proposed by Francfort and Marigo [Francfort and Marigo, 1998] to formulate brittle fracture, which takes into account both bulk elastic energy and the surface energy due to creation of cracks. Capable of predicting crack initiation, this class of approaches has attracted attention in the mathematicians' community [Del Piero et al., 2007, Chambolle et al., 2009, Marigo, 2010].

The regularized version of the variational theory of fracture was introduced by Bourdin *et al.* [Bourdin et al., 2008], which was later adopted by the engineers' community as a formulation more suitable for numerical simulations. Later the *phase field* formulation of fracture has become a synonym and a more popular name for this class of methods, see, e.g., [Miehe et al., 2010b, Hofacker and Miehe, 2012b, Borden et al., 2012].

In essence, phase field models for fracture employ a continuous field variable, called the *phase field*, to represent cracks. Feng and Prohl [Feng and Prohl, 2004], Hakim and Karma [Hakim and Karma, 2009], da Silva *et al.* [da Silva et al., 2013], and Babadjian and Millot [Babadjian and Millot, 2014] studied the relation between the phase field formulation and its sharp crack limit.

The main advantage of using a phase field is that the evolution of fracture surfaces follows from the solution of a coupled system of partial differential equations. In contrast to explicit descriptions of cracks, phase field descriptions do not require explicitly tracking the discontinuities in the displacement field. This significantly reduces implementation complexity, and is anticipated to be particularly advantageous when multiple branching and merging cracks are considered in three dimensions [Miehe et al., 2010b,a]. While the phase field method has mainly been employed to study quasi-static brittle fracture [Hakim and Karma, 2009, Miehe et al., 2010a, Dal Maso and Lazzaroni, 2010, Kuhn and Müller, 2010], it has also been extended successfully to dynamic problems [da Silva et al., 2013, Hofacker and Miehe, 2012a, Karma et al., 2001, Larsen, 2010, Bourdin et al., 2011, Schlüter et al., 2014] and ductile fracture [Verhoosel and de Borst, 2013, Charlotte et al., 2006, Ulmer et al., 2013]. Moreover, more sophisticated models for the phase field approximation have been developed which lead to a higher regularity of the phase field solution than H^1, e.g., H^2 for [Borden et al., 2014] and [Gomez et al., 2014]. See also [Amiri et al., 2014, Ulmer et al., 2012] for fracture in plates and shells simulated with the phase field model.

In this chapter, we present and analyze an explicit algorithm to solve for a rate-dependent formulation of the phase field modeling of brittle fracture. The formulation is based on dynamic force balance coupled with a gradient-type phase field evolution law due to [Miehe et al., 2010b]. We adopt the standard finite element method on unstructured meshes and second-order explicit time integrators for the implementation.

A strong motivation for adopting a fully explicit method is that it fits nicely with the GPU architecture. Hence, we aim for developing a massively parallel program to implement the present algorithm on GPU in the forthcoming chapter.

As is common to explicit methods, a critical time step is essential to ensure stability. The coupled problem at hand can be divided into an elastodynamic half problem and a phase field half problem, and the critical time step is the smaller of those of the half problems. As time evolves, the critical time step for the elastodynamic half problem increases but that for the phase field half problem decreases. Beginning from some instant, the critical time step for the entire problem decreases with time. Hence, an efficient implementation requires an efficient lower bound computation for the critical time step. This time adaptivity scheme will be elaborated in Section 1.3.3. With regard to the choice between explicit and implicit schemes, we refer the reader to Keyes *et al.* [Keyes et al., 2006] for a thorough discussion. In our case, as we will demonstrate, we observe linear scaling of the wall time per physical time as a function of the number of degrees of freedom. Hence, our method will be no worse than most classical implicit schemes in terms of scaling.

In addition to the phase field model proposed in [Miehe et al., 2010b] which only allows the phase field evolution to occur at the expense of strain energy due to tension and expansion but not compression, we made a systematic comparison with two other models: one that allows phase field evolution to occur due to any strain energy, and one due to [Amor et al., 2009] that allows only volumetric expansion and deviatoric deformation to contribute to the phase field dissipation but not volumetric compression.

The developed methodology allows us to perform interesting studies on, e.g., the different crack paths due to the different loading rates, the predictions from the aforementioned various phase field models, and the effect of the artificial viscosity, which is necessary for an explicit scheme for phase field simulations for fracture, but is optional for an implicit scheme.

This chapter will proceed as follows. In Section 1.2, we first recall the governing equations of the fracture models, including the phase field formulations for the variational description of brittle fracture. In Section 1.3, we adopt standard explicit methods to solve the initial boundary value problem presented in Section 1.2 and then elaborate on the adaptive strategy to determine the critical time step on the fly.

1.2 Problem statement

This section devotes to setting up the problem of brittle fracture modeled by the phase field formulation. To this end, in Section 1.2.1 we first recall the variational formulation in the sharp-crack case, where the crack path is part of the unknown. Then in Section 1.2.2 we recapitulate a regularized formulation, which, in the sense of Γ-convergence, converges to the variational formulation. In this regularized formulation we keep the expression of strain energy density general to accommodate different phase field models in the literature. To facilitate the implementation in the GPU with explicit integration, in Section 1.2.3 we adopt the idea in [Miehe et al., 2010b] to modify the formulation by introducing an artificial viscosity term. Then in Section 1.2.4 we introduce the weak form corresponding to the strong form in Section 1.2.3. Finally, we list three phase field models in the literature in Section 1.2.5 to be compared in subsequent sections.

1.2.1 Variational formulation of brittle fracture

We consider a two-dimensional (plane stress or plane strain) isotropic linear elastic solid initially occupying the open polygon Ω and want to determine its evolution in the time interval $[0, t_f]$. Let $\Gamma_D, \Gamma_N \subseteq \partial\Omega$ be such that $\Gamma_D \cup \Gamma_N = \partial\Omega$ and $\Gamma_D \cap \Gamma_N = \emptyset$, and $\boldsymbol{u}_D : \Gamma_D \times [0, t_f] \to \mathbb{R}^2$ and $\boldsymbol{t}_N : \Gamma_N \times [0, t_f] \to \mathbb{R}^2$ be prescribed displacement and traction boundary conditions. We also let $\rho \in \mathbb{R}^+$ and $\mathbf{b} : \Omega \times [0, t_f] \to \mathbb{R}^2$ denote the (constant) density of the solid and body force per unit mass exerted to the solid. According to Borden *et al.* [Borden et al., 2012], the variational formulation for the brittle fracture of the solid consists in finding the stationary point of the following Lagrangian:

$$L[\boldsymbol{u}, \dot{\boldsymbol{u}}] := \int_{\Omega\setminus\Gamma} \left\{ \frac{1}{2}\rho \dot{\boldsymbol{u}} \cdot \dot{\boldsymbol{u}} - \psi_0[\boldsymbol{\varepsilon}(\boldsymbol{u})] \right\} \, d\Omega + \int_{\Gamma_N} \boldsymbol{t}_N \cdot \boldsymbol{u} \, d\Gamma + \int_{\Omega\setminus\Gamma} \rho \mathbf{b} \cdot \boldsymbol{u} \, d\Omega - g_c|\Gamma|$$

among all $\boldsymbol{u} : \mathbb{R}^2 \times [0, t_f] \to \mathbb{R}^2$ that are bounded deformation functions of Ω and that satisfy

$$\boldsymbol{u} = \boldsymbol{u}_D, \quad \text{on } \Gamma_D \times [0, t_f]. \tag{1.1}$$

Here the dot symbol over a variable, $\dot{\square}$, denotes differentiation in time t, and $\Gamma = \Gamma(\boldsymbol{u}) \subset \Omega$ is the set of discontinuities of $\boldsymbol{u}(\cdot, t)$ such that the irreversibility of the crack is satisfied [Francfort and Marigo, 1998], i.e., for any $t_1, t_2 \in [0, t_f]$, $t_1 < t_2$, we have $\Gamma[\boldsymbol{u}(\cdot, t_1)] \subseteq \Gamma[\boldsymbol{u}(\cdot, t_2)]$. We let $|\Gamma|$ denote the length of Γ, $\psi[\boldsymbol{\varepsilon}(\boldsymbol{u})]$ the strain energy density which depends on the strain

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) := \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right) \tag{1.2}$$

as

$$\psi_0(\boldsymbol{\varepsilon}) := \frac{\lambda}{2}(\operatorname{tr} \boldsymbol{\varepsilon})^2 + \mu \|\boldsymbol{\varepsilon}\|^2,$$

with λ and μ Lamé constants such that $\mu > 0$ and $\lambda + 2\mu > 0$, and $g_c \in \mathbb{R}^+$ the strain energy release rate, the strain energy released per unit length of crack extension. Here superscript ^{T} denotes the transpose operation on a tensor, and $\|\cdot\| : \boldsymbol{\varepsilon} \mapsto \sqrt{\boldsymbol{\varepsilon} : \boldsymbol{\varepsilon}}$ denotes the Frobenius norm of a tensor.

1.2.2 Regularized variational formulation of brittle fracture

To develop a numerical method to approximate (1.2), we replace the sharp-crack description with a phase field description, where the phase field is denoted as $d : \Omega \to [0,1]$. In particular, regions with $d = 0$ and $d = 1$ correspond to perfect and fully broken states of the material, respectively. We remark that we have adopted a convention of the value of d following that of Miehe *et al.* [Miehe et al., 2010b] and that of the damage mechanics community [Voyiadjis and Mozaffari, 2013, Pham et al., 2011]. Other authors such as Bourdin *et al.* [Bourdin et al., 2008] and Borden *et al.* [Borden et al., 2012, 2014] have used a convention with the meanings of $d = 0$ and $d = 1$ reversed.

Let

$$\begin{aligned}
\mathscr{S}_u &:= \left\{ \boldsymbol{u} \in H^1\left(\Omega; \mathbb{R}^2\right) \times [0, t_f] \middle| \boldsymbol{u}(\cdot, t) = \boldsymbol{u}_D(\cdot, t) \text{ on } \Gamma_D \right\}, \\
\mathscr{S}_d &:= H^1(\Omega) \times [0, t_f],
\end{aligned}$$

then the regularized variational formulation reads: Find $(\boldsymbol{u}, d) \in \mathscr{S}_u \times \mathscr{S}_d$ that is the stationary point of the following functional:

$$\begin{aligned}
L_l[\boldsymbol{u}, \dot{\boldsymbol{u}}, d] := \int_\Omega \left\{ \frac{1}{2}\rho \dot{\boldsymbol{u}} \cdot \dot{\boldsymbol{u}} - \psi[\boldsymbol{\varepsilon}(\boldsymbol{u}), d] \right\} \, d\Omega + \int_{\Gamma_N} \boldsymbol{t}_N \cdot \boldsymbol{u} \, d\Gamma + \int_\Omega \rho \mathbf{b} \cdot \boldsymbol{u} \, d\Omega \\
- \frac{g_c}{2} \int_\Omega \left(\frac{d^2}{l} + l \nabla d \cdot \nabla d \right) \, d\Omega,
\end{aligned} \tag{1.3}$$

where l is a length scale such that when $l \to 0$, the regularized formulation Γ-converges to that with explicit crack representation (see Bourdin *et al.* [Bourdin et al., 2008] for the proof of the static anti-plane case). Here $\psi(\boldsymbol{\varepsilon}, d)$ is the strain energy density degraded by the phase field such that $\psi(\boldsymbol{\varepsilon}, 0) = \psi_0(\boldsymbol{\varepsilon})$ and that $\psi(\boldsymbol{\varepsilon}, d_1) \geq \psi(\boldsymbol{\varepsilon}, d_2)$ if $d_1 < d_2$.

The Euler-Lagrange equations of (1.3) are

$$-\rho\ddot{\boldsymbol{u}} + \operatorname{div}\boldsymbol{\sigma} + \rho\mathbf{b} = \mathbf{0}, \qquad \text{in } \Omega \times [0, t_f], \tag{1.4a}$$

$$-\frac{\partial\psi}{\partial d} - \frac{g_c}{l}\left(d - l^2\Delta d\right) = 0, \qquad \text{in } \Omega \times [0, t_f], \tag{1.4b}$$

$$\boldsymbol{\sigma}\cdot\boldsymbol{n} - \boldsymbol{t}_N = \mathbf{0}, \qquad \text{on } \partial_N\Omega \times [0, t_f], \tag{1.4c}$$

$$\frac{\partial d}{\partial \boldsymbol{n}} = 0, \qquad \text{on } \partial\Omega \times [0, t_f], \tag{1.4d}$$

where

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\varepsilon}, d) := \frac{\partial\psi}{\partial\boldsymbol{\varepsilon}}$$

is the Cauchy stress, the double-dot symbol, $\ddot{\square}$, means the second derivative with respect to t, and $\boldsymbol{n}$ is the unit outer normal to $\partial\Omega$. Here (1.4a) expresses the momentum conservation of the solid, (1.4b) defines the phase field evolution, and (1.4c) and (1.4d) are Neumann boundary conditions for $\boldsymbol{u}$ and d, respectively.

1.2.3 Rate-dependent form of the formulation

As the explicit methods fits nicely with the GPU paradigm especially in terms of thread and memory hierarchy, we convert the formulation (1.4) into a time-dependent form [Miehe et al., 2010b] with (1.4b) replaced by

$$\dot{d} = \begin{cases} \dfrac{1}{\eta}\left\langle -\dfrac{\partial\psi}{\partial d} - \dfrac{g_c}{l}\left(d - l^2\Delta d\right)\right\rangle_+, & d < 1, \\ 0, & \text{otherwise}, \end{cases} \tag{1.4b'}$$

where $\eta > 0$ is the *artificial viscosity*, a material parameter to be input, and $\langle a\rangle_\pm := (|a| \pm a)/2$ for all $a \in \mathbb{R}$.

The initial conditions for the problem can be posed as

$$\boldsymbol{u}(\cdot, 0) = \boldsymbol{u}_0, \quad \dot{\boldsymbol{u}}(\cdot, 0) = \boldsymbol{v}_0, \quad d(\cdot, 0) = d_0, \tag{1.5}$$

where $\boldsymbol{u}_0, \boldsymbol{v}_0 : \Omega \to \mathbb{R}^2$ and $d_0 : \Omega \to [0,1]$.

The time-dependent problem to be solved can now be stated as:

> Find $(\boldsymbol{u}, d)$ subjected to the partial differential equations (1.4a) and (1.4b'), initial conditions (1.5), and boundary conditions (1.1), (1.4c), and (1.4d).

1.2.4 The weak form

To facilitate the numerical computation with the FEM, we next state the weak form of the problem. In particular, to handle the second-order derivative inside the operator $\langle\cdot\rangle_+$ in (1.4b'), we introduce a new variable $d^\#$ such that $\dot{d} = \langle d^\# \rangle_+$ at all t.

To proceed, we let the test function spaces be

$$\begin{aligned}
\mathscr{V}_u &:= \left\{ \boldsymbol{w}(\cdot,t) \in H^1\left(\Omega;\mathbb{R}^2\right) \middle| \boldsymbol{w}(\cdot,t) = \boldsymbol{0} \text{ on } \Gamma_D \right\}, \\
\mathscr{V}_d &:= H^1(\Omega).
\end{aligned}$$

The weak form can be stated as: Find $(\boldsymbol{u}, d^\#, d) \in \mathscr{S}_u \times \mathscr{S}_d \times \mathscr{S}_d$ such that for all $t \in (0, t_f]$, $\boldsymbol{w} \in \mathscr{V}_u$ and $q \in \mathscr{V}_d$,

$$\begin{cases}
(\boldsymbol{w}, \rho\ddot{\boldsymbol{u}}) + a_d(\boldsymbol{u}, \boldsymbol{w}) = f(\boldsymbol{w}), \\
\left(q, \eta d^\#\right) = -\left(q, \dfrac{\partial \psi}{\partial d}\right) - \dfrac{g_c}{l}(q, d) - g_c l^2 (\nabla q, \nabla d), \\
\left(q, \dot{d}\right) = \left(q, \langle d^\# \rangle_+\right), \\
(\boldsymbol{w}, \boldsymbol{u}(\cdot,0)) = (\boldsymbol{w}, \boldsymbol{u}_0), \\
(\boldsymbol{w}, \dot{\boldsymbol{u}}(\cdot,0)) = (\boldsymbol{w}, \boldsymbol{v}_0), \\
(q, d(\cdot,0)) = (q, d_0),
\end{cases}$$

where for any scalar fields q_1 and q_2 and any vector fields $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$,

$$\begin{aligned}
(q_1, q_2) &= \int_\Omega q_1 q_2 \, d\Omega, \\
(\boldsymbol{w}_1, \boldsymbol{w}_2) &= \int_\Omega \boldsymbol{w}_1 \cdot \boldsymbol{w}_2 \, d\Omega, \\
a_d(\boldsymbol{w}_1, \boldsymbol{w}_2) &= \int_\Omega \boldsymbol{\sigma}[\boldsymbol{\varepsilon}(\boldsymbol{w}_1), d] : \nabla \boldsymbol{w}_2 \, d\Omega, \\
f(\boldsymbol{w}_1) &= \int_\Omega \rho \mathbf{b} \cdot \boldsymbol{w}_1 \, d\Omega + \int_{\Gamma_N} \boldsymbol{t}_N \cdot \boldsymbol{w}_1 \, d\Gamma.
\end{aligned}$$

1.2.5 Phase field models

In this section we recapitulate three phase field models for brittle fracture in solids. These models mainly differ in the choice of the strain energy density $\psi(\boldsymbol{\varepsilon}, d)$, for which we adopt the following general form:

$$\psi(\boldsymbol{\varepsilon}, d) = (1-d)^2 \psi_+(\boldsymbol{\varepsilon}) + \psi_-(\boldsymbol{\varepsilon}), \tag{1.6}$$

where $\psi_+(\boldsymbol{\varepsilon})$ and $\psi_-(\boldsymbol{\varepsilon})$ are such that $\psi_+(\boldsymbol{\varepsilon}) + \psi_-(\boldsymbol{\varepsilon}) = \psi_0(\boldsymbol{\varepsilon})$. The choices of $\psi_+(\boldsymbol{\varepsilon})$ and $\psi_-(\boldsymbol{\varepsilon})$ to be studied are tabulated in Table 1.1.

Table 1.1: Three phase field models for crack propagation in terms of the expression of the strain energy density. Note that the trace operator and the principal strains are understood in the three-dimensions setting, which accommodates both the plane stress and plane strain cases.

Model	Reference	$\psi_+(\boldsymbol{\varepsilon})$	$\psi_-(\boldsymbol{\varepsilon})$
A	[Bourdin et al., 2008]	$\psi_0(\boldsymbol{\varepsilon})$	0
B	[Amor et al., 2009]	$(\lambda/2 + \mu/3)\langle \operatorname{tr} \boldsymbol{\varepsilon} \rangle_+^2 + \mu \| \operatorname{dev} \boldsymbol{\varepsilon} \|^2$	$(\lambda/2 + \mu/3)\langle \operatorname{tr} \boldsymbol{\varepsilon} \rangle_-^2$
C	[Miehe et al., 2010b]	$(\lambda/2)\langle \operatorname{tr} \boldsymbol{\varepsilon} \rangle_+^2 + \mu \sum_{i=1}^3 \langle \varepsilon_i \rangle_+^2$	$(\lambda/2)\langle \operatorname{tr} \boldsymbol{\varepsilon} \rangle_-^2 + \mu \sum_{i=1}^3 \langle \varepsilon_i \rangle_-^2$

We remark here that references [Bourdin et al., 2008, Amor et al., 2009, Miehe et al., 2010b] all have an additional small number k with $0 < k \ll 1$ and $k = o(l)$ added to the coefficient of $\psi_+(\boldsymbol{\varepsilon})$, i.e., they have $[(1-d)^2 + k]$ instead of $(1-d)^2$

in (1.6), to prevent the lack of stiffness for completely cracked portions of the solid. Here, however, we can set $k = 0$ thus use (1.6) because we will solve the problem in a fully explicit scheme and thus do not need to invert any stiffness matrix.

With the form of $\psi(\boldsymbol{\varepsilon}, d)$ chosen, the corresponding stress tensor takes the form

$$\boldsymbol{\sigma}(\boldsymbol{\varepsilon}, d) = \frac{\partial \psi}{\partial \boldsymbol{\varepsilon}} = (1-d)^2 \frac{\partial \psi_+}{\partial \boldsymbol{\varepsilon}} + \frac{\partial \psi_-}{\partial \boldsymbol{\varepsilon}},$$

see Table 1.2.

Table 1.2: Three phase field models for crack propagation in terms of the stress-strain relation

Model	$\partial \psi_+ / \partial \boldsymbol{\varepsilon}$	$\partial \psi_- / \partial \boldsymbol{\varepsilon}$
A	$\lambda(\operatorname{tr} \boldsymbol{\varepsilon})\mathbf{1} + 2\mu\boldsymbol{\varepsilon}$	0
B	$(\lambda + 2\mu/3)\langle \operatorname{tr} \boldsymbol{\varepsilon}\rangle_+ \mathbf{1} + 2\mu \operatorname{dev} \boldsymbol{\varepsilon}$	$(\lambda + 2\mu/3)\langle \operatorname{tr} \boldsymbol{\varepsilon}\rangle_- \mathbf{1}$
C	$\lambda\langle \operatorname{tr} \boldsymbol{\varepsilon}\rangle_+ \mathbf{1} + 2\mu \sum_{i=1}^{3} \langle \varepsilon_i \rangle_+ \mathbf{n}_i \otimes \mathbf{n}_i$	$\lambda\langle \operatorname{tr} \boldsymbol{\varepsilon}\rangle_- \mathbf{1} + 2\mu \sum_{i=1}^{3} \langle \varepsilon_i \rangle_- \mathbf{n}_i \otimes \mathbf{n}_i$

Below we discuss these models and define the involved symbols.

Model A This is the original model proposed for similar formulations. It is convenient in that ψ is analytic in both d and $\boldsymbol{\varepsilon}$.

Model B This model assumes that both volumetric expansion and deviatoric deformation contribute to crack propagation but not volumetric compression. As a result, the formulation involves a decomposition of $\boldsymbol{\varepsilon}$ into volumetric and deviatoric contributions:

$$\operatorname{vol} \boldsymbol{\varepsilon} := \frac{1}{3}(\operatorname{tr} \varepsilon)\mathbf{1}, \quad \operatorname{dev} \boldsymbol{\varepsilon} := \boldsymbol{\varepsilon} - \operatorname{vol} \boldsymbol{\varepsilon}.$$

Note here that the trace operator is understood in the three-dimensions setting, which accommodates both the plane stress and plane strain cases.

Model C This model postulates that the stress degradation is due to a combination of tensile loading and volumetric expansion. Hence the definition of $\psi_+(\boldsymbol{\varepsilon})$ involves the spectral decomposition of $\boldsymbol{\varepsilon}$, i.e., the principal strains ε_i, $i = 1, 2, 3$, and the corresponding principal directions $\mathbf{n}_i$, such that

$$\boldsymbol{\varepsilon}\mathbf{n}_i = \varepsilon_i \mathbf{n}_i, \quad \mathbf{n}_i \cdot \mathbf{n}_j = \delta_{ij},$$

where *no* summation is assumed for the index i, and δ_{ij} is the Kronecker delta.

1.3 Numerical solution

In this section we adopt standard procedures to obtain explicit methods to solve the initial boundary value problem presented in Section 1.2. In the sequel we will introduce the semi-discrete and discrete formulations in Sections 1.3.1 and 1.3.2, respectively. After that, in Section 1.3.3 we will discuss an issue unique to the explicit phase field formulation of crack evolution, namely, the decreasing critical time step as time evolves, and then present our solution to this problem by adopting an adaptive time step based on the current solution. As a consequence of the need of time step adaptivity, only one-step time integrators will be adopted.

1.3.1 Semi-discrete Galerkin formulation

We discretize the domain Ω with a mesh family $\{\mathscr{T}_h\}$, each member characterized by the mesh size h. Let H denote the set of all nodes, then we approximate $(\boldsymbol{u}, d)$ with the standard first-order (P_1 or Q_1) finite element basis functions associated with all nodes $A \in H$:

$$\boldsymbol{u}(\boldsymbol{x}, t) \doteq \sum_{A \in H} \sum_{i=1}^{2} \mathrm{u}_P(t) N_A(\boldsymbol{x}) \boldsymbol{e}_i, \quad d(\boldsymbol{x}, t) \doteq \sum_{A \in H} \mathrm{d}_A(t) N_A(\boldsymbol{x}),$$

where $\boldsymbol{e}_i$ is the unit vector along the ith Cartesian direction, and $P := 2(A-1)+i$ is a lumped index for the combination (A, i). Then standard Galerkin approximations yield the following matrix form of the problem

$$\mathbf{M}\ddot{\mathbf{u}} = \mathbf{F} - \mathbf{K}(\mathbf{u},\mathbf{d})\mathbf{u}, \tag{1.7a}$$

$$\dot{\mathbf{d}} = \left\langle \overline{\mathbf{M}}^{-1}\mathbf{Y}\left(\mathbf{u},\mathbf{d}\right)\right\rangle_{+}, \tag{1.7b}$$

with initial conditions determined from $\boldsymbol{u}_0$, $\dot{\boldsymbol{u}}_0$, and d_0. Here $\mathbf{u} = \{\mathrm{u}_P\}$ and $\mathbf{d} = \{\mathrm{d}_A\}$ are vectors that contain the time-dependent nodal degrees of freedom (DOFs) of $\boldsymbol{u}$ and d, respectively, and the explicit expressions of the matrices involved in (1.7) read

$$K_{PQ}(\mathbf{u},\mathbf{d}) := a_d(N_A\boldsymbol{e}_i, N_B\boldsymbol{e}_j), \tag{1.8a}$$

$$M_{PQ} := (N_A, 1)\delta_{AB}\delta_{ij}, \tag{1.8b}$$

$$F_P := f(N_A\boldsymbol{e}_i), \tag{1.8c}$$

$$\overline{M}_{AB} := \eta(N_A, 1)\delta_{AB}, \tag{1.8d}$$

$$Y_A\left(\mathbf{u},\mathbf{d}\right) := \begin{cases} Y_A^-\left(\mathbf{u},\mathbf{d}\right), & \mathrm{d}_A < 1, \\ 0, & \text{otherwise}, \end{cases} \tag{1.8e}$$

where

$$Y_A^-\left(\mathbf{u},\mathbf{d}\right) := -\left(N_A, \frac{\partial\psi}{\partial d}\right) - \frac{g_c}{l}(N_A, d) - g_c l^2(\nabla N_A, \nabla d),$$

$Q := 2(B-1) + j$ is the DOF index for $\boldsymbol{u}$ associated with node B and Cartesian direction $j = 1, 2$, and we have lumped mass matrices $\mathbf{M}$ and $\overline{\mathbf{M}}$ into diagonal matrices. Note that since $\overline{\mathbf{M}}$ is diagonal and positive definite, (1.7b) is equivalent to

$$\overline{\mathbf{M}}\dot{\mathbf{d}} = \left\langle\mathbf{Y}\left(\mathbf{u},\mathbf{d}\right)\right\rangle_{+}.$$

1.3.2 Fully discrete formulations

We discretize the time interval $[0, t_f]$ into small increments $0 = t_0 < t_1 < \ldots < t_N = t_f$ and define $\Delta t_k := t_k - t_{k-1}$. Note that the time step sizes $\{\Delta t_k\}$ may be different from each other, due to stability considerations.

Here we adopt standard second-order methods for time integration: central difference for the displacement field and Heun's method for the phase field. The approximate solutions of $\mathbf{u}(t_k)$, $\dot{\mathbf{u}}(t_k)$, $\ddot{\mathbf{u}}(t_k)$, $\mathbf{d}(t_k)$, and $\dot{\mathbf{d}}(t_k)$ are denoted as $\mathbf{u}_k$, $\mathbf{v}_k$, $\mathbf{a}_k$, $\mathbf{d}_k$, and $\mathbf{r}_k$, respectively. To simplify the calculation of (1.8e), we will always perform the integration for $\mathbf{d}$ with $\mathbf{Y}(\mathbf{u}, \mathbf{d})$ replaced by $\mathbf{Y}^-(\mathbf{u}, \mathbf{d})$ in (1.7b), and then replace all those entries exceeding unity by unity.

The overall algorithm is given in Algorithm 1, taking into account the interdependence of the quantities in the computation and the consideration of stability, both of which we will discuss in the sequel.

Algorithm 1: Algorithm to update the solution from time step k to $k+1$

```
input  : $\mathbf{u}_k$, $\mathbf{v}_k$, $\mathbf{d}_k$, $\Delta t_{cu}^{LB}$, and $\Delta t_{cd}^{LB}$
output : $\Delta t_{k+1}$, $\mathbf{u}_{k+1}$, $\mathbf{v}_{k+1}$, and $\mathbf{d}_{k+1}$
Set $\Delta t_{k+1} \leq \Delta t_c^{LB}$ where $\Delta t_c^{LB}$ is given by (1.14)
Update $\mathbf{u}_{k+1}$, $\mathbf{v}_{k+1}$, and $\mathbf{d}_{k+1}$ in the following order:
$(1.9a) \to (1.10a) \to (1.10b) \to (1.9b) \to (1.9c) \to (1.10c) \to (1.10d) \to (1.9d) \to (1.9e)$
```

Central difference method for displacement integration The central difference method consists in computing $\mathbf{u}_{k+1}$, $\mathbf{v}_{k+1}$, and $\mathbf{a}_{k+1}$ from $\mathbf{u}_k$, $\mathbf{v}_k$, and $\mathbf{a}_k$ according to

$$\begin{cases} \mathbf{M}\mathbf{a}_{k+1} + \mathbf{K}(\mathbf{u}_{k+1}, \mathbf{d}_{k+1})\mathbf{u}_{k+1} = \mathbf{F}_{k+1}, \\ \mathbf{u}_{k+1} = \mathbf{u}_k + \Delta t \mathbf{v}_k + \dfrac{\Delta t^2}{2}\mathbf{a}_k, \\ \mathbf{v}_{k+1} = \mathbf{v}_k + \dfrac{\Delta t}{2}(\mathbf{a}_k + \mathbf{a}_{k+1}). \end{cases}$$

An efficient implementation of this algorithm is given by Hughes [Hughes, 1987,

Section 9.1.1]

$$\mathbf{M}\mathbf{a}_k = \mathbf{F}_k - \mathbf{K}(\mathbf{u}_k, \mathbf{d}_k)\mathbf{u}_k, \tag{1.9a}$$

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \Delta t \mathbf{v}_k + \frac{\Delta t^2}{2}\mathbf{a}_k, \tag{1.9b}$$

$$\mathbf{v}^*_{k+1} = \mathbf{v}_k + \frac{\Delta t}{2}\mathbf{a}_k, \tag{1.9c}$$

$$\mathbf{M}\mathbf{a}_{k+1} = \mathbf{F}_{k+1} - \mathbf{K}(\mathbf{u}_{k+1}, \mathbf{d}_{k+1})\mathbf{u}_{k+1}, \tag{1.9d}$$

$$\mathbf{v}_{k+1} = \mathbf{v}^*_{k+1} + \frac{\Delta t}{2}\mathbf{a}_{k+1}. \tag{1.9e}$$

To start the process, we determine $\mathbf{u}_0$, $\mathbf{v}_0$, and $\mathbf{d}_0$ from the initial conditions (1.5), and $\mathbf{a}_0$ from

$$\mathbf{M}\mathbf{a}_0 = \mathbf{F}_0 - \mathbf{K}(\mathbf{u}_0, \mathbf{d}_0)\mathbf{u}_0.$$

Heun's method for the phase field The Heun's method, a special case of second-order Runge-Kutta methods, updates the solution from $(\mathbf{d}_k, \mathbf{r}_k)$ to $(\mathbf{d}_{k+1}, \mathbf{r}_{k+1})$ according to:

$$\overline{\mathbf{M}}\mathbf{r}_k = \left\langle \mathbf{Y}^-(\mathbf{u}_k, \mathbf{d}_k)\right\rangle_+, \tag{1.10a}$$

$$\mathbf{d}^*_{k+1} = \min\{1, \mathbf{d}_k + \Delta t \mathbf{r}_k\}, \tag{1.10b}$$

$$\overline{\mathbf{M}}\mathbf{r}^*_{k+1} = \left\langle \mathbf{Y}^-(\mathbf{u}_{k+1}, \mathbf{d}^*_{k+1})\right\rangle_+, \tag{1.10c}$$

$$\mathbf{d}_{k+1} = \min\left\{1, \mathbf{d}_k + \frac{1}{2}\Delta t\left(\mathbf{r}_k + \mathbf{r}^*_{k+1}\right)\right\}, \tag{1.10d}$$

where the minimum operation is taken componentwise.

1.3.3 Time step adaptivity due to stability considerations

As is typical of explicit methods, the proposed combination of methods is conditionally stable. As will be shown later, the critical time step decreases as the crack propagation progresses, and hence an adaptive scheme to determine a stable time step is necessary, which we will elaborate in this section as well.

To study the stability properties, we only consider the case with $\mathrm{d}_A < 1$, $\forall A \in H$, and rewrite (1.8e) as

$$\mathbf{Y}(\mathbf{u},\mathbf{d}) = \overline{\mathbf{F}} - \overline{\mathbf{K}}(\mathbf{u})\mathbf{d},$$

where

$$\overline{K}_{AB}(\mathbf{u}) := g_c l^2(\nabla N_A, \nabla N_B) + \left(N_A, N_B\left[\frac{g_c}{l} + 2\psi_+[\boldsymbol{\varepsilon}(\boldsymbol{u})]\right]\right),$$
$$\overline{F}_A := (N_A, 2\psi_+[\boldsymbol{\varepsilon}(\boldsymbol{u})]).$$

Let $\lambda_{\max}(\mathbf{u},\mathbf{d})$ and $\overline{\lambda}_{\max}(\mathbf{u})$ be the maximum eigenvalues of the following generalized eigenvalue problems, respectively:

$$\mathbf{K}(\mathbf{u},\mathbf{d})\boldsymbol{\Psi} = \lambda\mathbf{M}\boldsymbol{\Psi}, \quad \overline{\mathbf{K}}(\mathbf{u})\overline{\boldsymbol{\Psi}} = \overline{\lambda}\,\overline{\mathbf{M}}\,\overline{\boldsymbol{\Psi}}.$$

Then standard stability analysis of the central difference method and Heun's method yields the following stability limit of the time step:

$$\Delta t_c = \min\{\Delta t_{cu}, \Delta t_{cd}\}, \quad \Delta t_{cu} := \frac{2}{\sqrt{\lambda_{\max}(\mathbf{u},\mathbf{d})}}, \quad \Delta t_{cd} := \frac{2}{\overline{\lambda}_{\max}(\mathbf{u})}. \tag{1.11}$$

Since the phase field d is non-decreasing, it can be shown that $\lambda_{\max}(\mathbf{u},\mathbf{d}) \leq \lambda_{\max}(\mathbf{u},\mathbf{d}_0)$, where $\mathbf{d}_0$ is the initial condition for $\mathbf{d}$, and hence a safe estimate for Δt_{cu} is given by

$$\Delta t_{cu} \leq \frac{2}{\sqrt{\lambda_{\max}(\mathbf{u},\mathbf{d}_0)}}.$$

On the other hand, $\overline{\lambda}_{\max}(\mathbf{u})$ is approximately an affine function of $\psi_+[\boldsymbol{\varepsilon}(\boldsymbol{u})]$, and thus if the loading to the system increases, Δt_{cd} decreases. As a result, an adaptive procedure to always make sure that the calculation is within the stability range is indispensable. For this purpose, we take advantage of the GPU architecture and compute a lower bound for the critical time step by determining an upper bound for the maximum eigenvalue of $\overline{\mathbf{M}}^{-1}\overline{\mathbf{K}}$ at every time step.

Let e be a typical element and $\overline{\mathbf{k}}^e(\mathbf{u})$ and $\overline{\mathbf{m}}^e$ the element contribution to $\overline{\mathbf{K}}(\mathbf{u})$

and $\overline{\mathbf{M}}$, respectively. Then from a property of the Rayleigh quotient and Gershgorin's circle theorem Gerschgorin [1931], an upper bound for $\overline{\lambda}_{\max}(\mathbf{u})$ is given by

$$\overline{\lambda}_{\max}(\mathbf{u}) \leq \max_e \overline{\lambda}^e_{\max}(\mathbf{u}) \leq \max_e \max_i \sum_j \left|\overline{A}^e_{ij}(\mathbf{u})\right|, \tag{1.12}$$

where $\overline{\lambda}^e_{\max}(\mathbf{u})$ is the maximum eigenvalue of the generalized eigenvalue problem associated with $\overline{\mathbf{k}}^e(\mathbf{u})$ and $\overline{\mathbf{m}}^e$, and $\overline{A}^e_{ij}$ is the i, j component of $\overline{\mathbf{A}}^e(\mathbf{u}) := (\overline{\mathbf{m}}^e)^{-1}\overline{\mathbf{k}}^e(\mathbf{u})$. As a result, we define the following computable lower bound:

$$\Delta t^{LB}_{cd} := \min_e \Delta t^e_{cd}, \quad \Delta t^e_{cd} := \frac{2}{\max_i \sum_j \left|\overline{A}^e_{ij}(\mathbf{u})\right|}. \tag{1.13a}$$

A lower bound for $\Delta t_{cu}(\mathbf{u}, \mathbf{d}_0)$, Δt^{LB}_{cu}, can also be obtained in a similar way:

$$\Delta t^{LB}_{cu} := \min_e \Delta t^e_{cu}, \quad \Delta t^e_{cu} := \frac{2}{\sqrt{\max_i \sum_j \left|A^e_{ij}(\mathbf{u}, \mathbf{d}_0)\right|}}, \tag{1.13b}$$

where $\mathbf{A}^e(\mathbf{u}, \mathbf{d}_0) := (\mathbf{m}^e)^{-1}\mathbf{k}^e(\mathbf{u}, \mathbf{d}_0)$. Nevertheless, Δt^{LB}_{cu} only needs to be computed once at the beginning. Then a lower bound of Δt_c can be obtained by

$$\Delta t^{LB}_c = \min\left\{\Delta t^{LB}_{cu}, \Delta t^{LB}_{cd}\right\}. \tag{1.14}$$

Chapter 2

GPU implementation

2.1 Essence of massively parallel programming

Despite the growing literature in the application of phase field modeling of fracture, the computational cost of such models is sensitive to the choice of the regularization parameter in conjunction with the mesh size, as the mesh has to be fine enough to resolve high gradients of the phase field appearing in the transition zones between cracked and un-cracked materials, whose width is on the order of the regularization parameter [Miehe et al., 2010b]. This parameter can be interpreted as either a mathematical construction or an intrinsic material parameter. In this work we will adopt the first meaning in subsequent sections. Strictly speaking, even in the quasi-static setting, the limit of gradient flows for similar Ambrosio-Tortorelli functionals as this regularization parameter tends to 0 is still a difficult unresolved problem [Feng and Prohl, 2004, Babadjian and Millot, 2014]. The general understanding is that it has to be small enough compared to the characteristic length of the computational domain to make the simulation meaningful. Numerically resolving this regularization length scale is one of the main computational challenges of implementing such models.

Recently, graphics processing units (GPUs) which are capable of massive parallelization have had success in accelerating many numerical computations. Therefore, massively parallel programming with the GPU is a promising solution for the problem of high computational costs of the phase field methods. Here we mention Cecka *et* *al.* [Cecka et al., 2011] as one of the early applications of the GPU to finite element methods.

In this work, we develop a massively parallel program to implement our explicit algorithm to solve for a rate-dependent formulation of the phase field modeling of brittle fracture on NVIDIA GPUs using the Compute Unified Device Architecture (CUDA).

Essentially the GPU favors the same or similar operations on a massive collection of data. In this context, the updating of nodal finite element data (displacement, velocity, and phase field) naturally fits this requirement. In this work we will provide a detailed algorithm to demonstrate how such a simulation can be done with GPU.

This chapter devotes to the presentation of our GPU algorithm for the discrete problem described in Section 1.3. We first give an overview of the GPU architecture in Section 2.2, focusing on the necessary trade-offs, such as storing versus repeatedly computing data, in implementing the problem at hand. Then in Section 2.3 we describe the algorithm used to implement the discrete form of the phase field problem suited to the GPU architecture.

2.2 Introduction to GPU programming

We first briefly introduce the GPU architecture and programming. As we will see, the most suitable type of procedure for the GPU is one that executes the same instructions over a large set of data. For the problem at hand, the updating of nodal data (displacement, velocity, and phase field) between time steps is a suitable procedure to be GPU-parallelized, since for each node, the needed data is associated with only its neighboring nodes and elements. Another procedure to be parallelized is the determination of a lower bound for the critical time step by evaluating the right hand side of (1.14), a procedure that allows parallelizing the calculation between elements.

We have adopted the Compute Unified Device Architecture (CUDA) designed by the company NVIDIA to organize the program. The main idea of this architecture is to run the backbone of the program as a serial code on the central processing unit

(CPU) while calling the GPU device as a kernel (or subroutine) to run the parallel procedures.

For readers' convenience, we have summarized the most frequently used terms regarding the GPU architecture in Table 2.1.

Table 2.1: Glossary in the GPU architecture

Term	Definition
Host	The CPU
Device	The GPU
Kernel	A function executed in parallel on the device
Streaming multiprocessors (SMs)	The part of the GPU that runs the CUDA kernels
Thread	A process on the device executing the kernel. Only a certain number (typically 1,536) of threads can fit in a SM.
Block	A group of threads that cooperates through their shared memory and that can be synchronized. Only a certain number (typically 8) of blocks can fit in a SM.
Grid	The entire set of blocks executing a single kernel
Registers	On-chip memory assigned to a single thread
Shared memory	On-chip memory shared among threads of a single block
Global memory	Off-chip memory accessible to all threads

We next briefly recapitulate key concepts relevant to the algorithm design of the problem at hand.

Execution model Whenever a kernel is invoked, the device is executed by generating a large number of threads. Each thread is able to complete simple tasks such as elementary operations and data transfer. When all threads have completed their task, the execution of the code is transferred back to the host, until either another kernel is called or the program terminates. Note that there is no predictable order among the threads in execution; hence, attention must be paid to avoid the race condition, an undesired situation in which different execution orders of the threads lead to different outputs.

All threads of the GPU device are grouped into blocks. A block defines the extent to which data can be exchanged and to which synchronization among threads is possible during the course of the kernel execution, which becomes clearer with the memory hierarchy illustrated.

Memory hierarchy There are a few types of memory in the GPU architecture whose characteristics the programmer must take into account when organizing a kernel. We compare the major types of memory in Table 2.2.

Table 2.2: Memory hierarchy

	Global memory	Shared memory	Registers
Amount	Abundant	Limited	Very limited
Range of accessibility	All threads	Blockwise	Thread
Speed of data transfer	Slow	Fast	Fast
Data remains between invocations of kernels	Yes	No	No

Figure 2.1 offers an overview of the CUDA device memory model. There are also other types of memory in the GPU device, namely constant memory and texture memory, which we will not elaborate in this work.

Optimizing data transfer and repetition of calculation To optimize the parallel device code, kernels should minimize transactions to and from the global memory, as well as the amount of registers and shared memory usage so that the kernel is executed by the maximum allowed number, or a smaller number but as close as possible, of blocks per SM (typically 8) and the maximum allowed number of threads per SM (typically 1,536) in the device. This includes: (a) transferring only the minimum amount of data from the global memory to shared memory and registers, and (b) calculating intermediate results as often as needed. In the next section, we describe how we deal with such an optimization problem when implementing our numerical algorithm.

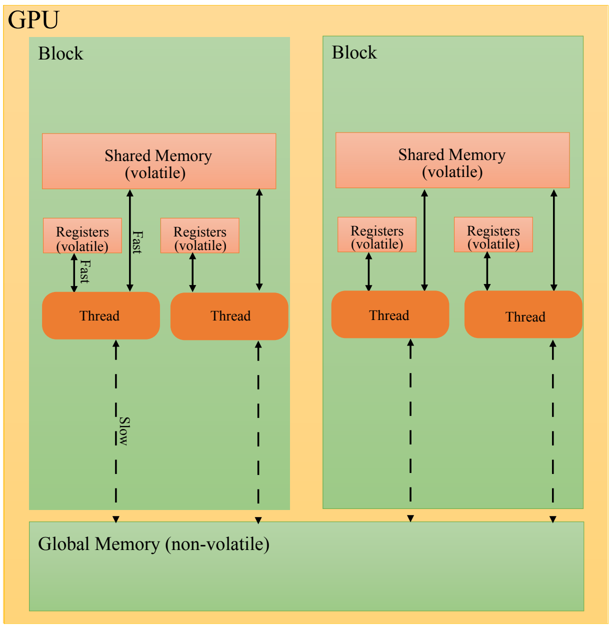


Figure 2.1: Thread and memory hierarchy of the GPU (reproduced and revised from Kirk and Hwu [Kirk and Hwu, 2010, Figure 3.7], with permission from the authors). When the GPU kernel is called, data is transferred from the CPU to the global memory of the GPU, then to the shared memory and registers. The global memory is non-volatile, i.e., its contents remain between invocations of the GPU; the shared memory and the registers are volatile. When the GPU is being executed, thousands of threads are generated to simultaneously complete simple tasks on different data. These are grouped into blocks. Each thread owns some memory called registers, and shares some memory with other threads in the same block called shared memory. Threads in different blocks cannot share memory during the course of kernel execution nor can they be synchronized.

2.3 GPU algorithm for the phase field problem

Now, we describe how we implement the numerical algorithm detailed in Section 1.3 in a massively parallel way.

Thread assignment Judging from the nature of the data structure of the problem at hand, it is natural to associate threads with either elements or nodes, for different parts of the calculation. In particular, we associate the threads with the *nodes* for most of the calculations during the time step except when calculating a lower bound for the critical time step we associate threads with *elements*. The reason for the latter is apparent as long as (1.14) is concerned, and that for the former is because the FEM DOFs are associated with the nodes: associating the threads with the nodes easily avoids race conditions. We do need to pay the price in that data for different nodes of the same element is repeatedly computed, by different threads.

Block division As explained in Section 2.2, all threads are grouped into blocks. Here we explain how we do so and its implications when the threads are associated with the nodes. The case of assigning threads to elements is analogous and simpler, and hence will not be elaborated.

We first divide the computational domain into mutually disjoint subdomains, then group all nodes within the same subdomain into a block. In general, each block should have almost the same number of nodes in order to balance work loads. This balance can be achieved with the package Metis [Karypis and Kumar, 1998]. Here for simplicity we perform the subdivision with a rectangular grid, as shown in Figure 2.2. Note that during the calculation, in order to update quantities associated with all nodes of a certain block, it is also necessary to read (but not write) data associated with nodes in other blocks sharing an element with any node of the block of interest.

Code body Algorithm 2 details the body of the program, including codes executed in the host (CPU) and in the device (GPU). During each time step, multiple kernels are invoked to complete the tasks in Algorithm 1. In Table 2.3 we list the input and

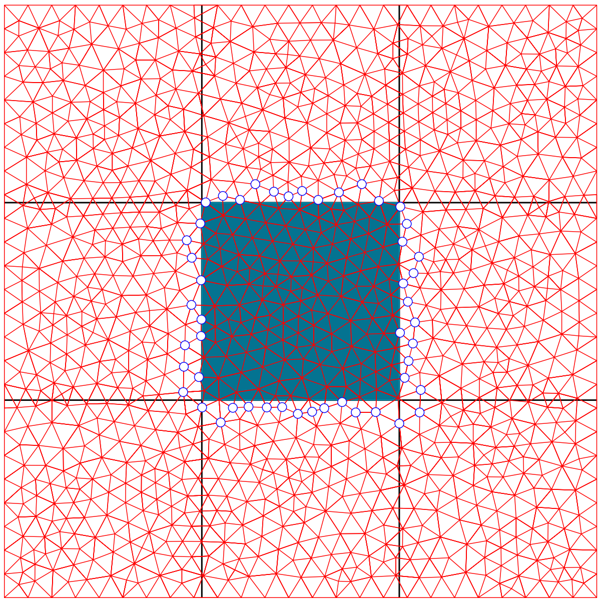


Figure 2.2: Illustrative example of a block division by subdividing a domain into nine subdomains. Threads associated with nodes in the same subdomain are grouped into the same block. Each block is responsible of updating data associated with all nodes inside the corresponding subdomain in each time step, and hence needs data of the current time step of all associated nodes plus that of the neighboring nodes of these nodes. For example, the block associated with the square subdomain in the center needs data from all nodes marked with “o” as well as those from its own nodes, nodes located in the dark blue region.

output for each of these kernels. All these kernels are categorized into four types:

(I) to compute the allowed Δt based on (1.11);

(II) to calculate the intermediate nodal field values ($\mathbf{a}$, $\mathbf{r}$, or $\mathbf{r}^*$);

(III) to update $\mathbf{u}$, $\mathbf{v}$, $\mathbf{d}$, or $\mathbf{d}^*$;

(IV) to update boundary conditions for $\mathbf{u}$ and $\mathbf{v}$.

The reason to have multiple kernels is because even within a time step, computing some intermediate results still requires data from neighboring blocks. Here, we briefly explain each type of the kernels.

Type *I* kernels. Kernels I_i and I_{ii} compute $\Delta t_{cu}^{LB,bk}$ and $\Delta t_{cd}^{LB,bk}$ for each block, respectively, which is elaborated in Algorithm 3. To find Δt_{cu}^{LB} and Δt_{cd}^{LB}, we employ a serial code in the host rather than in the device, since there are a manageable number of blocks (up to 1,024) in the numerical examples to come. It is worthwhile to mention that regarding Algorithm 3, in order to find the minimum value inside each block, a well-known reduction algorithm of logarithmic complexity is utilized. Here, for brevity we refer the readers to Kirk and Hwu [Kirk and Hwu, 2010, Chapter 6].

Type *II* kernels. Type *II* kernels calculate either $\mathbf{a}$ or $\mathbf{r}$, the time rate of change of $\mathbf{d}$. Algorithm 4 illustrates how these kernels work.

Type *III* kernels. Type *III* kernels update nodal values of $\mathbf{u}$, $\mathbf{v}$, or $\mathbf{d}$. Here Algorithm 5 details this process.

Note that type *III* kernels are executed by as many threads as the number of input values so that each thread is responsible for updating one value.

Type *IV* kernels. After updating $\mathbf{u}$ or $\mathbf{v}$, type *IV* kernels are invoked to impose the Dirichlet boundary conditions by as many threads as the number of boundary nodes. To speed up the process, we collected the addresses of the nodal values of $\mathbf{u}$ or

Algorithm 2: GPU implementation of Algorithm 1. Subscript f refers to the values at the final time.

```
input  : $\mathbf{u}_0$, $\mathbf{v}_0$, $\mathbf{d}_0$, and $t_f$
output: $\mathbf{u}_f$, $\mathbf{v}_f$, and $\mathbf{d}_f$
/* Kernels of types $I$, $II$, and $III$ are discussed in Algorithms 3,
   4, and 5, respectively.                                          */
Call kernel $I_i$ to compute $\Delta t_{cu}^{LB,bk}$ for each block according to (1.11)
In the host, compute $\Delta t_{cu}^{LB}$ as the minimum among all $\Delta t_{cu}^{LB,bk}$
/* $\Delta t_{cu}^{LB}$ only has to be computed once                  */
Set $t = 0$ and $k = 0$
Call kernel $II_i$ to compute $\mathbf{r}_0$
Call kernel $II_{ii}$ to compute $\mathbf{a}_0$
while $t < t_f$ do
    Call kernel $I_{ii}$ to update $\Delta t_{cd}^{LB,bk}$ for each block according to (1.11)
    In the host, compute $\Delta t_{cd}^{LB}$ as the minimum among all $\Delta t_{cd}^{LB,bk}$
    Set $\Delta t_c^{LB} = \min\{\Delta t_{cu}^{LB}, \Delta t_{cd}^{LB}\}$ per (1.14) and $\Delta t = \min\{\Delta t_c^{LB}, t_f - t\}$
    Call kernel $III_i$ to update $\mathbf{u}_{k+1}$, $\mathbf{v}_{k+1}^*$, and $\mathbf{d}_{k+1}^*$ with (1.10b), (1.9b), and
    (1.9c)
    Call kernel $IV_i$ to set boundary conditions for $\mathbf{u}$
    Call kernel $II_{ii}$ to compute $\mathbf{r}_{k+1}^*$ with (1.10c)
    Call kernel $III_{ii}$ to update $\mathbf{d}_{k+1}$ with (1.10d)
    Call kernel $II_{ii}$ to compute $\mathbf{r}_{k+1}$ with (1.10a)
    Call kernel $II_i$ to compute $\mathbf{a}_{k+1}$ with (1.9d)
    Call kernel $III_{iii}$ to update $\mathbf{v}_{k+1}$ with (1.9e)
    Call kernel $IV_{ii}$ to set boundary conditions for $\mathbf{v}$
    $t \leftarrow t + \Delta t$
    $k \leftarrow k + 1$
/* Vectors in the same of the following groups can, and should,
   occupy the same memory space: ($\mathbf{u}_k$, $\mathbf{u}_{k+1}$), ($\mathbf{v}_k$, $\mathbf{v}_{k+1}^*$, $\mathbf{v}_{k+1}$),
   ($\mathbf{a}_k$, $\mathbf{a}_{k+1}$), ($\mathbf{d}_k$, $\mathbf{d}_{k+1}$), and ($\mathbf{r}_k$, $\mathbf{r}_{k+1}$).           */
```

Table 2.3: Summary of input and output of the kernels. Here bk means blockwise quantities.

Kernel	Input	Output	Remarks
I_i	–	$\Delta t_{cu}^{LB,bk}$	(1.13b)
I_{ii}	$\mathbf{u}$	$\Delta t_{cd}^{LB,bk}$	(1.13a)
II_i	$\mathbf{u}$, $\mathbf{d}$	$\mathbf{a}$	(1.9a), (1.9d)
II_{ii}	$\mathbf{u}$, $\mathbf{d}$	$\mathbf{r}$	(1.10a), (1.10c)
III_i	$\mathbf{u}_k$, $\mathbf{v}_k$, $\mathbf{d}_k$	$\mathbf{u}_{k+1}$, $\mathbf{v}^*_{k+1}$, $\mathbf{d}^*_{k+1}$	(1.9b), (1.9c), (1.10b)
III_{ii}	$\mathbf{d}^*_{k+1}$	$\mathbf{d}_{k+1}$	(1.10d)
III_{iii}	$\mathbf{v}^*_{k+1}$	$\mathbf{v}_{k+1}$	(1.9e)
IV_i	$\mathbf{u}$	$\mathbf{u}$ (with boundary conditions imposed)	
IV_{ii}	$\mathbf{v}$	$\mathbf{v}$ (with boundary conditions imposed)	

Algorithm 3: GPU implementation of type I kernels based on (1.14)

```
input  : u (only for kernel $I_{ii}$)
output: $\Delta t_{cu}^{LB,bk}$ or $\Delta t_{cd}^{LB,bk}$ (where bk stands for block)
Transfer the following data to the shared memory: local connectivity, nodal
coordinates, and for kernel $I_{ii}$, also nodal values of u
Compute $\Delta t^e_{cu}$ (or $\Delta t^e_{cd}$) for each element (associated with one thread) per
(1.13b) (or (1.13a)) and store it in the registers (1.11)
Synchronize all blocks (barrier synchronization)
Find the minimum $\Delta t_{cu}^{LB,bk} = \min_e \Delta t^e_{cu}$ (or similarly for $\Delta t_{cd}^{LB,bk}$) among all the
threads of a block (1.12)
Transfer $\Delta t_{cu}^{LB,bk}$ (or $\Delta t_{cd}^{LB,bk}$) to the global memory
```

Algorithm 4: Algorithm for the GPU implementation of type II kernels based on Algorithm 2

```
input  : u and d
output: a or r
Transfer the following data to the shared memory: local connectivity, nodal
coordinates, and nodal values of u and d
Compute a (or r) for each node (associated with one thread) with (1.9a) and
(1.9d) [or (1.10a) and (1.10c)] by looping over its neighboring elements and
summing up their contribution in the registers
Transfer the registers to the global memory
```

Algorithm 5: Algorithm for the GPU implementation of kernel III based on Algorithm 2

```
input  : $\mathbf{u}_k$, $\mathbf{v}_k$, $\mathbf{v}^*_{k+1}$, $\mathbf{d}_k$, $\mathbf{d}^*_{k+1}$
output: $\mathbf{u}_{k+1}$, $\mathbf{v}^*_{k+1}$, $\mathbf{v}_{k+1}$, $\mathbf{d}^*_{k+1}$, $\mathbf{d}_{k+1}$
Call the required intermediate nodal field values from global memory
Do arithmetic operations based on formulas (1.9b), (1.9c), (1.9e), (1.10b), (1.10d)
Assign the result at the same location of input at global memory
```

$\mathbf{v}$ and stored them consecutively in the global memory in the pre-processing stage. A type IV kernel is then constructed such that each thread is responsible of updating the boundary value of one node. When such a kernel is invoked, each thread first reads from the global memory the pre-collected address corresponding to this thread and then updates the values with the relevant pre-defined boundary values. This way of pre-processing and arrangement of intermediate data leads to speeding up the global memory transactions.

Following Algorithm 2, with details just explained, a massively parallel program has been constructed on the GPU architecture to solve the phase field formulation for crack propagation presented in this work.

Chapter 3

Numerical examples

In this chapter, we present three representative sets of numerical examples to demonstrate the performance of the proposed algorithm for the phase field fracture models. An NVIDIA Kepler K20M GPU and two Intel Xeon E5-2670 (8 Cores, 2.6GHz, 20MB Cache, 8.0GT) CPUs were used in the calculations.

The three sets of examples include a tension test, a shear test, and a symmetric bending test. Throughout this section we will use the standard P_1 elements and lumped mass matrices $\mathbf{M}$ and $\overline{\mathbf{M}}$. We will adopt the values of the parameters given in Table 3.1.

Table 3.1: Default parameter values for the examples

Name	Symbol	Value
Lamé constant	λ	120GPa
Shear modulus	μ	80GPa
Critical energy release rate	g_c	2.7×10^{-3}kN/mm
Density	ρ	7000 kg/m^3
Artificial viscosity	η	1.0×10^{-6}kNs/mm^2
Displacement loading rate (lower)	$\dot{u}_D = \dot{u}_{DL}$	10mm/s
Displacement loading rate (higher)	$\dot{u}_D = \dot{u}_{DH}$	100mm/s

3.1 Cracked square plate under a tension test

We first investigate a square plate with a horizontal initial crack at the middle height starting from the left end and ending at the plate center. This square plate has edge lengths of $L = 100$mm. The geometric setup is depicted in Figure 3.1. The specimen is under a direct tension test, in which a monotonically increasing displacement with magnitude $\dot{u}_D t$ is imposed on both the top and the bottom edges. As in Table 3.1, we will test for two loading rates $\dot{u}_{DL}$ and $\dot{u}_{DH}$, in Sections 3.1.1 and 3.1.2, respectively. The regularization parameter is set to be $l = 1$mm. We simulate the evolution from $t = 0$ to some final time t_f ($t_f = 7.0 \times 10^{-3}$s for $\dot{u}_D = \dot{u}_{DL}$ and $t_f = 2 \times 10^{-3}$s for $\dot{u}_D = \dot{u}_{DH}$). In the simulations, the sample is discretized uniformly with 45,616 three-noded triangular elements. In the GPU kernel the nodes or elements are grouped into 16×16 blocks.

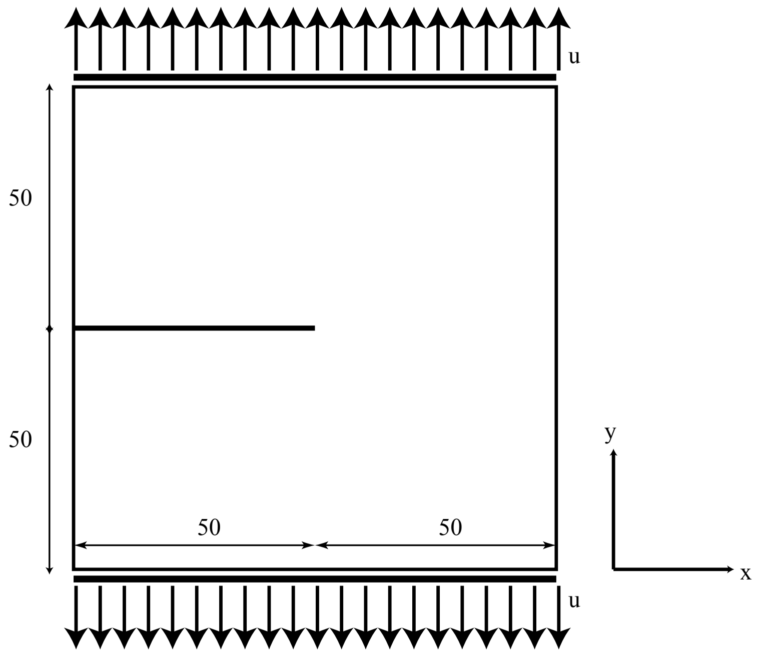


Figure 3.1: Schematic of a cracked square plate (unit: mm) under a single-edge-notched tension test. A monotonically increasing displacement loading $\boldsymbol{u} = \dot{u}_D t \boldsymbol{e}_y$ is applied on the top edge and $\boldsymbol{u} = -\dot{u}_D t \boldsymbol{e}_y$ on the bottom edge.

3.1.1 A lower loading rate

Figure 3.2 shows snapshots of the phase field contours obtained with models A, B, and C (see Table 1.2) at selected instants $\{0.5, 0.75, 1.0\}t_f$, using the default parameters in Table 3.1 with $\dot{u}_D = \dot{u}_{DL}$. It is observed that all models produce a similar crack path up to the fully broken stage. The crack topologies of models A and C agree well with the results in [Miehe et al., 2010b].

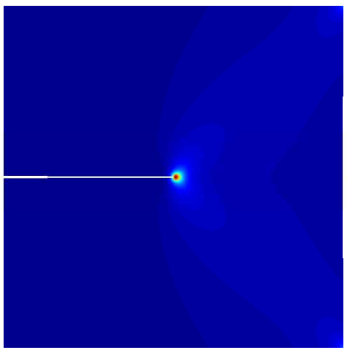
Figure 3.3 plots the evolution of the total vertical reaction applied at the sample's top (or bottom) edge for the three models. All models show similar trends, though model B gives rise to slightly higher values. After the crack starts to evolve, approximately at $t = 0.58t_f$, the total reaction begins to decrease.

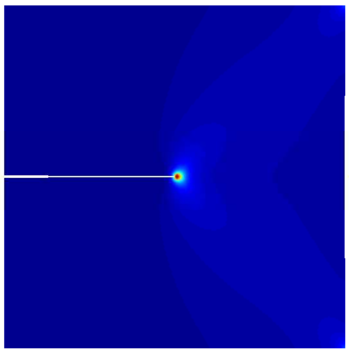
Figure 3.4 shows the evolution of the *effective total crack length*, denoted $|\Gamma|$, for models A, B, and C. According to (1.3), $|\Gamma|$ is defined as

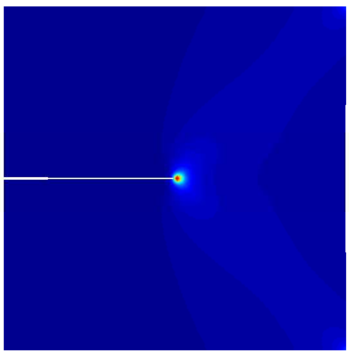
$$|\Gamma| = \text{initial crack length} + \frac{1}{2}\int_{\Omega}\left(\frac{d^2}{l} + l\nabla d\cdot\nabla d\right)\, d\Omega, \tag{3.1}$$

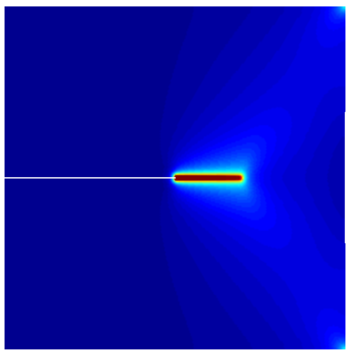
which takes into account both the initial and propagated cracks. It can be observed that the effective crack length increases until a certain time/value, after which it shows a plateau, indicating that the crack has completely propagated through the sample.

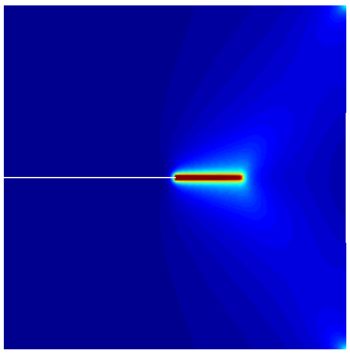
Here, the final values for the total crack lengths are larger than L. This may be due to the irreversibility of the phase field model (1.4b'). Unlike the method used in [Miehe et al., 2010b] and [Borden et al., 2012] where the computed phase field is always the solution to an elliptic problem similar to (1.4b), i.e., without taking into account its history, our solution to the phase field strictly enforces that $\dot{d} \geq 0$ everywhere and hence tends to smear the phase field profile near where $d = 1$, leading to a higher effective crack length. On the other hand, numerical experiments show that refining the finite element mesh helps reducing this broadening effect. According to [Bourdin et al., 2008], the effect of the mesh size h and the regularization parameter l on the effective critical energy release rate $g_{c,eff}$ is summarized by $g_{c,eff} = g_c[1 + h/(4l)]$, which implies that the computed crack length should be *smaller* than the exact one at the same load, to the contrary of the observation here. Hence, the apparent higher

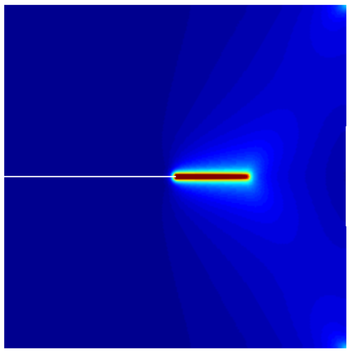

(a) $0.5t_f$

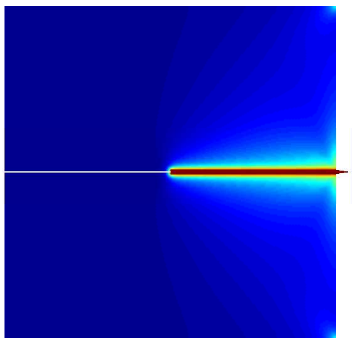

(b) $0.5t_f$

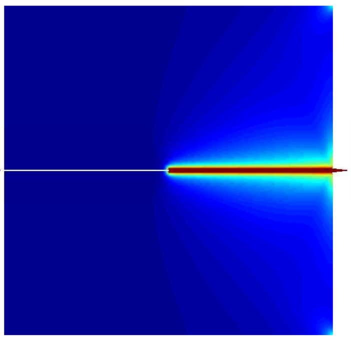

(c) $0.5t_f$

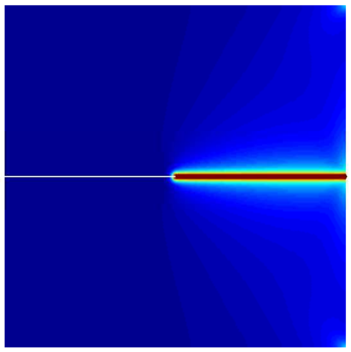

(d) $0.75t_f$

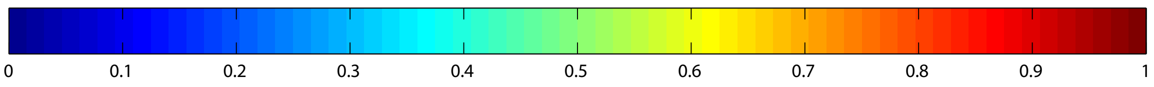

(e) $0.75t_f$


(f) $0.75t_f$


(g) t_f


(h) t_f


(i) t_f



(j)

Figure 3.2: Cracked square plate under a tension test with $\dot{u}_D = \dot{u}_{DL}$. Here $t_f = 7.0 \times 10^{-3}$s. Phase field contours of models A (left column), B (middle column), and C (right column) (see Table 1.2) at different instants are shown in deformed configurations with the displacement scaled. The same parameters are used for all three phase field models. Due to symmetry we modeled only the upper half of each sample, which is discretized into 45,616 three-noded triangular elements. The initial crack was explicitly introduced, and hence in the deformed configuration it appears as a white line. We observe a straight crack in each case.

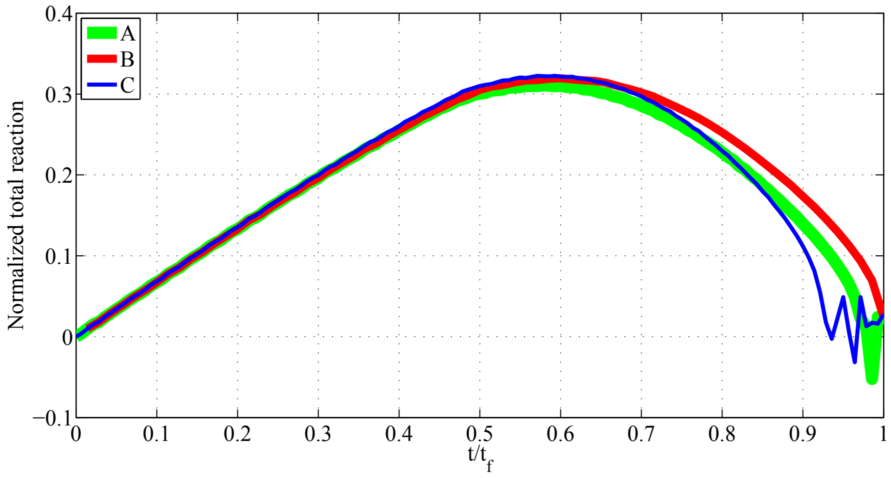


Figure 3.3: Cracked square plate under a tension test with $\dot{u}_D = \dot{u}_{DL}$. Total vertical reaction on the top edge versus time for models A, B, and C. The total reaction is normalized by $2E\dot{u}_D t_f/L \cdot L = 29.12$kN/mm, i.e., by the total reaction applied to the top edge at $t = t_f$ in the case without any crack or phase field evolution. Here $t_f = 7.0 \times 10^{-3}$s. All models give rise to similar trends for the total reaction at the top. The total reaction begins to decrease at approximately $t = 0.58t_f$.

crack length may be due to the inadequacy of the irreversibility scheme adopted for the Ambrosio-Tortorelli-type functional here.

Figure 3.5 plots the calculated lower bound for the critical time step Δt_c^{LB} computed with (1.14) as a function of t. Here, we observe, as expected, that as the phase field evolves, beginning from a certain time, Δt_c^{LB} completely depends on Δt_{cd}^{LB}, a lower bound for the critical time step for the phase field half-problem. Hence at the fully broken stage the time step in use is significantly smaller (up to 200 times) than its initial value. We reiterate here that in the early stage of the crack propagation, the critical time step Δt_c^{LB} is dictated by that of the elastodynamic half problem, Δt_{cu}^{LB}, which *increases* as the crack propagates. Hence, instead of tracking this increasing critical time step with adaptivity, we simply use the one determined from the first time step. In contrast, in the late stage, i.e., when $\Delta t_{cd}^{LB} \leq \Delta t_{cu}^{LB}$ (beyond $0.9t_f$ in this case), then $\Delta t_c^{LB} = \Delta t_{cd}^{LB}$ decreases as the crack propagates, which is when we start to adaptively refine the time step on the fly.

We next perform a study on the artificial viscosity η on the solution. Figure 3.6

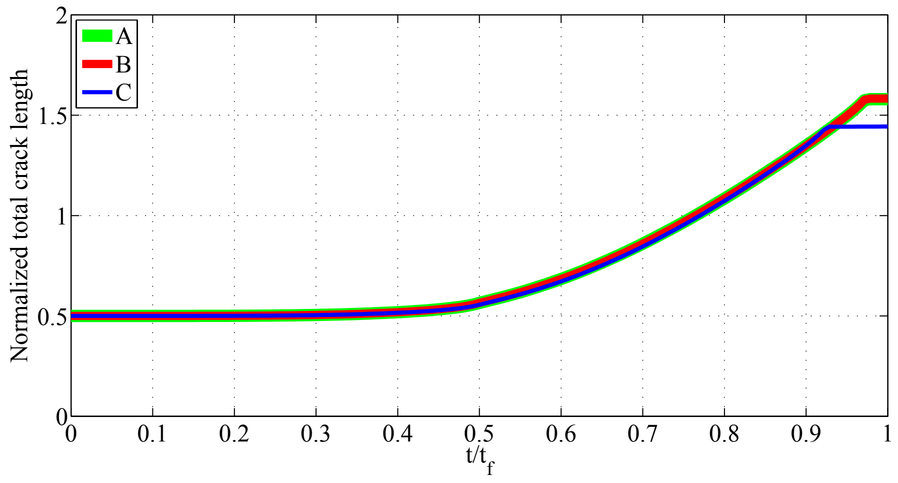


Figure 3.4: Cracked square plate under a tension test with $\dot{u}_D = \dot{u}_{DL}$. Evolution of the total crack length as defined in (3.1) normalized by the edge length L for models A, B, and C. Note that the initial crack length is $0.5L$. Here $t_f = 7.0 \times 10^{-3}$s. It is observed that the final crack lengths are larger than L, which might come from the irreversibility of the phase field evolution.

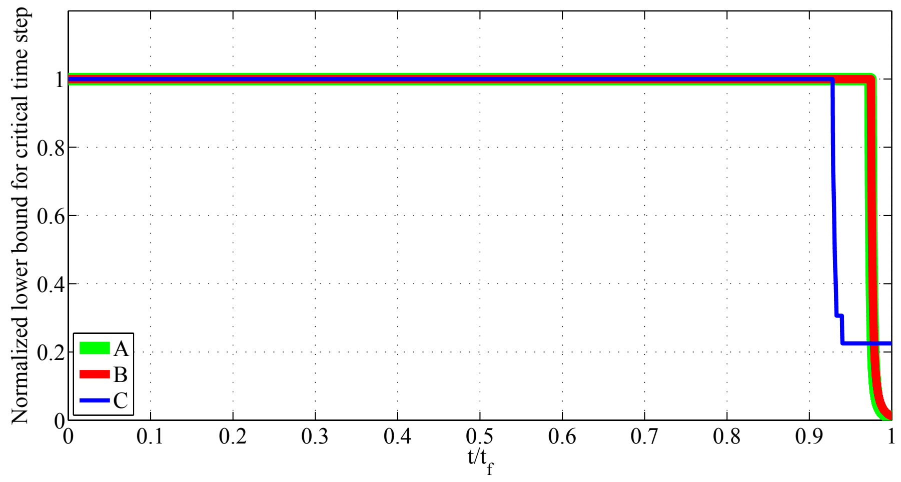


Figure 3.5: Cracked square plate under a tension test with $\dot{u}_D = \dot{u}_{DL}$. A lower bound for the critical time step is calculated with (1.14) and normalized with Δt_{cu}^{LB} at $t = 0$, which is the same for all models. Here, the data are collected from the solution of models A, B, and C. Beginning from a certain point, as the phase field evolves, the lower bound decreases and Δt_c^{LB} solely depends on Δt_{cd}^{LB}. Here $t_f = 7.0 \times 10^{-3}$s. The reader is referred to the body text for an explanation of this trend.

shows three phase field contours computed with three different η values at the same time instant, with all other parameters the same. The results are anticipated: all samples show the same crack path, while as η increases, the evolution of the crack becomes slower.

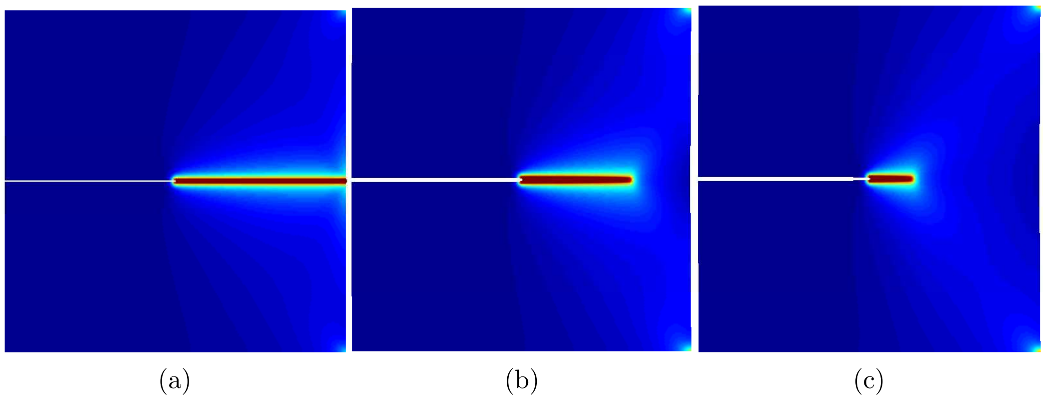


Figure 3.6: Cracked square plate under a tension test with $\dot{u}_D = \dot{u}_{DL}$. Phase field contours for three samples of model C at $t = t_f$ are shown. Here $t_f = 7.0 \times 10^{-3}$s. The values of η for these samples are: (a) 1.0×10^{-6}kNs/mm^2, (b) 2.0×10^{-6}kNs/mm^2, and (c) 5.0×10^{-6}kNs/mm^2. Other parameters are the same and are given in Table 3.1. As seen, for higher value of η, the evolution of crack at $t = t_f$ becomes slower.

Figure 3.7 depicts their corresponding evolution of the total vertical reaction at the top. As shown, as η increases, higher values for the total reaction is obtained, and the softening becomes less pronounced. It is also seen that for all samples the softening starts at approximately the same time, from which we may conclude that the artificial viscosity η does not affect the onset of crack propagation.

3.1.2 A higher loading rate

In this section, we push the loading rates high enough to see crack branching, i.e., $\dot{u}_D = \dot{u}_{DH}$. For other parameters we use the same values as in Table 3.1. We simulate the evolution from $t = 0$ to some final time $t_f = 2 \times 10^{-3}$s. Figure 3.8 is the counterpart of Figure 3.2 for this loading rate, where snapshots of the phase field contours obtained with all models are shown at different instants. We observe crack

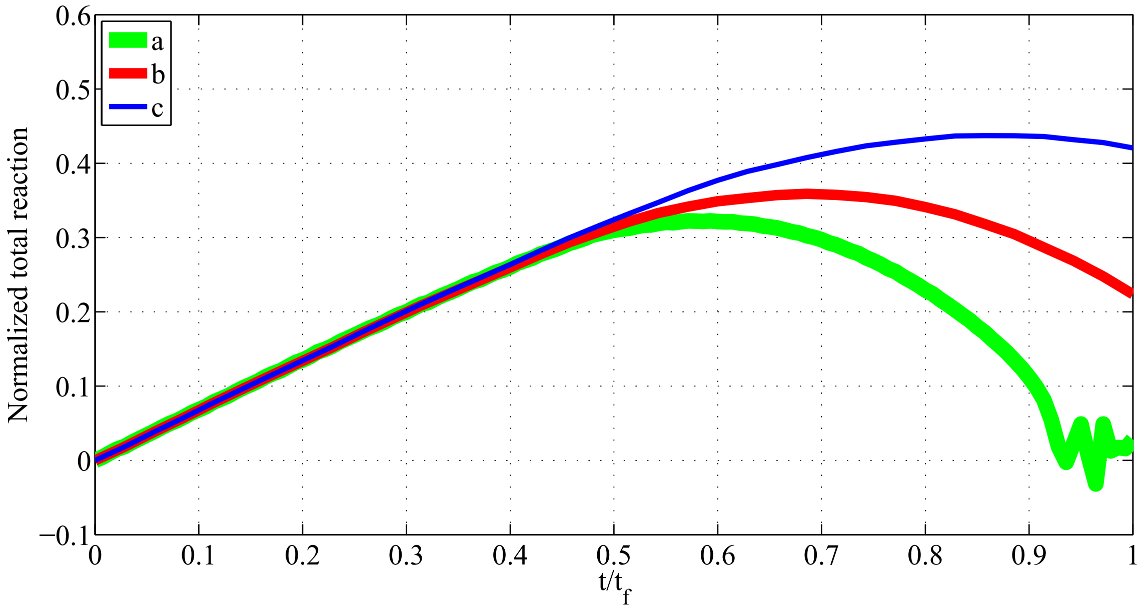
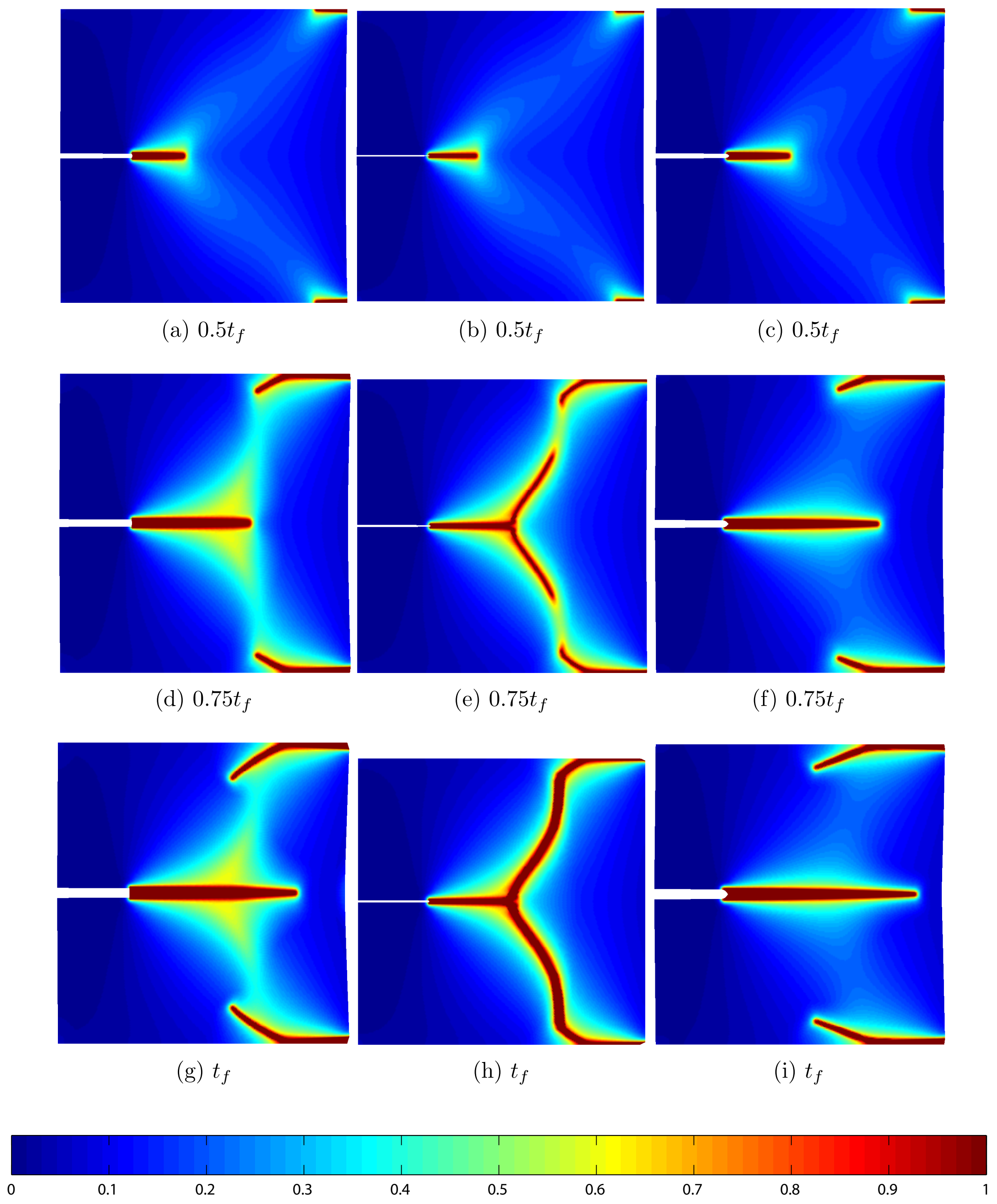


Figure 3.7: Cracked square plate under a tension test with $\dot{u}_D = \dot{u}_{DL}$. The evolution of the total vertical reaction on the top for edge three samples of model C with different values of η: (a) 1.0×10^{-6}kNs/mm^2, (b) 2.0×10^{-6}kNs/mm^2, and (c) 5.0×10^{-6}kNs/mm^2. Here $t_f = 7.0 \times 10^{-3}$s. It is seen that the higher η is, the higher total reaction is. It can also be observed that η does not affect the onset of the onset of crack propagation.

branching only for model B. On the other hand, cracking emanating from the top and bottom edges is observed for all models, which may be considered artifacts of the method.

Finally, Figure 3.9 plots the total reaction on the top edge versus time for models A, B, and C. Compared to Figure 3.3, here we see some more softening, and a delayed peak of the reaction (note the different values of t_f in the two figures).



(a) $0.5t_f$ (b) $0.5t_f$ (c) $0.5t_f$

(d) $0.75t_f$ (e) $0.75t_f$ (f) $0.75t_f$

(g) t_f (h) t_f (i) t_f

(j)

Figure 3.8: Cracked square plate under a tension test with $\dot{u}_D = \dot{u}_{DH}$. Phase field contours of models A (left column), B (middle column), and C (right column) (see Table 1.2) at different instants ($\{0.5, 0.75, 1.0\}t_f$) are shown in deformed configurations with the displacement scaled. Here $t_f = 2.0 \times 10^{-3}$s. We observe a straight crack for models A and C, while a branched crack for model B. Also observed are cracking from the top and bottom edges in all models, which may be considered artifacts of the method.

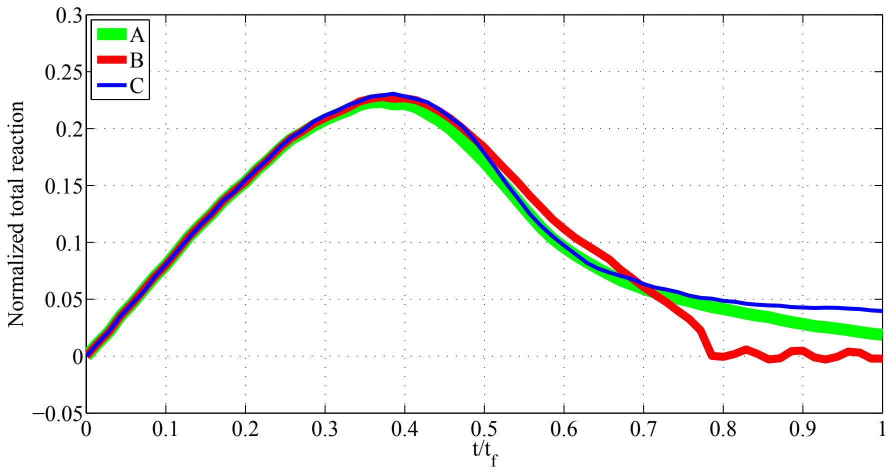


Figure 3.9: Cracked square plate under a tension test with $\dot{u}_D = \dot{u}_{DH}$. Total vertical reaction on the top edge versus time for models A, B, and C. The total reaction is normalized by $2E\dot{u}_D t_f/L \cdot L = 29.12$kN/mm, i.e., by the total reaction applied to the top edge at $t = t_f$ in the case without any crack or phase field evolution. Here $t_f = 2.0 \times 10^{-3}$s. All models give rise to similar results. The total reaction begins to decrease at approximately $t = 0.4t_f$. Compared to Figure 3.3, here we see later and more softening for all models (note the different values of t_f in the two figures).

3.2 Pure shear test of a cracked specimen

As the second example, we simulate a shear test of a squared plate with an initial crack at the middle height. The geometric setup and loading are shown in Figure 3.10. The regularization parameter is set to be $l = 1$mm.

We ran the simulation from $t = 0$ to $t = t_f$ where $t_f = 9.0 \times 10^{-3}$s. The mesh has 143,456 triangular elements and in the GPU kernel the nodes or elements are grouped into 32×32 blocks.

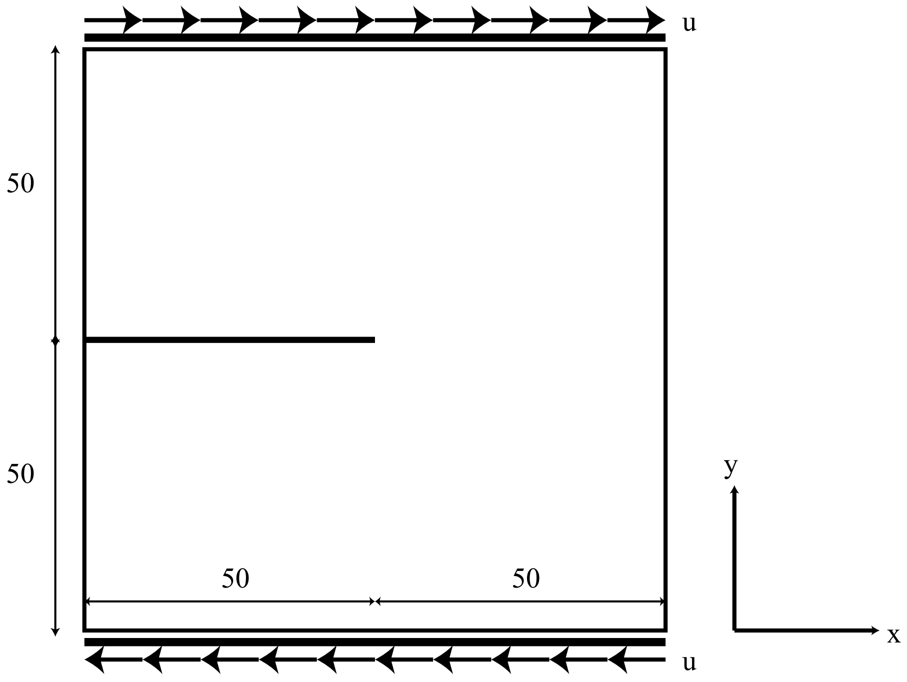
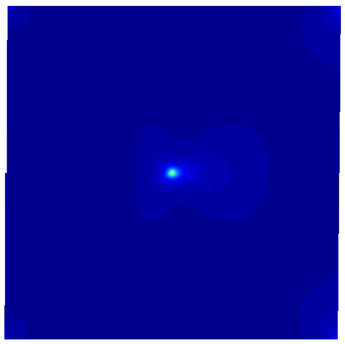
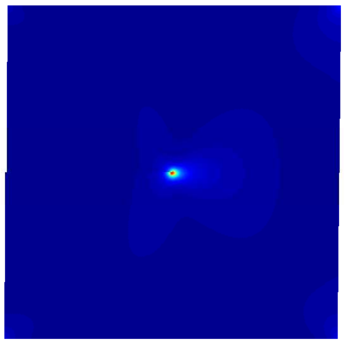
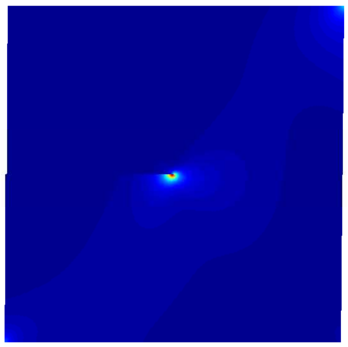


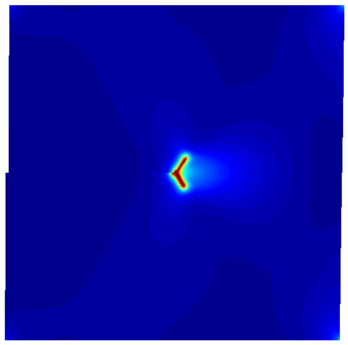
Figure 3.10: Schematic of a cracked square plate (unit: mm) under a displacement shear load. A monotonically increasing antisymmetric displacement loading $\boldsymbol{u} = \dot{u}_D t \boldsymbol{e}_x$ is applied on the top and $\boldsymbol{u} = -\dot{u}_D t \boldsymbol{e}_x$ on the bottom edge.

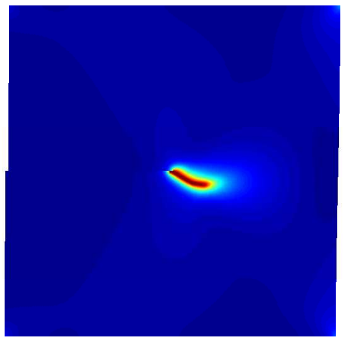
Figure 3.11 shows snapshots of the phase field contours obtained with models A, B, and C, at different instants $\{0.5, 0.75, 1.0\}t_f$, where $t_f = 9.0 \times 10^{-3}$s. For model A, in which the phase field evolution can be due to both tension and compression, the crack is branched, while for models B and C, the crack just kinks, with different kink angles. The crack topologies of models A and C agree well with a similar shear test in [Miehe et al., 2010b].

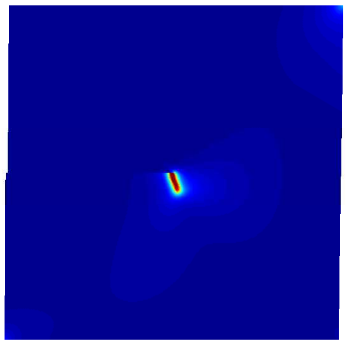

(a) $0.5t_f$

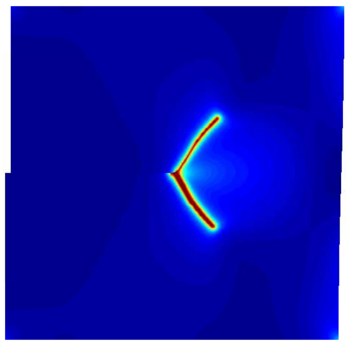

(b) $0.5t_f$

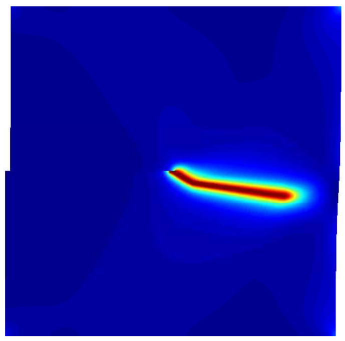

(c) $0.5t_f$

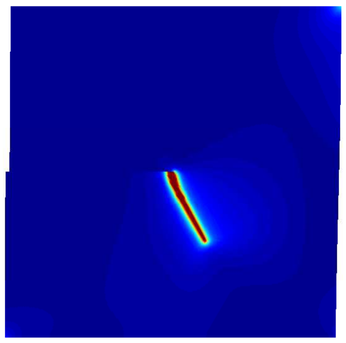

(d) $0.75t_f$

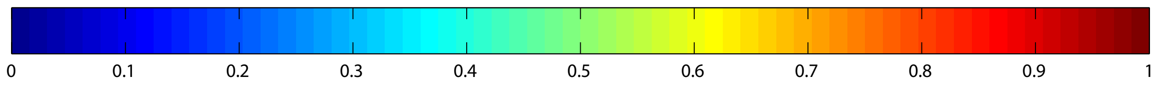

(e) $0.75t_f$


(f) $0.75t_f$


(g) t_f


(h) t_f


(i) t_f



(j)

Figure 3.11: Cracked square plate under a shear test. Phase field contours of models A (left column), B (middle column), and C (right column) (see Table 1.2) at different instants are shown in deformed configurations with the displacement scaled. Here $t_f = 9.0 \times 10^{-3}$s. The same parameters are used for all three phase field models. The sample is discretized into 143,456 three-noded triangular elements. The initial crack was explicitly introduced. For model A, a clear bifurcation into two branches is observed, while for models B and C the crack just kinks.

We show more results to compare with the tension test. Figure 3.12 depicts the evolution of the total reaction on the top (or bottom) edge for models A, B, and C. Figure 3.13 shows the evolution of the effective total crack length $|\Gamma|$. Figure 3.14 plots the calculated lower bound for the critical time step Δt_c^{LB} computed with (1.14) as a function of t. The trend is similar to the tension test, see Figure 3.5.

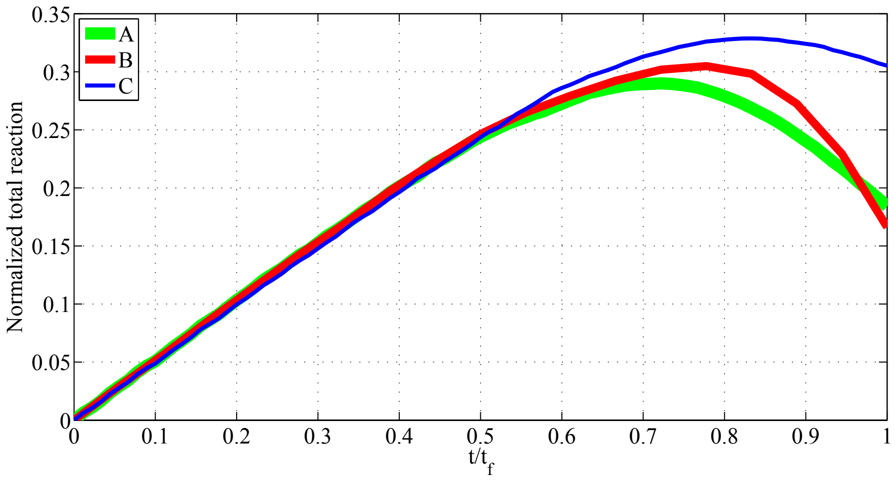


Figure 3.12: Pure shear test of a cracked specimen. The evolution of the total horizontal reaction on the top for models A, B, and C are shown. The data are normalized by $2\mu\dot{u}_D t_f/L \cdot L = 16$kN/mm, i.e., the total horizontal reaction on the top edge at $t = t_f$ if the specimen were unbroken. Here $t_f = 9.0 \times 10^{-3}$s. We observe overlapping curves for the three models before the onset of crack propagation.

Figure 3.15 shows the phase field contours obtained from three different values of the artificial viscosity η at time $t = t_f$. All other parameters are the same. Like Figure 3.6, although their propagation speeds are different, all samples evolve in a similar fashion. Finally, Figure 3.16 shows their corresponding total vertical reaction. The higher η, the lower the rate of crack evolution, and the higher the total vertical reaction on the top.

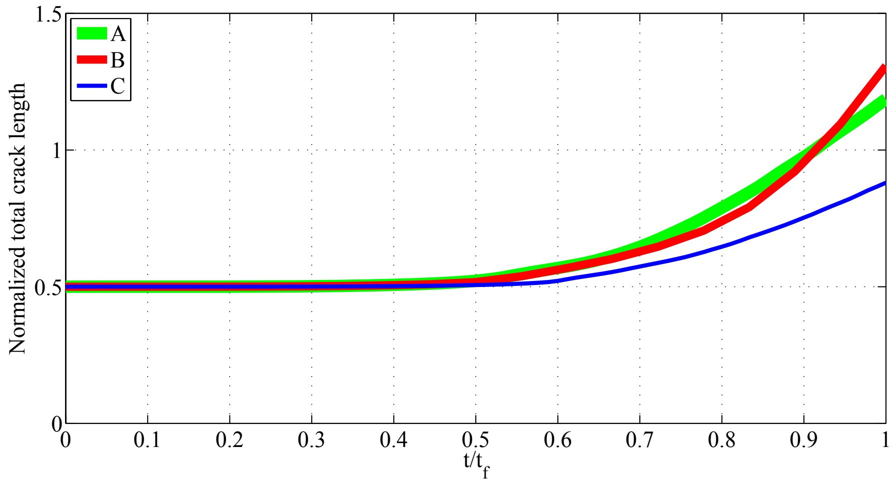


Figure 3.13: Pure shear test of a cracked specimen. Evolution of the total crack length as defined in (3.1) normalized by the edge length L for models A, B, and C. Note that the initial crack length is $0.5L$, and $t_f = 9.0 \times 10^{-3}$s.

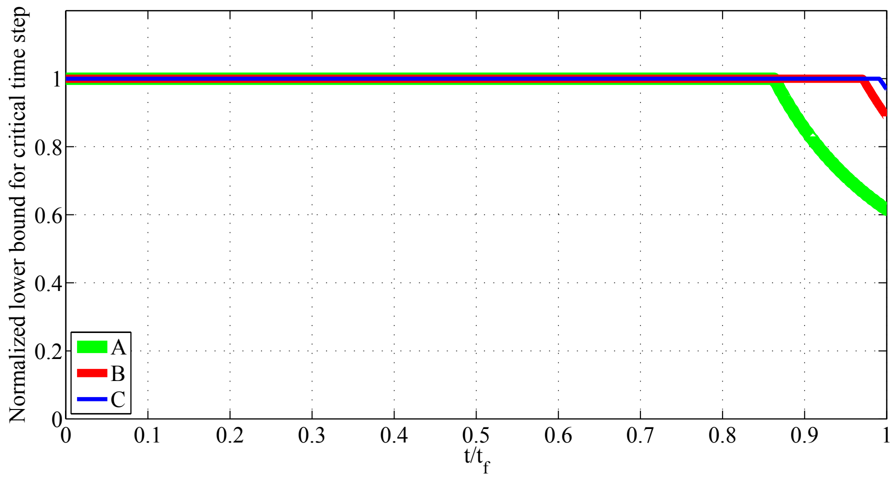


Figure 3.14: Pure shear test of a cracked specimen. A lower bound for the critical time step is calculated with (1.14) and normalized with Δt_{cu}^{LB} at $t = 0$, which is the same for all models. Here, the data are collected from the solution of models A, B, and C. Beginning from a certain point, as the phase field evolves, the lower bound decreases and Δt_c^{LB} solely depends on Δt_{cd}^{LB}. The reader is referred to Section 3.1 for an explanation of this trend. Here $t_f = 9.0 \times 10^{-3}$s.

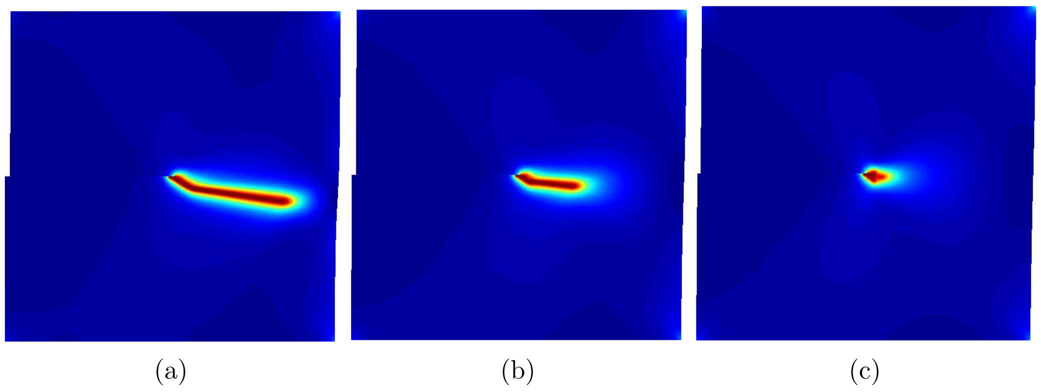


Figure 3.15: Cracked square plate under a shear test. Phase field contours for three samples of model B at $t = t_f$ are shown. Here $t_f = 9.0 \times 10^{-3}$s. The values of η for the samples are: (a) 1.0×10^{-6}kNs/mm^2, (b) 2.0×10^{-6}kNs/mm^2, and (c) 5.0×10^{-6}kNs/mm^2. Other parameters are the same. As seen, for a higher value of η, the evolution of the crack becomes slower. Nevertheless, all samples show a similar crack path.

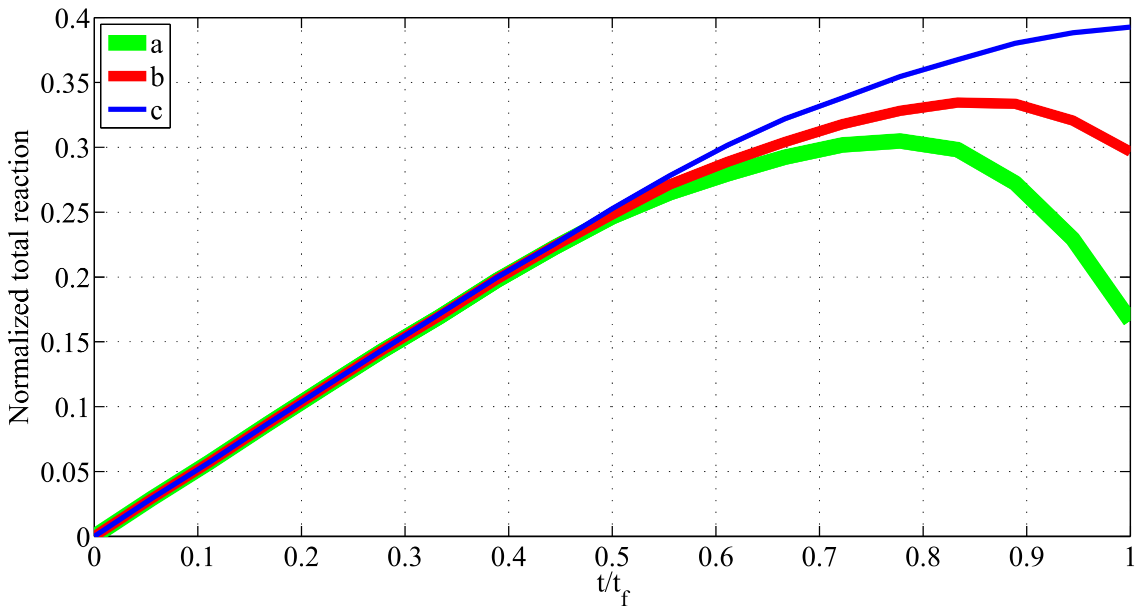


Figure 3.16: Pure shear test of a cracked specimen. Total vertical reaction on the top for three samples of model B with different values of η: (a) 1.0×10^{-6}kNs/mm^2, (b) 2.0×10^{-6}kNs/mm^2, and (c) 5.0×10^{-6}kNs/mm^2. The higher η the more is total reaction. Here $t_f = 9.0 \times 10^{-3}$s.

3.3 Symmetric bending test

We now investigate a bending test of a simply supported cracked beam. The geometry and the loading setup are shown in Figure 3.17. In contrast to the previous examples, here a uniformly distributed incremental *force* $w = 0.3\text{kN/mm}^2\ t/t_f$ rather than displacement is applied on a certain area on the top. Here $t_f = 0.03\text{s}$.

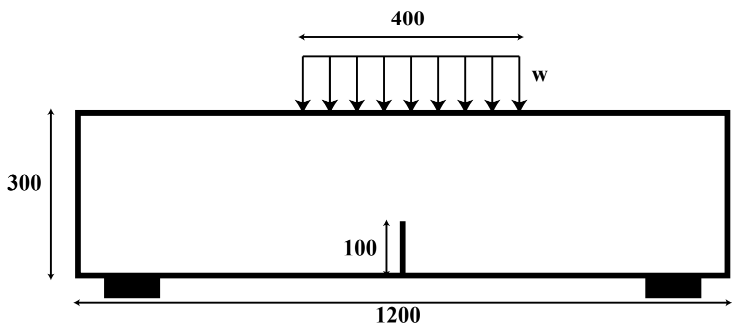
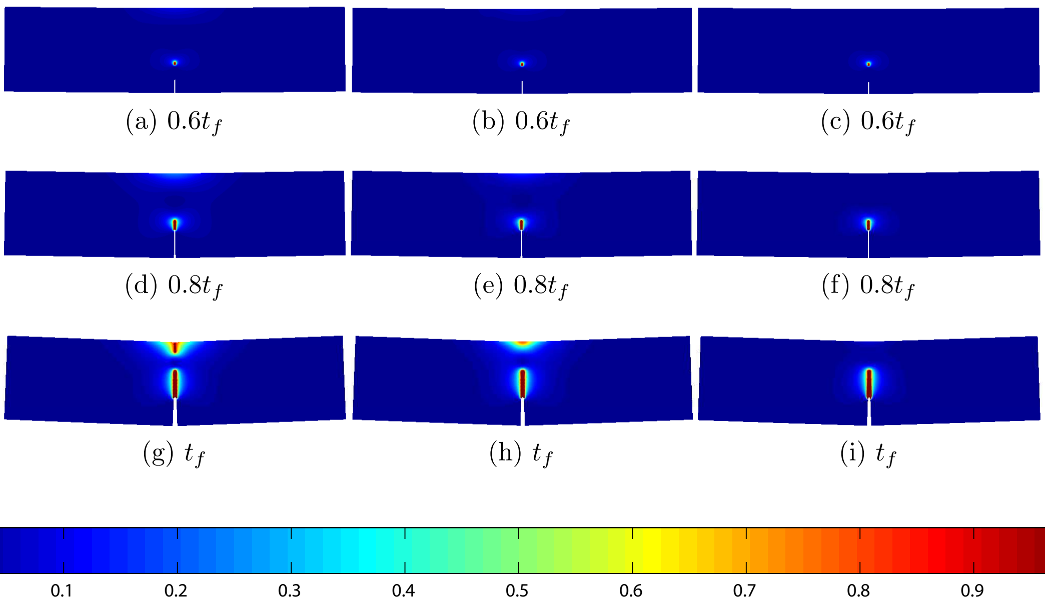


Figure 3.17: Symmetric bending test. A uniform monotonically increasing force loading is applied on the indicated area in the middle of the top edge. The specimen is initially cracked from the middle of bottom edge upwards.

We set the regularization parameter to be $l = 4\text{mm}$. Due to symmetry, only the left half of the beam is simulated with appropriate symmetric boundary conditions. This left half of the computational domain is discretized into 103,340 three-noded triangular elements. In the GPU kernel the nodes or elements are grouped into 24×24 blocks.

Figure 3.18 shows the snapshots of the phase field contours computed with models A, B, and C at selected instants $t = \{0.6, 0.8, 1.0\}t_f$. Here $t_f = 0.03\text{s}$. Other parameters are given in Table 3.1. From the figure, it is observed that the crack propagates upward for all models while the phase field under the loading area reaches high values with models A and B as well. From this aspect, model C may be the model of choice for many applications.



(j)

Figure 3.18: Symmetric bending test. Phase field contours at selected instants $t = \{0.6, 0.8, 1.0\}t_f$ obtained with models A (left column), B (middle column), and C (right column) are shown in the deformed configuration with the displacement scaled. Here $t_f = 0.03$s. Due to symmetry, only the left half of each sample is modeled, which is discretized into 103,340 three-noded triangular elements. The initial crack was explicitly introduced, and hence in the deformed configuration it appears as a white line. As seen, the initial crack propagates upward in all models. Moreover, there is another region with high phase field values right under the loading area for models A and B.

3.4 Conclusion

We have designed a GPU algorithm for the phase field approach to fracture, where the displacement field is a solution of an elastodynamic problem, while the phase field is updated according to a rate-dependent formulation, so that we obtain a fully explicit scheme. The explicit method fits nicely with the GPU architecture, especially in terms of the thread and memory hierarchies. This is a strong motivation for adopting a fully explicit method.

To ensure stability, we designed a time adaptivity strategy to account for the decreasing critical time step during the evolution of the field. The time step does not decrease until the latest stages of fracture when it becomes significantly smaller. As will be seen in the scalability study in A.2, this time adaptivity scheme maintains the overall scaling of the method, which will be as good as that of an implicit scheme, if not better.

The effect of the artificial viscosity on the solution is systematically studied. We observe that this parameter does not affect the crack path or the onset of crack propagation, although once started to propagate, it gives rise to different crack speeds, and also to different stress distributions.

A comparison among the studied phase field models is summarized as follows:

1. For the tension test, a low loading rate gives rise to straight crack paths for all models; however, when the loading rate is higher, only model B predicts a branching crack in the setup we tested, and the other two models still predict a straight crack.

2. For the shear test, model A predicts crack branching due its lack of discrimination of tension and compression. Models B and C predict a kink, but with different kink angles.

3. For the bending test, models A and B predict unphysical damage ahead of the crack, while model C does not.

In conclusion, generally model B and model C provide more physical predictions than model A for most materials. But there is no clear winner between models B and C in terms of the phenomena they are able to capture.

When the loading rate is high, we observe some non-physical damage from the edges where we impose the displacement loading, which we believe is an artifact of the method, and may worth further investigation.

Chapter 4

Where is the crack?

4.1 Introduction

The phase field method [Bourdin et al., 2008, Miehe et al., 2010b, Borden et al., 2012] is a powerful way of simulating crack propagation in that it can predict crack emergence and branching, under certain circumstances, without explicitly tracking the crack path. One of the main ideas of this method is to employ a smeared representation of discrete cracks. However, in some applications it is still convenient to have the explicit crack path available, or even to develop a mechanism to introduce crack paths to partially replace a smeared crack propagation model, such as the approaches described in [Jirásek and Zimmermann, 2001, Tamayo-Mas and Rodríguez-Ferran, 2014, Wang and Waisman, 2016]. One application area is leakage problems [Pozrikidis, 2013].

Hydraulic fracturing is another example for which a crack identification scheme may be useful. This is a technique used in the oil and gas industry, where fractures are propagated by high-pressure liquid inside them, see [Hunsweck et al., 2013] and references therein. With a phase field (or damage) approach, it becomes challenging how to impose the pressure loading on the rock mass exerted by the fracturing fluid, since most phase field approaches to date were developed for traction-free crack faces. Existing approaches that employ a phase field (or damage) approach to simulate hydraulic fracturing can be classified into two families.

The first family assumes either a known pressure field or a known displacement field [Bourdin et al., 2012, Chukwudozie et al., 2013, Mikelić et al., 2013a,b, Wheeler et al., 2014]. Clearly an accurate simulation of the process requires considering a fully coupled system in order to obtain the spatial variation of the pressure field along the fracture, especially when the fluid viscosity is not negligible, as in most applications. Hence, this family of models can be considered as intermediate results towards more sophisticated models.

The second family defines an anisotropic permeability tensor [Miehe et al., 2015, Zhao et al., 2015] or an indicator function [Mikelić et al., 2015b,a] related to the phase field, in order to obtain the fluid flow rate inside the fracture. When the solid of interest is at least moderately porous, these approaches are of technical importance; nevertheless, when the solid is barely permeable, having a pressure field defined everywhere seems costly as the phase field itself is already an additional scalar field to solve for, if one compares an explicit crack approach with the pressure field only defined on the crack path. In this case, having the crack path available from a phase field approach may offer a possibility of developing less expensive numerical schemes for impermeable solids.

Identifying the crack path from a phase field approach or damage model has been the interest of a couple of contributions. For example, Tamayo-Mas and Rodríguez-Ferran [Tamayo-Mas and Rodríguez-Ferran, 2015] proposed a medial-axis-based approach in the context of a continuous failure model characterized by a damage field. Therein, the authors proposed to model the crack path using the θ-simplified medial axis of a certain isoline of the damage variable. Bottoni *et al.* [Bottoni et al., 2015] developed another approach for searching the crack path from a damage field in up to three-dimensions. In the case of two-dimensions, they obtained location points one after another by finding the local maximum of the damage variable within a specific line at a distance from the last determined point.

In this work, we build a crack identification scheme by taking advantage of the variational structure of the phase field approach, although the scheme can also be applied to a solution from a damage model. The methodology is built on the so called *equivalent phase field* of a given curve, which is the generalization from the analytic phase field solution from the potential energy functional in a special case. The proposed algorithm is able to identify not only a simple crack but also a branched crack. The method is also capable of distinguishing multiple cracks close to each other up to a certain clearance.

This algorithm consists of three main steps. First of all, the non-maximum suppression method is employed to identify a bounding area within which the crack is believed to be. This is one of the main novelties of this work, in that the gradient information of the phase field is exploited as well as the value. The bounding area provides the topological information of the crack path, and thus especially advantageous in the case of a branched crack. Then, a number of interpolation points are determined from this bounding area, from which a cubic spline is built as the initial guess for the crack. This step is comparable to the approach introduced in [Bottoni et al., 2015] but ours ensures: (1) a more uniform distance between the identified points and also (2) a stable weighted average expression to determine these points rather than local maximum. Afterwards, an optimization step is undertaken to improve the crack path. This is another novelty of our work, in that the crack tip and junction locations will be significantly improved with this optimization step, even though the overall accuracy of the algorithm is limited by a resolution at best twice the mesh size. All of these steps are developed from the construction of the equivalent phase field.

The remaining of this chapter will proceed as follows. In Section 4.2, we recall the phase field formulations for the variational description of brittle fracture, and introduce the concept of equivalent phase field of a given curve. Then in Section 4.3, we formulate the crack identification problem as an optimization problem by virtue of this equivalent phase field. Afterwards, in Section 4.4 we detail the proposed algorithm for this optimization problem, followed by Section 4.5, where we provide three representative sets of examples to demonstrate the performance of the proposed algorithm. Finally we draw some conclusions in Section 4.6.

4.2 The phase field approach to fracture and the equivalent phase field

Here we recapitulate the phase field formulation for brittle fracture, followed by the introduction of the so-called equivalent phase field.

Consider a two-dimensional (plane stress or plane strain) solid occupying an open Lipschitz domain Ω in the undeformed configuration. Let $\boldsymbol{u} : \Omega \to \mathbb{R}^2$ be the unknown displacement field corresponding to a certain load. Let $\partial_D\Omega, \partial_N\Omega \subseteq \partial\Omega$ be such that $\partial_D\Omega \cup \partial_N\Omega = \partial\Omega$ and $\partial_D\Omega \cap \partial_N\Omega = \emptyset$, and $\boldsymbol{u}_D : \partial_D\Omega \to \mathbb{R}^2$ and $\boldsymbol{t}_N : \partial_N\Omega \to \mathbb{R}^2$ be prescribed displacement and traction boundary conditions, respectively. Finally, let $\boldsymbol{b} : \Omega \to \mathbb{R}^2$ be the prescribed body force field.

The static case The phase field approach to fracture is built on energy minimization with respect to $\boldsymbol{u}$ and its jump set, which we denote as $\Gamma = \Gamma(\boldsymbol{u}) \subset \Omega$. Let $|\Gamma|$ denote the length of Γ. Then the total potential energy of the fractured brittle solid can be written as:

$$\Pi[\boldsymbol{u}] := \int_{\Omega\setminus\Gamma} \psi_0[\boldsymbol{\varepsilon}(\boldsymbol{u})]\, d\Omega - \int_{\Omega} \boldsymbol{b}\cdot\boldsymbol{u}\, d\Omega - \int_{\partial_N\Omega} \boldsymbol{t}_N\cdot\boldsymbol{u}\, d\Gamma + g_c|\Gamma|, \tag{4.1}$$

where $\psi_0[\boldsymbol{\varepsilon}(\boldsymbol{u})]$ is the strain energy density that depends on the strain $\boldsymbol{\varepsilon}(\boldsymbol{u}) = [\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T]/2$, and $g_c > 0$ is the strain energy release rate, the strain energy released per unit length of crack extension.

To develop a numerical method to approximate the sharp crack description (4.1), the phase field approach replaces the sharp crack path Γ with a phase field d, where d is a scalar field, $d : \Omega \to [0, 1]$. There exist different conventions for the meaning of the extreme values of d. Here we adopt one where regions with $d = 0$ and $d = 1$ correspond to pristine and fully broken states of the material, respectively.

One of the ingredients of the phase field approach is the crack length functional, which takes the following form:

$$\Gamma_\ell[d] := \frac{1}{2}\int_\Omega \left(\frac{d^2}{\ell} + \ell \nabla d \cdot \nabla d \right) d\Omega,$$

where $\ell > 0$ is the regularization length scale, which may also be interpreted as a material property, e.g., the size of the process zone.

To take into account any pre-existing crack, we define $\partial_d\Omega \subset \overline{\Omega}$ to be a set with Hausdorff dimension 1, and $d_0 : \partial_d\Omega \to [0,1]$ as the Dirichlet boundary condition for d. With this, we define the affine space of the admissible $\boldsymbol{u}$ and d fields to be

$$\mathscr{S}_u := \left\{ \boldsymbol{u} \in H^1\left(\Omega; \mathbb{R}^2\right) \middle| \boldsymbol{u} = \boldsymbol{u}_D \text{ on } \partial_D\Omega \right\},$$
$$\mathscr{S}_d := \left\{ d \in H^1(\Omega) \middle| 0 \le d \le 1 \text{ a.e.}, d = d_0 \text{ on } \partial_d\Omega \right\},$$

The regularized variational formulation for brittle fracture of the solid reads: Find $(\boldsymbol{u}, d) \in \mathscr{S}_u \times \mathscr{S}_d$ that minimizes the following potential energy:

$$\Pi_\ell[\boldsymbol{u}, d] := \int_\Omega \psi[\boldsymbol{\varepsilon}(\boldsymbol{u}), d]\, d\Omega - \int_\Omega \boldsymbol{b} \cdot \boldsymbol{u}\, d\Omega - \int_{\partial_N\Omega} \boldsymbol{t}_N \cdot \boldsymbol{u}\, d\Gamma + g_c \Gamma_\ell[d]. \tag{4.2}$$

Here, $\psi[\boldsymbol{\varepsilon}(\boldsymbol{u}), d]$ is the strain energy density dependent on the phase field, which satisfies $\psi[\boldsymbol{\varepsilon}(\boldsymbol{u}), d = 0] = \psi_0[\boldsymbol{\varepsilon}(\boldsymbol{u})]$ and $\psi[\boldsymbol{\varepsilon}(\boldsymbol{u}), d_1] \ge \psi[\boldsymbol{\varepsilon}(\boldsymbol{u}), d_2]$ whenever $d_1 < d_2$. Note that the fracture energy contribution is given by the last term, i.e., $\Gamma_\ell[d]$ plays the role of $|\Gamma|$ in the sharp crack description (4.1).

It can be shown that when $\ell \to 0$, the solution to the minimization problem (4.2) Γ-converges to that of (4.1), and also $\Gamma_\ell[d]$ converges to $|\Gamma|$.

The dynamic case The formulation (4.2) can be generalized to dynamic fracture, for which the following Lagrangian can be written:

$$L[\boldsymbol{u}, \dot{\boldsymbol{u}}, d] := \int_\Omega \frac{1}{2}\rho \dot{\boldsymbol{u}} \cdot \dot{\boldsymbol{u}}\, d\Omega - \Pi_\ell[\boldsymbol{u}, d],$$

where the dot symbol over a variable, $\dot{\Box}$, denotes differentiation with respect to time t.

Then for some final time $t_f > 0$, Hamilton's principle gives rise to the following formulation: Find the stationary point of the following action functional:

$$S[\boldsymbol{u}, d] := \int_0^{t_f} L[\boldsymbol{u}, \dot{\boldsymbol{u}}, d]\, dt$$

among all $(\boldsymbol{u}, d) : \Omega \times [0, t_f] \to \mathbb{R}^2 \times [0, 1]$, $(\boldsymbol{u}(\cdot, t), d(\cdot, t)) \in \mathscr{S}_u \times \mathscr{S}_d$ with the boundary conditions $\boldsymbol{u}_D$, $\boldsymbol{t}_N$, and d_0 possibly dependent on t, for $t \in [0, t_f]$.

Note that the irreversibility of the crack can be explicitly enforced by requiring $d(\cdot, t_1) \leq d(\cdot, t_2)$, a.e., whenever $t_1 < t_2$.

Analytic solution for a special case Let $\Omega = \mathbb{R}^2$, $\partial_d\Omega = \{(x, y)|y = 0\}$, and $d_0 = 1$. Minimizing (4.2) with the constraint $\boldsymbol{u} \equiv 0$ and the additional boundary condition $\partial d/\partial y \to 0$ as $y \to \pm\infty$ leads to following phase field solution $\boldsymbol{x}$ Miehe et al. [2010b]:

$$d(\boldsymbol{x}) = \exp\left(-\frac{|y|}{\ell}\right). \tag{4.3}$$

The equivalent phase field The analytic solution above motivates the definition of the so called *equivalent phase field*, for expressing the phase field equivalent (to some degree) to the crack set Γ. This notion is instrumental in the crack identification algorithm to be introduced later. For $\Gamma \subset \Omega$, we define the equivalent phase field as $d_\Gamma \in C^0(\Omega)$,

$$d_\Gamma : \boldsymbol{x} \mapsto \exp\left(-\frac{\operatorname{dist}(\boldsymbol{x}, \Gamma)}{\ell}\right), \tag{4.4}$$

where $\operatorname{dist}(\boldsymbol{x}, \Gamma) := \inf_{\boldsymbol{y} \in \Gamma} |\boldsymbol{x} - \boldsymbol{y}|$ is the distance from $\boldsymbol{x}$ to Γ. The equivalent phase field of a branched crack is depicted in Figure 4.1.

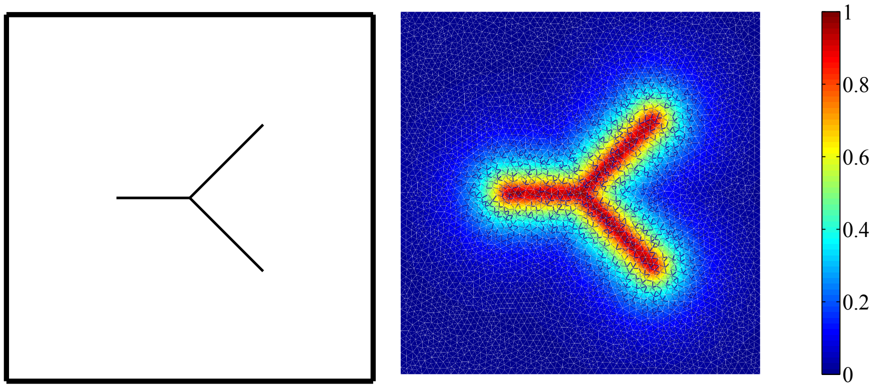


Figure 4.1: Contour plot of the equivalent phase field (right) of a branched crack (left). The equivalent phase field is the basis of the proposed crack detection algorithm. In this work we have adopted a convention for the (equivalent) phase field in which $d = 0$ and $d = 1$ correspond to the pristine and fully damaged states of the material, respectively.

4.3 Problem statement

This section sets up the problem of identifying the crack from a phase field description as a constrained optimization problem.

Let $(\boldsymbol{u}, d)$ be the minimizer of (4.2), or a numerical approximation to it. We aim to develop a practical method to obtain $\Gamma \subset \Omega$ with Hausdorff dimension 1 whose equivalent phase field d_Γ, as defined in (4.4), is the closest to d in a certain norm. With these, the crack identification problem can be (loosely) stated as: Among all possible curves contained in Ω, find Γ that minimizes:

$$N_d[\Gamma] := \|d - d_\Gamma\|^2. \tag{4.5}$$

Here we would like to remark on two aspects.

1. In practice it is difficult to search among all possible curves, hence we limit ourselves to a class of cubic splines contained in Ω in order to render the problem with finite dimensions.

2. The choice for $\|\cdot\|$ is not unique. The natural one is $\|\cdot\| = (\Gamma_\ell[\cdot])^{1/2}$; however,

our experience shows that $\|\cdot\| = \|\cdot\|_{L_2(\Omega)}$ is a more cost effective choice, since this choice avoids calculating ∇d_Γ at Gauss integration points. Moreover, we experienced very slow convergence for the former choice under certain conditions.

4.4 Algorithm

In this section, we present an algorithm for the problem described in Section 4.3. The algorithm is able to identify not only simple cracks but also branched cracks, and multiple cracks with a certain clearance. Note that our algorithm is primarily developed for the case in which d is expressed with finite element shape functions associated with some mesh $\mathscr{T}_h$; in case an exact solution of d is available as input, we will adopt any admissible finite element mesh $\mathscr{T}_h$ for Ω and then replace d by its interpolant with $\mathscr{T}_h$, which is still denoted by d when there is no risk of confusion.

Here an admissible finite element mesh $\mathscr{T}_h$ is such that the intersection of any two elements (considered as closed sets) of $\mathscr{T}_h$ can only be empty, a common vertex, or a common edge.

The entire algorithm is summarized in Table 4.1 with an illustration with a simple crack, whereas each main step therein is explained in details by the following subsections. Within this Section 4.4, we present an example of a simple crack with an unstructured triangular mesh (and some steps also with a structured square mesh) to better explain the steps; readers are referred to Section 4.5 for more involved examples.

4.4.1 Identifying a bounding area using non-maximum suppression

The first step of the algorithm is to find an area that contains the crack set Γ, with the assumption that (4.4) is a good model for the relation between d and Γ. For convenience, we let this area be made of complete elements. More precisely, we seek $E \subset \mathscr{T}_h$ be the set of elements whose area $\Omega_E := \bigcup\{e \in E\}$ is presumed to contain

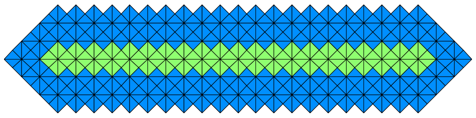
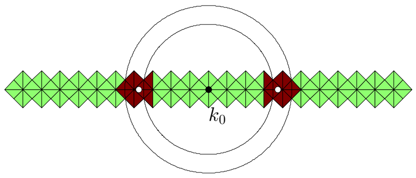
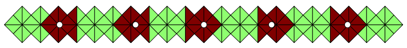
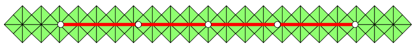
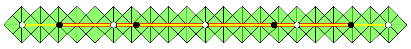
Identify the minimum element set E_0 that covers the region $d^{-1}([d_{\text{crit}}, 1])$ (all elements shown on the right) and a subset E (elements in green) that covers the ridge of the phase field d with Algorithm 6. Also see Figure 4.3.	
Identify a set of initial knots K with Algorithm 7, starting from the black dot in the middle. Also see Figure 4.7.	
Continue to identify a set of initial knots K with Algorithm 7. Also see Figure 4.8.	
Construct a cubic spline that interpolates K as the first iteration of Γ, which is denoted $\Gamma^{(0)}$ (See Section 4.4.3).	
Minimize (4.5) to get an optimized cubic spline as Γ (See Section 4.4.3). Also see Figure 4.10b and its caption.	

Table 4.1: A bird's-eye-view of the crack detection algorithm

the crack path Γ. The crack path Γ will then be searched within this region Ω_E. A possible approach is to let E be the set of all elements e such that $\max_e d \geq d_{\text{crit}}$ with some d_{crit} smaller than but close to 1, i.e., the minimum set of complete elements that covers $d^{-1}([d_{\text{crit}}, 1])$. To reduce the arbitrariness in the choice of this parameter d_{crit}, here we introduce another approach named *non-maximum suppression*, inspired by edge detection algorithms for image processing. The main idea of non-maximum suppression used in this work is to retain only elements possibly containing the ridge of the d profile, which corresponds to the crack path.

The general case The main idea of non-maximum suppression adapted to the problem at hand is to exploit the directions of ∇d between neighboring elements. Here we will explain the idea in the context of P_1 elements with the general algorithm given in Algorithm 6. Consider any element e, since ∇d is constant in this element, we denote it as $\nabla_e d$. Let d assume values d_a, d_b, and d_c at its three nodes a, b, and c, such that $d_a > \{d_b, d_c\}$. The essence of non-maximum suppression is, **if** $d(\boldsymbol{x}_a + \epsilon(\nabla d)_e) > d_a$ **when** $\epsilon \to 0^+$**, then element** e is on only one side of Γ, and thus **need not be considered for crack path identification**, i.e., element e should not be in E. This idea is illustrated in Figure 4.2. Figure 4.3 shows the bounding area E for a simple crack following Algorithm 6.

Simplification in the case of a Cartesian mesh We next illustrate the algorithm specialized to a setting very similar to the context of image processing: When $\mathscr{T}_h$ is a Cartesian grid of mesh with Q_1 square elements. A typical element and a few of its neighbors are shown in Figure 4.4, along with the row and column numbers shown next to the nodes. Let us consider element e shown therein and the general case in which the maximum of d over this element is attained at one of the nodes, say (i, j), without loss of generality. Then clearly $\nabla_e d$ evaluated at node (i, j) points into element e_n. In this case, the condition to remove element e can be cast on the nodal

Algorithm 6: Algorithm for the non-maximum suppression method to identify the region E that contains Γ

```
input  : $d \in \mathscr{S}_d$ interpolated with mesh $\mathscr{T}_h$
output: An element set $E$ (that is supposed to contain $\Gamma$) as a union of
        elements

Initialize $E = E_0 := \{e \in \mathscr{T}_h | \max_e d \geq d_{\text{crit}}\}$
for $e \in E$ do
    Let $p \in e$ be the centroid of $\arg\max_e d$, where $e$ is understood as a closed
    set
    /* In most cases $p$ is the node of $e$ with the maximum $d$ value;
       in the special case in which the maximum of $d$ over $e$ is
       attained at multiple discrete points, an edge, or the entire
       element                                                    */
    if $p \in \partial e$ then
        Compute $\nabla d$ at point $p$ from element $e$, denoted $\nabla_e d(p)$
        Find element $e_n$ which contains $p_\epsilon := p + \epsilon \nabla d$ for some $\epsilon \in (0, \rho_{e_n})$,
        where $\rho_{e_n}$ is the inradius of $e_n$
        /* If there are multiple $e_n$'s containing $p_\epsilon$, choose any one
           */
        if $\nabla_e d(p) \cdot \nabla_{e_n} d(p) > 0$ then
            $E \leftarrow E \setminus \{e\}$
```

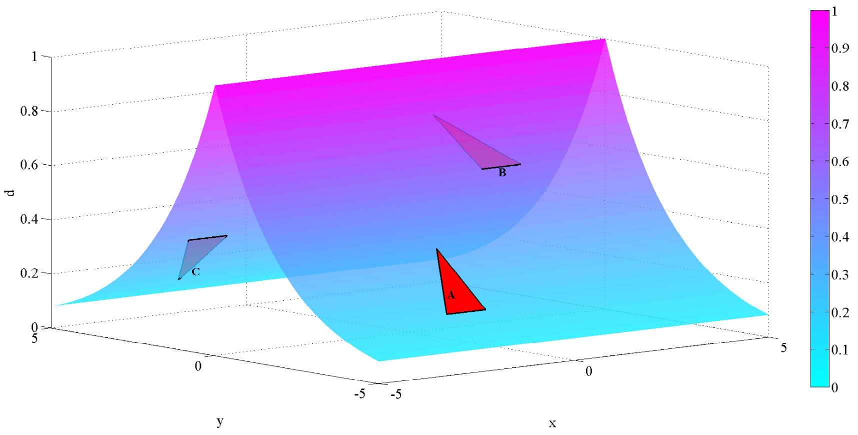


Figure 4.2: Illustration of the idea of the non-maximum suppression algorithm. The surface plot is the equivalent phase field d_Γ for a crack located at $\Gamma = \{(x, y)|y = 0\}$. The discrete phase field d over P_1 elements A, B, and C are also shown in the figure as red triangles, where the values of d at the nodes of these elements are assumed to coincide with d_Γ, for clarity of explanation. It is clear that the direction of ∇d are along the y-direction in both elements A and B, but negative y-direction in element C. However, from the gradient criterion described in the text and also from direct observation, elements A and C are located on either side of Γ and will not contain any part of Γ while element B intersects Γ. The outcome of the non-maximum suppression algorithm (Algorithm 6), E, will contain element B but not element A or C.

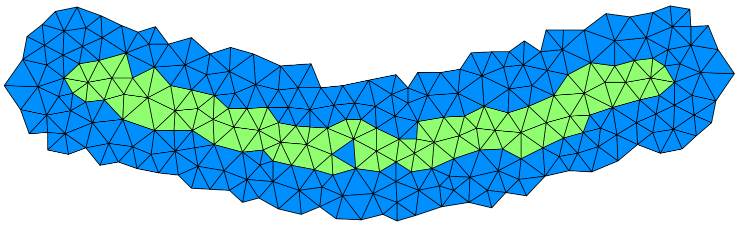


Figure 4.3: Schematic showing the set of elements E_0 (with $d_{\text{crit}} = 0.6$) (green and blue) and E (green) as identified from Algorithm 6. The crack path will be sought in the green region. As seen, for the chosen value of d_{crit}, the non-maximum suppression algorithm results in a much smaller set of elements for later identification of the crack set Γ.

values of d:

$$\left(d_{(i+1,j)} - d_{(i,j)}\right)\left(d_{(i,j)} - d_{(i-1,j)}\right) + \left(d_{(i,j+1)} - d_{(i,j)}\right)\left(d_{(i,j)} - d_{(i,j-1)}\right) > 0, \quad \text{if } d_{(i,j)} > \left\{d_{(i-1,j)}, d_{(i-1,j-1)}, d_{(i,j-1)}\right\}. \tag{4.6a}$$

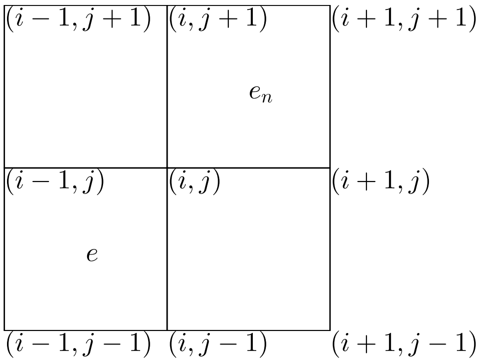


Figure 4.4: A typical element e in a Cartesian mesh and three of its neighbors are shown. One of these neighbors is marked e_n. See the text for its significance. Also shown are the row and column numbers of the nodes.

In the very special case of exactly two nodes sharing the same edge attaining the maximum value of d over element e, for example, $d_{(i-1,j)} = d_{(i,j)} > \{d_{(i-1,j-1)}, d_{(i,j-1)}\}$, then we retain element e if the said edge is on the ridge of the d profile. In other words, we remove element e if

$$\left(d_{(i-1,j+1)} + d_{(i,j+1)} - 2d_{(i,j)}\right)\left(2d_{(i,j)} - d_{(i-1,j-1)} - d_{(i,j-1)}\right) > 0, \quad \text{if } d_{(i-1,j)} = d_{(i,j)} > \left\{d_{(i-1,j-1)}, d_{(i,j-1)}\right\}. \tag{4.6b}$$

It is reminded that both equations of (4.6) should be generalized with symmetry. A detailed example with nodal values of d is given in Figure 4.5 for the reader interested in following exactly the criterion (4.6). Another example with a curved crack is shown in Figure 4.6.

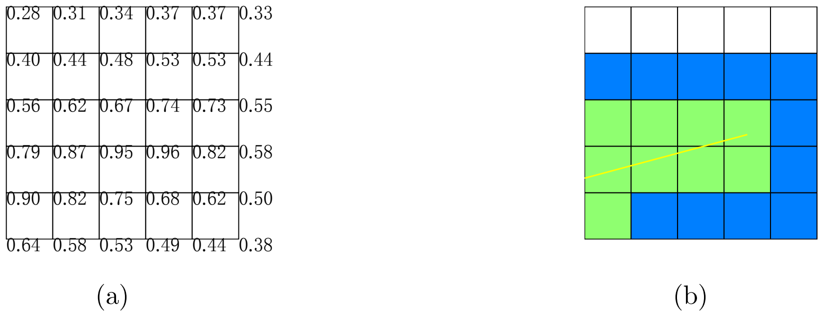


Figure 4.5: (a) A subarea of a Cartesian mesh with the phase field d values marked at the nodes. These values are the equivalent phase field values of the crack shown in yellow in (b) with $\ell = 2\sqrt{2}a$ where a is the side length of the elements. (b) Schematic showing the effect of Algorithm 6: The set of elements E_0 (with $d_{\text{crit}} = 0.6$) is shown in either green or blue, and the set E is shown in green. The crack path will be sought in the green region. It can be seen that some elements, even though with a nodal point having a d value of 0.82, are not retained per (4.6).

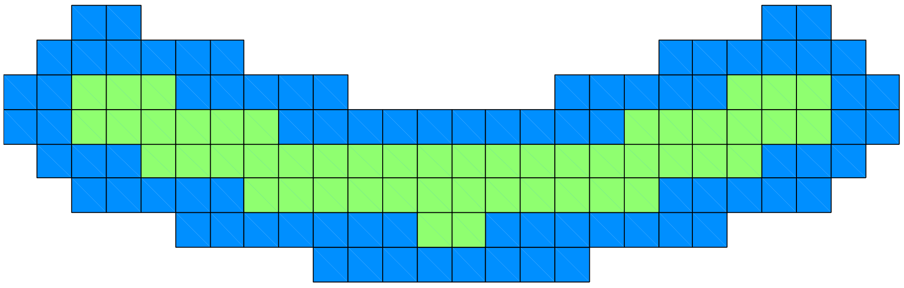


Figure 4.6: Schematic showing the set of elements E_0 (with $d_{\text{crit}} = 0.6$) (green and blue) and E (green) as identified from Algorithm 6. The crack path will be sought in the green region. See the text for more explanations.

4.4.2 Identifying an initial set of knots and their connectivity

Having obtained the bounding area Ω_E, we next identify a set of knots in Ω_E, denoted as K, and the connectivity (or incidences) of these knots, denoted as I, with which we will define cubic splines as the initial guess for the crack set Γ. Ideally, K consists of local maxima for d, see, for example, points on the line $\{y = 0\}$ in Figure 4.2. The steps to identify K and I are described in Algorithm 7.

We start with a provisional knot $k_0 \in \Omega_E$, which is one of the points that attains the maximum value of d in this region, typically a nodal point. Then we draw an annulus A centered at k_0 with outer and inner radii as $(\alpha + 1/2)h$ and $(\alpha - 1/2)h$, respectively, where h denotes the maximum element diameter in E and $\alpha \geq 2$ is a constant. The choice of α sets the resolution of the crack detection algorithm, which is given by $(\alpha + 1/2)h$.

We then search the next knot in the vicinity of the intersection of the annulus A and the region Ω_E. More precisely, we determine the next knot location by taking a weighted average of the nodal points belonging to each cluster of elements (denoted as E_i^A in Algorithm 7) that intersect A, see (4.8). These weights depend on the nodal solutions of d at these nodes according to (4.7). The expression (4.7) is, again, motivated by (4.3), with the consideration of making the weights finite. The expression (4.8) involves the coordinates and d-values of a group of nodes, and hence is relatively stable. A brief derivation of this formula is as follows.

Let Γ be straight for the sake of simplicity. Then let p_1 and p_2 be two nodes on different sides of Γ with d_Γ values d_1 and d_2, respectively. Let the distances of these two points to Γ be ρ_1 and ρ_2, respectively. Then from (4.4),

$$d_i = \exp(-\rho_i/\ell), \quad i = 1, 2.$$

If $d_1, d_2 < 1$, then $\rho_1/\rho_2 = \log_e d_1 / \log_e d_2$. As a result, Γ goes through the point p (the desired knot) on the line segment p_1p_2 with $\|p-p_1\|/\|p-p_2\| = \rho_1/\rho_2 = \log_e d_1/\log_e d_2$. The coordinates of p can be obtained by the formula (4.8) with the points p_1 and p_2 and corresponding weights $(\log_e d_1)^{-1}$ and $(\log_e d_2)^{-1}$.

Algorithm 7: Algorithm to identify a set of knots K and their connectivity I

```
input  : Element set $E$ from Algorithm 6, $\alpha \geq 2$, $\epsilon$ such that $0 < \epsilon \ll 1$
output: A set of knots $K$ and their connectivity $I$ as a set of unordered pairs
        of members of $K$

Choose any $k_0 \in \arg\max_{\Omega_E} d$, where $\Omega_E = \bigcup\{e \in E\}$.
Let $K = \{k_0\}$, $K^* = \emptyset$, and $E_r = E$
/* $k_0$ is a provisional knot, which will be removed after its first
   usage.                                                          */
while $E_r \neq \emptyset$ do
    Choose any $k^* \in K \setminus K^*$
    $K^* \leftarrow K^* \cup \{k^*\}$
    Let $A$ be the annulus centered at $k^*$ with outer and inner radii as
    $(\alpha + 1/2)h$ and $(\alpha - 1/2)h$, respectively, where $h$ denotes the largest
    diameter of the elements in $E$
    Find $E^A = \bigcup\{e \in E_r | e \cap A \neq \emptyset\}$
    Partition $E^A$ into $n$ clusters $\{E_i^A\}_{i=1}^n$ with Algorithm 8
    for $i = 1$ to $n$ do
        Let $P_i^A = \{p_{ij}\}_{j=1}^{N_i}$ be the set of all nodes in $E_i^A$
        Let $w_{ij}$ for each node $p_{ij}$ be determined from:

            $$w_{ij} = \begin{cases} 1/\left|\log_e d\left(p_{ij}\right)\right|, & \text{if } d\left(p_{ij}\right) < 1 - \epsilon, \\ 1/\left|\log_e (1 - \epsilon)\right|, & \text{otherwise.} \end{cases} \quad (4.7)$$

        Determine knot $k_i^A$ from

            $$k_i^A = \frac{\sum_{j=1}^{N_i} w_{ij} p_{ij}}{\sum_{j=1}^{N_i} w_{ij}} \quad (4.8)$$

        /* Here $k_i^A$ and $p_{ij}$ are understood as position vectors in
           $\mathbb{R}^2$.                                             */
        $K \leftarrow K \cup \{k_i^A\}$, $I \leftarrow I \cup \{(k^*, k_i^A)\}$
    if $k^* = k_0$ then
        $K = K \setminus \{k_0\}$, and remove all entries of $I$ involving $k_0$
    else
        Let $B$ be the closed disc enclosed by the outer circle of $A$
        $E_r \leftarrow E_r \setminus \bigcup\{e \in E_r | \text{any node of } e \text{ is in } B\}$
```

Finally, to reduce the arbitrariness of the choice of k_0, we will remove this knot from the list after first using it. Here we remark that the removal of k_0 is similar to the back correction step in Bottoni et al. [2015].

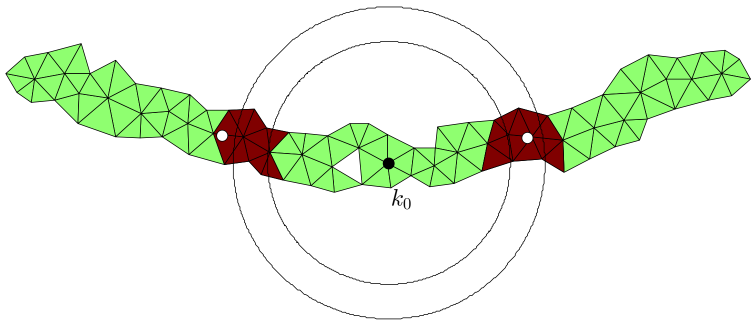


Figure 4.7: Identifying a set of initial knots K and their connectivity I: According to Algorithm 7, we start from k_0 (black dot in the middle) and draw an annulus A with outer and inner radii as $(\alpha + 1/2)h$ and $(\alpha - 1/2)h$, respectively. Elements intersected by A (E^A) are shown in brown, which will be partitioned into two clusters E_1^A and E_2^A by Algorithm 8 according to the connectivities of the elements. Algorithm 7 then identifies one knot from each cluster, shown as white dots.

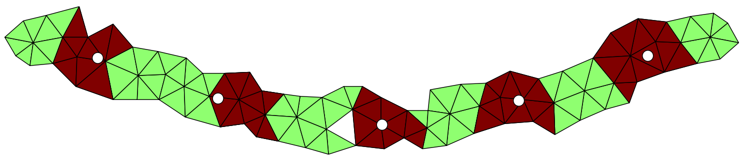


Figure 4.8: A chain of knots identified to represent a simple crack. Only the elements in E are shown. Each knot is obtained from each cluster of elements E_i^A (shown in brown) with Algorithm 7.

4.4.3 Constructing and optimizing the cubic spline representation of Γ

With the set of knots K and their connectivity I identified from Algorithm 7, we will represent the crack set Γ by spline interpolation.

Algorithm 8: Algorithm for partitioning E^A into disjoint subsets (clusters)

```
input  : An element set $E^A \subset E$
output: $\{E_i^A\}_{i=1}^n$ as disjoint subsets of $E^A$
$E_r \leftarrow E$
/* Here $E_r$ represents the remaining elements of $E$ to be
   processed.                                                    */
$i = 1$
while $E_r \neq \emptyset$ do
    Take any $e \in E_r$
    Let $N_e := \{e' \in E^A | e' \neq e, e'$ shares a node with $e\}$
    Let $E_i^A = \{e\} \cup N_e$ and $E_{i,r}^A = E_i^A \setminus \{e\}$
    while $E_{i,r}^A \neq \emptyset$ do
        Take any $e' \in E_{i,r}^A$
        $E_i^A \leftarrow E_i^A \cup N_{e'}$
        $E_{i,r}^A \leftarrow E_{i,r}^A \setminus \{e'\}$
    $E_r \leftarrow E_r \setminus E_i^A$
    $i \leftarrow i + 1$
```

Spline construction For clarity here we will first present the case of a simple crack. We sequentially number the knots of K following their connectivity given in I from 1 to N, and let $\boldsymbol{x}_i$ be the Cartesian coordinates of the ith knot. We then connect the knots using straight segments (chords), and define the cumulative chord length coordinate $s \in [s_1, s_N]$ with $s_1 = 0$, $s_i - s_{i-1} = |\boldsymbol{x}_i - \boldsymbol{x}_{i-1}|$, $i = 2, \dots, N$. Finally, we build $\boldsymbol{\gamma} : [s_1, s_N] \to \mathbb{R}^2$ as cubic spline functions such that $\boldsymbol{\gamma}(s_i) = \boldsymbol{x}_i$, $i = 1, \dots, N$, and set, as the initial guess, $\Gamma^{(0)} = \boldsymbol{\gamma}([s_1, s_N])$. Here we specify the end conditions following Rangarajan et al. [2015]:

$$\frac{d\boldsymbol{\gamma}}{ds}(s_1) = \frac{\boldsymbol{x}_2 - \boldsymbol{x}_1}{s_2 - s_1}, \quad \frac{d\boldsymbol{\gamma}}{ds}(s_N) = \frac{\boldsymbol{x}_N - \boldsymbol{x}_{N-1}}{s_N - s_{N-1}}. \tag{4.9}$$

In the case of a branched crack, spline functions are constructed separately for each branch to obtain $\Gamma^{(0)}$.

Spline optimization The last step is to minimize (4.5) to obtain an optimized cubic spline. The objective function is taken as (4.5), with the knot locations $\{\boldsymbol{x}_i\}$ as arguments. To improve the solvability, we allow most knots to move only in the direction normal to $\Gamma^{(m)}$. More precisely, for the mth iteration, $m = 0, 1, \ldots$, we impose the following constraints to the optimization problem (see also Figure 4.9):

$$\left[\boldsymbol{x}_i^{(m+1)} - \boldsymbol{x}_i^{(m)}\right] \cdot \left(\frac{d\gamma}{ds}(s_i)\right)^{(m)} = 0, \quad \forall k_i \in K \setminus K_{\mathrm{tjc}}. \tag{4.10}$$

Here, $K_{\mathrm{tjc}} \subset K$ denotes the set of crack tip knots and junction knots, along with the knots at which the curvature of the crack path is high enough, i.e., knots k_i such that

$$\kappa_i \alpha h \geq \beta, \tag{4.11}$$

where κ_i is the curvature of $\Gamma^{(m)}$ at knot k_i, αh is an indication of the knot spacing, and $\beta > 0$ is a parameter to be specified, which takes a value of 0.4 in subsequent numerical examples. In practice we obtain the list of knots in K_{tjc} according to $\Gamma^{(0)}$ and do not update it for later iterations for the sake of simplicity.

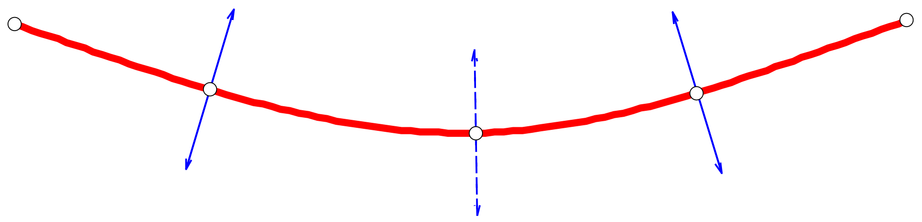


Figure 4.9: Constraints for the optimization process. Knots not in K_{tjc} are constrained to move only along the local normal direction as shown with solid or dashed arrows. Note that whether an interior knot (a knot other than the crack tip knots and the junction knots) belongs to K_{tjc} depends on the value of β. For example, the middle knot above may or may not need to be constrained, and hence is marked with dashed arrows.

This optimization process can be implemented using a black-box optimization toolbox, as shown in Algorithm 9. Each time upon obtaining a converged solution from this toolbox, the constraints are adjusted according to (4.10) again to initialize the next iteration, until the following tolerance condition is satisfied:

$$\begin{aligned} &\sqrt{\sum_{\forall k_i \in K} \left\| \boldsymbol{x}_i^{(m+1)} - \boldsymbol{x}_i^{(m)} \right\|_2^2} \leq \texttt{TolX} \\ \text{or } &\left| N_d \left[d_{\Gamma^{(m+1)}} \right] - N_d \left[d_{\Gamma^{(m)}} \right] \right| \leq \texttt{TolFun}, \end{aligned} \tag{4.12}$$

where $\|\cdot\|_2$ denotes the ℓ^2-norm (i.e., Euclidean norm) of a vector, `TolX` is the termination tolerance on the argument in MATLAB optimization toolbox `fmincon`[1], and `TolFun` is the termination tolerance on the function value. In other words, we will enforce the same termination tolerances for `fmincon` and for this constraint update. The outcome of the optimization process is depicted in Figure 4.10a.

Optionally, we can re-sample to update the knot locations to render a spline representation with almost equal spacings of the knots (strictly speaking, equal spacings of chord length coordinate s among the knots), see Figure 4.10b.

Algorithm 9: Algorithm to optimize the splines with updating constraints

```
input  : $\Gamma^{(0)}$
output: $\Gamma$
Set $m = 0$
while (4.12) is not satisfied do
    Update the constraints in (4.10) for all $k_i \in K \setminus K_{\text{tjc}}$
    Minimize (4.5) with the knot locations $\{\boldsymbol{x}_i\}$ as arguments subjected to the
    constraints (4.10). The resulting crack path is denoted $\Gamma^{(m+1)}$.
    $m \leftarrow m + 1$
$\Gamma \leftarrow \Gamma^{(m)}$
```

4.5 Representative examples

In this section, we present three representative sets of examples to demonstrate the performance of the proposed algorithm for crack identification. These examples consist of a curved crack, a branched crack, and two non-intersecting cracks. Moreover,

[1]The optimization process throughout this work is conducted in MATLAB R2015a.

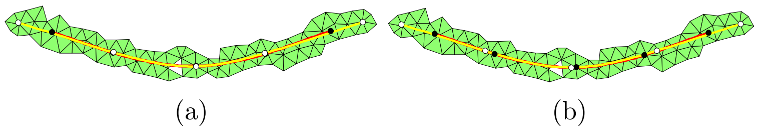


Figure 4.10: Schematics showing the optimization step to obtain Γ. (a) The initial guess of the crack path $\Gamma^{(0)}$ is shown in red with the knots in white and the optimized crack path Γ is in yellow with the knots in black. In this example, the initial and optimized sets of knots almost completely overlap except for those representing the crack tips. Here the termination tolerances `TolX` and `TolFun` are set to $1 \cdot 10^{-6}$ and $1 \cdot 10^{-7}$, respectively. (b) As an optional step, we re-sampled the spline to obtain a new set of knots with almost equal spacings and repeated the optimization step. The end results are shown as the yellow curve with the knots in black, to compare with $\Gamma^{(0)}$ as a red curve with the knots in white. In both cases, the crack tip locations are significantly improved by the optimization step.

for each example we provide a parameter study for readers interested in implementing the algorithm in MATLAB. Note that we did not provide the derivative of the objective function with respect to the arguments, for the sake of simplicity.

To better evaluate our results, we will follow this procedure to obtain the phase field d as input for all examples: We first assume a crack path, denoted Γ^* (to be specified in each subsection). Afterwards, we obtain the equivalent phase field d_{Γ^*} from (4.4). With that, we identify the bounding element set E by applying Algorithm 6 to d_{Γ^*}. Next, with the mesh $\mathscr{T}_h$, we minimize (4.2) with the nodal values of d_{Γ^*} in Ω_E and zero displacement field $\boldsymbol{u} = \boldsymbol{0}$ as constraints. The phase field solution d of this minimization problem will be taken as the input phase field for the examples. In other words, d and d_{Γ^*} necessarily coincide only at the nodes within Ω_E.

An advantage of setting the phase field input d as above is that we can regard Γ^* as the "exact solution" against which we can assess the output of the algorithm Γ. On the other hand, we did not directly use d_{Γ^*} as inputs to avoid too ideal a situation. To quantify the error of the proposed crack identification algorithm, we define the following parameter that relates Γ and Γ^* as the *normalized error*:

$$\frac{e}{h} := \frac{1}{h}\sqrt{\frac{\sum_{k_i \in K} \|\boldsymbol{x}_i - \pi(\boldsymbol{x}_i)\|^2}{N}}, \tag{4.13}$$

where $\{\boldsymbol{x}_i\}$ are the knots of Γ and $\pi(\boldsymbol{x}_i)$ is the closest point on Γ^* from $\boldsymbol{x}_i$, except when $\boldsymbol{x}_i$ is a crack tip (or junction) knot of Γ, in which case $\pi(\boldsymbol{x}_i)$ is the corresponding crack tip (junction) knot of Γ^*.

Throughout this section, we will adopt the default values of the parameters and options given in Table 5.1, unless otherwise noted.

Table 4.2: Default parameter values for the examples

Name	Symbol	Value
Mesh size (largest diameter of elements of $\mathscr{T}_h$)	h	0.02
Regularization length scale	ℓ	0.1
Critical phase field value	d_{crit}	0.6
Resolution parameter	α	4.0
Parameter in (4.11)	β	0.4
Norm for defining the objective function in (4.5)	$\lVert\cdot\rVert$	$\lVert\cdot\rVert_{L_2(\Omega)}$

4.5.1 Simple curved crack

This example is chosen to prove that the algorithm is capable of identifying cracks with a considerable curvature respect to the resolution $(\alpha + 1/2)h$, see Figure 4.11. Here in separate snapshots the element sets E_0 and E, the initial guess for the crack path, $\Gamma^{(0)}$, the output of the algorithm Γ, and the assumed crack path Γ^* are shown. In this example, Γ^* is taken as the cubic spline passing through the points $(0.3, 0.5)$, $(0.5, 0.2)$, and $(0.7, 0.5)$ with the end conditions given by (4.9). As seen from the figure, the optimization process clearly leads to a better crack path (Γ versus $\Gamma^{(0)}$) with regard to Γ^*.

Figure 4.12 shows the same example except that α is set to be 2. Therefore, the initial guess $\Gamma^{(0)}$ is expected to be closer to Γ^*.

This example, as well as the upcoming ones, has been solved using the MATLAB optimization toolbox `fmincon`. We have tested three algorithm options `active-set`, `interior-point`, and `SQP` (sequentially quadratic programming), and summarized

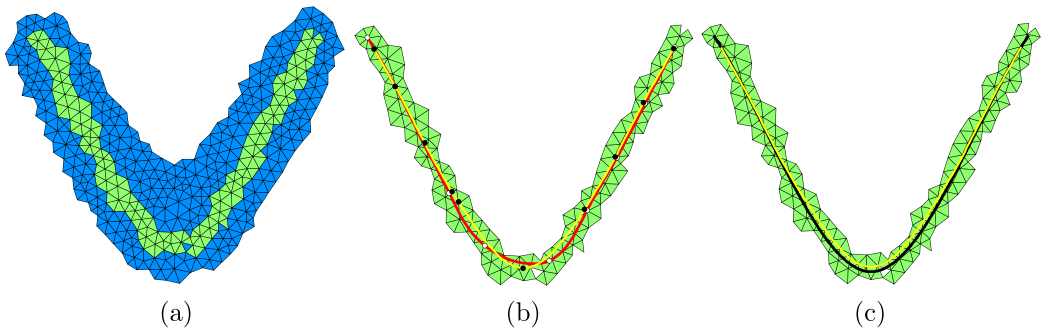


Figure 4.11: A simple curved crack. (a) The element set E_0 corresponding to d_{crit} (see Algorithm 6) is shown in either blue or green. The element set E as obtained from the non-maximum suppression algorithm (Algorithm 6) defining the bounding area is shown in green. The crack path Γ is sought in the green region. (b) The initial set of knots as identified by Algorithm 7 are shown in white. The cubic spline, in red, is then constructed as $\Gamma^{(0)}$, the initial guess of Γ. Afterwards, an optimization process is followed with the constraints given in (4.10) to optimize the knot locations (in black) and therefore Γ (in yellow). (c) Comparison of Γ^*, the "exact solution" in black, and Γ, the identified crack path from the proposed method in yellow. Good agreement is observed.

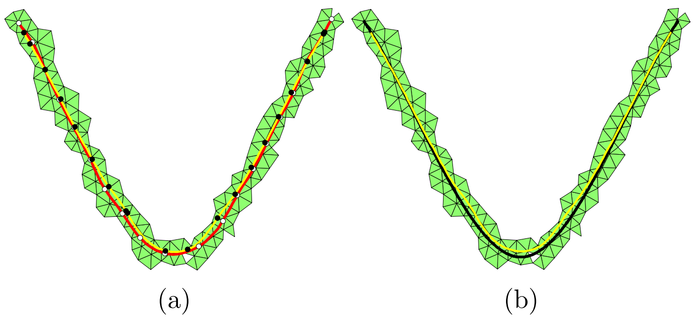


Figure 4.12: Simple curved crack. Results with the same example as that in Figure 4.11, except for the resolution parameter α which is set to 2.0 instead of the default value. As a result, more knots are created, but $\Gamma^{(0)}$ obtained is closer to Γ to start with. The color codes are the same as Figure 4.11. (a) The cubic spline, in red, is $\Gamma^{(0)}$, and the yellow curve is Γ. The knots are also shown for both curves. (b) Comparison of Γ (in yellow) and Γ^* (in black). Good agreement is observed.

the performance in Table 4.3. It can be observed that with a smaller α value, much more iterations are needed, which may be due to the larger number of optimization variables, as there are more knots. Judging from the gain in the accuracy, this higher resolution may not be needed for the example at hand.

Table 4.3: Performance study of the example of a simple curved crack. Here the termination tolerances `TolX` and `TolFun` are set to $1 \cdot 10^{-6}$ and $1 \cdot 10^{-5}$, respectively.

α	Minimization algorithm option	Number of evaluations of the objective function	Number of constraint updates	Normalized error given by (4.13)
2.0	`Interior-point`	4926	2	0.33
	`Active-set`	617	1	0.33
	`SQP`	2934	2	0.33
4.0	`Interior-point`	1832	2	0.43
	`Active-set`	357	1	0.43
	`SQP`	1058	2	0.43

We also tested the setting $\|\cdot\| = (\ell\Gamma_\ell[\cdot])^{1/2}$ for this example, with the factor ℓ introduced to make this option comparable with the default choice. The algorithm options `interior-point` and `SQP` require 1786 and 596 function evaluations and 3 and 2 constraint updates, respectively. However, the option active-set did not converge with more than 2552 function evaluations.

4.5.2 Branched crack

In this example, we set Γ^* to be a branched crack shown in Figure 4.1. This crack consists of three straight segments with the junction at $(0.5, 0.5)$, and the three crack tips at $(0.3, 0.5)$, $(0.7, 0.7)$, and $(0.7, 0.3)$. Due to the constraints given by (4.10), in the optimization stage, most knots can only move in a direction perpendicular to the curve's path. Figure 4.13 shows the element sets E_0, E, the initial guess of the crack path $\Gamma^{(0)}$, the converged crack path Γ, and the assumed crack path Γ^*. It is evident that the proposed method is able to detect a crack path with junctions. A performance study is given in Table 4.4.

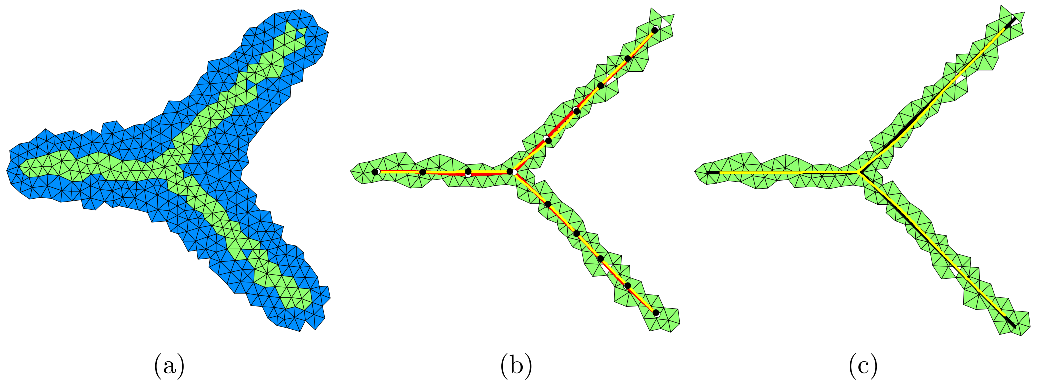


Figure 4.13: Branched crack example. (a) The element set E_0 corresponding to d_{crit} (see Algorithm 6) is shown in either blue or green. The element set E as obtained from the non-maximum suppression algorithm (Algorithm 6) defining the bounding area is shown in green. The crack path Γ is sought in the green region, which is a thin branched layer of elements. (b) The initial set of knots as identified by Algorithm 7 are shown in white. The cubic spline, in red, is then constructed as $\Gamma^{(0)}$, the initial guess of Γ. Afterwards, an optimization process is followed with the constraints given in (4.10) to optimize the knot locations (in black) and therefore Γ (in yellow). It is reminded that for a branched crack (as identified as $\Gamma^{(0)}$), constraints in (4.10) are not applied on the junction knot nor on the crack tip knots. (c) Comparison of Γ^*, the "exact solution" in black, and Γ, the identified crack path from the proposed method in yellow.

4.5.3 Multiple cracks

We next showcase an example where Γ^* is made of two cracks with a certain clearance to demonstrate that the proposed algorithm can distinguish multiple cracks. In other words, if the bounding element set E is made of groups of mutually not connected (not sharing any node) elements and these groups are well separated, characterized by the resolution $(\alpha + 1/2)h$, then they can be detected as multiple cracks.

For this example, let Γ^* consist of a horizontal straight crack and a vertical straight crack. The coordinates of the crack tips are: $(0.3, 0.5), (0.5, 0.5)$ for the horizontal crack, and $(0.6, 0.3), (0.6, 0.7)$ for the vertical one. Therefore, the distance between

Table 4.4: Performance study of the example of a branched crack. Here the termination tolerances `TolX` and `TolFun` are set to $1 \cdot 10^{-6}$ and $1 \cdot 10^{-7}$, respectively.

Minimization algorithm option	Number of evaluations of the objective function	Number of constraint updates	Normalized error given by (4.13)
`Interior-point`	435	1	0.45
`Active-set`	87	1	0.43
`SQP`	435	1	0.45

the two crack tips is 0.1, which is bigger than $(\alpha + 1/2)h = 0.09$. The results of this example are depicted in Figure 4.14. A performance study is given in Table 4.5.

Table 4.5: Performance study of the example of multiple cracks. Here the termination tolerances `TolX` and `TolFun` are set to $1 \cdot 10^{-6}$ and $1 \cdot 10^{-5}$, respectively.

Minimization algorithm option	Number of evaluations of the objective function	Number of constraint updates	Normalized error given by (4.13)
`Interior-point`	350	1	0.47
`Active-set`	100	1	0.38
`SQP`	350	1	0.47

4.6 Conclusion

We have designed a crack detection algorithm to identify the crack path from the phase field approach to fracture. The algorithm is capable of identifying not only a simple crack but also multiple cracks and a branched crack, as demonstrated in the numerical examples. We have also provided a performance study for each example in terms of a comparison of optimization algorithms available in MATLAB.

The proposed algorithm is facilitated by the non-maximum suppression method, which limits the search of the crack path to a small subset of the computational

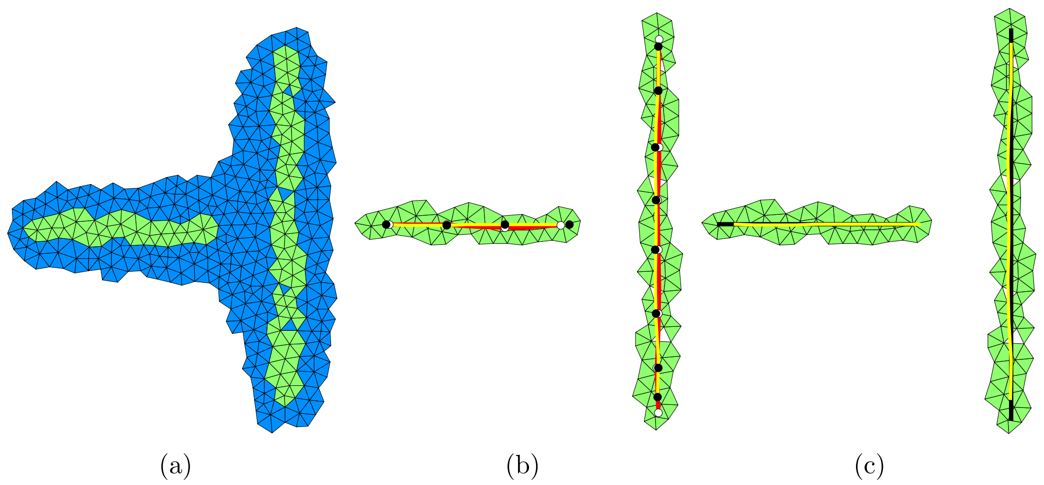


Figure 4.14: Multiple crack example. (a) The element set E_0 corresponding to d_{crit} (see Algorithm 6) is shown in either blue or green. The element set E as obtained from the non-maximum suppression algorithm (Algorithm 6) defining the bounding area is shown in green. As seen, the area made of elements in E_0 is connected while that made of elements in E contains two separate zones of elements. As the distance between the two green zones is larger than the resolution, the algorithm is capable of detecting them as two cracks (instead of a branched one). (b) The initial set of knots as identified by Algorithm 7 are shown in white. The cubic spline, in red, is then constructed as $\Gamma^{(0)}$, the initial guess of Γ. Afterwards, an optimization process is followed with the constraints given in (4.10) to optimize the knot locations (in black) and therefore Γ (in yellow). Note that no constraint is applied on the crack tip knots. (c) Comparison of Γ^*, the "exact solution" in black, and Γ, the identified crack path from the proposed method in yellow.

domain, and which, more importantly, provides an approximation of the underlined topology of the crack path being sought.

Another key ingredient is the so called *equivalent phase field*, for expressing the phase field equivalent (to some degree) to some crack set Γ. This definition enables us to determine a way to evaluate how much the identified crack path is away from being equivalent to the given phase field, and thus renders the crack identification problem variational.

The best possible resolution for this crack detection algorithm is related to the mesh size. As a consequence, the method can distinguish cracks close to each other up to a certain distance. We did not show negative examples but if two cracks are too close to each other, they may be identified as one crack, possibly with branches.

The developed algorithm can be combined with one on crack opening, for more elaborate interpretation of phase field simulations. This will be the topic of Chapter 5.

Chapter 5

Variational approach to calculate the crack opening

5.1 Introduction

In this chapter, we aim to calculate the crack opening from a phase field approach to fracture. This can be considered as a second step for identifying the crack geometry, see Chapter 4 for the crack path identification.

The calculation of crack openings can be useful in some applications. One example is to estimate the durability of concrete structures. The cracks are more likely exposed to fluid or corrosive chemicals inside, hence, it is beneficial to investigate the openings [Pijaudier-Cabot et al., 2009].

Another example is hydraulic fracturing where the high pressure liquid inside the fracture causes further propagation. In most studies, the fluid pressure is obtained from a lower dimensional lubrication equation [Irzal et al., 2013, Schrefler et al., 2006]. In order to solve the fracture flow from the lubrication equation, the crack openings need to be known.

One method was proposed by Dufour *et al.* [Dufour et al., 2012] where the idea is to compare two effective non local strain: The one that controls damage and the other derived from a strong discontinuity analysis. The estimation was illustrated in a one-dimensional example close to failure.

In this chapter, we offer a variational approach to calculate the openings. The idea is to stabilize the effect of both the smeared crack and its equivalent explicit one by means of a virtual work equilibrium. Therein, phase field gradient is used as a weighted function to calculate the openings from displacement profiles $\boldsymbol{u}$. In Section 5.3, we provide three representative examples to test the performance of our method. Throughout this chapter, we frequently use the terminologies used in Chapter 4

5.2 Approach to calculate the crack opening

In this section, we set up the problem of calculating the openings from phase field descriptions of crack. First, we introduce a variational formulation to replace the discrete definition of a jump set with a sufficient condition. Next, we present our variational approach which is based on the principle of virtual work. Finally, we discuss the special case of branched cracks where the method needs to be slightly modified to tackle the case.

5.2.1 The special case

Consider $\Omega = \mathbb{R}^2$. Let $\Gamma \subset \mathbb{R}^2$ be an identified crack path for $d(\boldsymbol{x}) = \exp\left(-\frac{|y|}{\ell}\right)$. Also, let $\boldsymbol{u}_y = u\boldsymbol{e}_y$ be such that $\boldsymbol{u}_y(y) \to \boldsymbol{u}^{\pm}$ when $y \to \pm\infty$, see Section 4.2. Now, we define H_ℓ, as the openings, by a variational formulation:

$$H_\ell[\boldsymbol{u}, d] := \lim_{\ell\to 0} \int_{-\infty}^{\infty} -\boldsymbol{u}_y \frac{\partial d}{\partial y}\, dy = \boldsymbol{u}^+ - \boldsymbol{u}^-, \tag{5.1}$$

One sufficient condition for (5.1) is as below:

$$\lim_{\ell\to 0} \int_{0}^{\infty} \left(\left(\boldsymbol{u}^+ - \boldsymbol{u}^-\right) - \left(\boldsymbol{u}_y(y) - \boldsymbol{u}_y(-y)\right)\right) \frac{\partial d}{\partial y}\, dy = 0. \tag{5.2}$$

5.2.2 Variational formulation

With respect to (5.1), we present a general approach to calculate the openings from any phase field topology. Let $(\boldsymbol{u}, d)$ be a solution of (4.2), or an approximation to it. We equivalently identify $(\boldsymbol{u}, \Gamma)$ as an approximation of (4.1). Readers are referred to Chapter 4 for more explanations.

The main body of the approach is as follows: Let be δP a virtual pressure. We take virtual work by δP on the identified crack (Γ) be equal to the one on the smeared crack (d). Therefore, the following variational equality takes place:

$$\int_{\Gamma} \delta P(s) \cdot H(s)\, ds = \int_{\Omega} -\left(\delta P \circ \gamma^{-1} \circ \pi(\boldsymbol{x})\right) \left((\boldsymbol{u} - \boldsymbol{u}_{ref}) \cdot \nabla d\right)\, d\Omega. \tag{5.3}$$

where $\boldsymbol{u}_{ref}$ is the rigid body motion of the sample. This term can be computed as below.

$$\boldsymbol{u}_{ref} = \frac{1}{\|\partial\Omega\|} \int_{\partial\Omega} \boldsymbol{u}\, ds. \tag{5.4}$$

Note that in (5.3), $\pi(\boldsymbol{x})$ is the closest point projection of $\boldsymbol{x}$ on Γ, and $\boldsymbol{\gamma} : [s_1, s_N] \to \mathbb{R}^2$ is defined as a map, see Chapter 4.

5.2.3 Numerical discretization

In this section, we adopt standard procedures to obtain Galerkin approximation for the problem in hand. We discretize the domain Ω with a mesh family $\{\mathscr{T}_h\}$, each member characterized by the mesh size h. We approximate δP and H merely on Γ with the standard first-order finite element basis functions associated with the knots $(k \in K)$. Note that since the virtual pressure is free to choose, we merely use the basis functions to approximate it. See below:

$$H(s) = \sum_{i \in K} H_i N_i(s), \tag{5.5a}$$

$$\delta P(s) = \sum_{i \in K} \delta P_i N_i(s). \tag{5.5b}$$

Standard Galerkin approximations yield the following matrix form for (5.3):

$$\mathbf{M}\mathbf{H} = \mathbf{F}, \tag{5.6}$$

where $\mathbf{H} = \{H_P\}$ is a vector that contains the nodal degrees of freedom for H. The explicit expressions of the matrices involved in (5.6) read:

$$\sum_{i\in K}\left(\sum_{j\in K}\int_\Gamma \left(N_i(s)N_j(s)\right) H_j \, ds\right) = -\sum_{j\in K}\int_\Omega \left(N_j \circ \gamma^{-1} \circ \pi(\boldsymbol{x})\right)\left(\boldsymbol{u}\cdot\nabla d\right)\, d\Omega. \tag{5.7}$$

5.2.4 The case of branching

In general, the method work for any smeared representative of a crack. However, from our experience, this way sometimes may not be accurate for the case of a branched crack, as the phase field topology is too smooth near the junctions. Hence, for the sake of accuracy, we modified the phase field topology and the region of integration in (5.3).

We proceed as follows: Let j be a junction and aj be a simple identified branch for a given phase field, see Figure 5.5. For aj, first we identify the region $\Omega_{aj} \subset \Omega$, which is the set of all points in Ω closer to aj than other branches, and in order to calculate the openings at aj, we take Ω_{aj} for (5.3) as the region integration in place of Ω. Second, we replace the current topology of the phase field by that of the *equivalent phase field* of aj.

Moreover, we slightly modify the phase field profile as follows: For point j we virtually extend aj along the same direction to $j\prime$, one knot spacing apart from j. Hence, the new branch is $ajj\prime$. And, we consider the equivalent phase field for $ajj\prime$ as the one to be incorporated in (5.3). This way will also prevent the high gradient phase field at j, now will be like a crack tip, to unstabilize the results. The results, shown in Section 5.3.2, prove that the current way leads to an opening profile aligned with that of the chosen benchmark.

5.3 Numerical examples

In this section, we present three representative sets of numerical examples to demonstrate the performance of our variational approach. The examples consist of a curved crack, a branched crack, and multiple cracks.

In this section, we investigate a square plate with some initial cracks. Two geometric setups are depicted in Figure 5.1. For the simple curved case and the branched case, the specimen is under a direct tension test in which a displacement with magnitude $\boldsymbol{u}_D$ is imposed on the top and bottom edges (Figure **??**) while for the case of multiple cracks, the loading is imposed on all edges (Figure **??**).

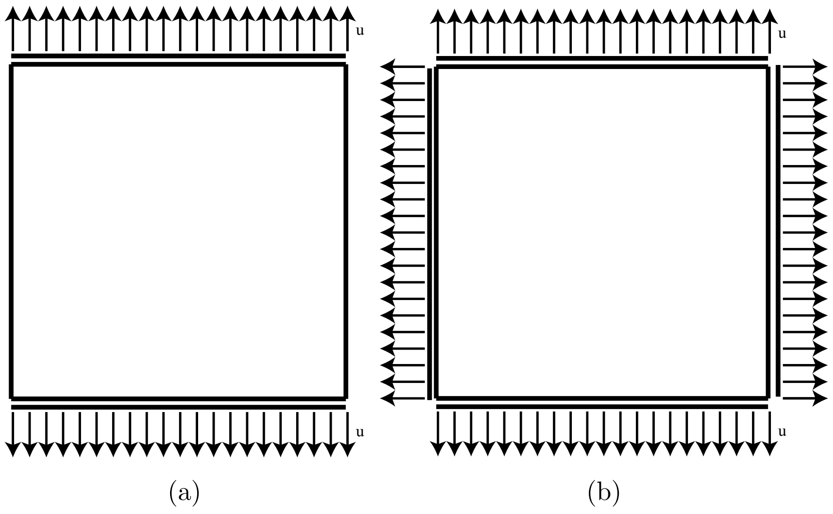


Figure 5.1: Schematic of a cracked square plate under a single-edge-notched tension test. A uniform displacement loading $\|u_D\|$ normal to the edges is applied on (a) top and bottom edges for the cases in 5.3.1 and 5.3.2, (b) on the whole edges for the case of Section 5.3.3. Also, some initial crack is assumed for each set of examples.

Throughout this section, in order to solve for (4.2), we will adopt the values of the parameters given in Table 5.1, unless otherwise noted.

To better evaluate our results, we will follow this procedure to obtain the displacement $\boldsymbol{u}$ field and the phase field d as input for all examples: We first assume a crack path, either Γ or Γ^*, see Chapter 4. Afterwards, we obtain the equivalent phase field d_{Γ^*} from (4.3). With that, we identify the bounding element set E to

Table 5.1: Default parameter values for the examples

Name	Symbol	Value
Lamé constant	λ	120GPa
Shear modulus	μ	80GPa
Critical energy release rate	g_c	2.7×10^{-3}kN/mm
length scale ratio	h/ℓ	0.2
Boundary displacement	u_D	10^{-3}mm

d_{Γ^*}. Next, with the mesh $\mathscr{T}_h$, we minimize (4.2) with the nodal values of d_{Γ^*} in Ω_E. We impose boundary displacement $\boldsymbol{u} = \boldsymbol{u}_D$ so that a tension test be modeled. The approximation is then taken as input for the subsequent examples.

Let $\boldsymbol{u}^*$ be the minimizer of $\Pi(\cdot, \Gamma^*)$ with the constraint $\boldsymbol{u}^* = \boldsymbol{u}_D$ on $\partial\Omega_D$, where Π is defined in (4.1). We take $\boldsymbol{u}^*$ as a benchmark to evaluate our calculation of the crack opening from the approximation of (4.2) for the subsequent examples. Let $\boldsymbol{u}$ be obtained from an approximation of (4.2). In order to justify our benchmark, we define ε as (5.8):

$$\varepsilon := \frac{\|\boldsymbol{u} - \boldsymbol{u}^*\|_{L_2(\Omega)}}{\|\boldsymbol{u}^*\|_{L_2(\Omega)}}, \tag{5.8}$$

An advantage of setting the phase field input d as above is that Γ^* is used in a consistent way for the calculation of both $\boldsymbol{u}$ and $\boldsymbol{u}^*$.

5.3.1 Simple curved crack

This example is chosen to prove that the algorithm is capable of calculating the openings with a considerable curvature, see Figure 5.2. Here in separate snapshots the element set E, and the output from the crack detection algorithm Γ, and the assumed crack path Γ^* are shown. In this example, Γ^* is taken as the cubic spline passing through the points $(0.3, 0.5)$, $(0.5, 0.2)$, and $(0.7, 0.5)$ with appropriate end conditions following [Rangarajan et al., 2015].

Figure 5.3 plots the crack opening by $\boldsymbol{u}$, labeled as *variational approach* and $\boldsymbol{u}^*$ as *benchmark*.

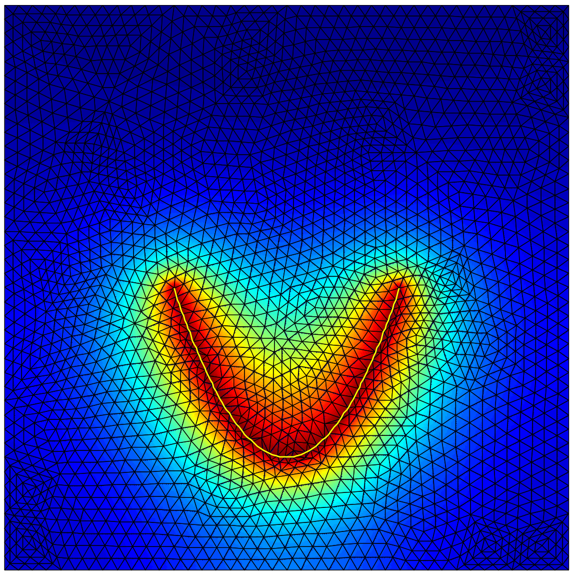


Figure 5.2: A simple curved crack. Phase field topology, and Γ^*, the "exact solution" in yellow, are shown. There is a conforming mesh to Γ^*.

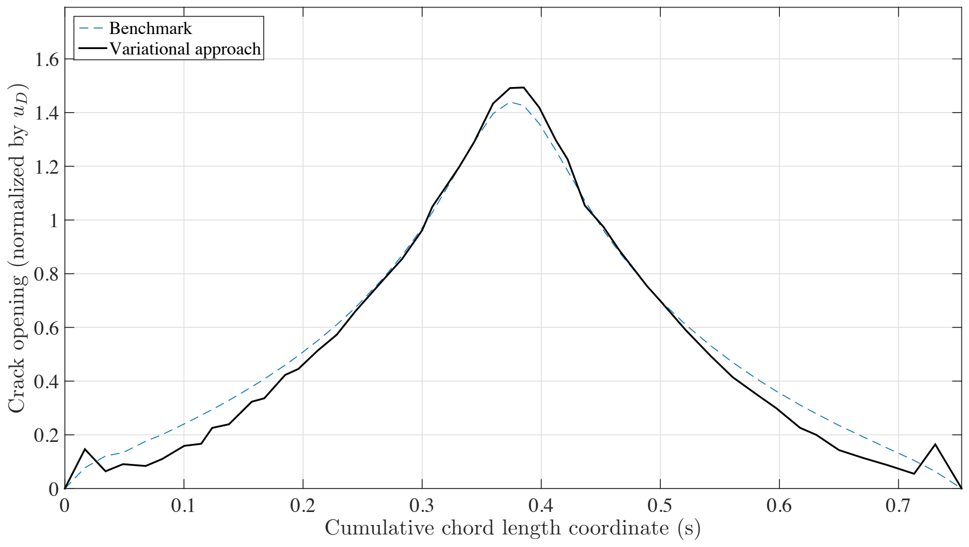


Figure 5.3: A simple curved crack. A comparison of opening by $\boldsymbol{u}$ and $\boldsymbol{u}^*$ is made. Good agreement is observed. $\boldsymbol{u}_{ref}$ is calculated by (5.4).

5.3.2 Branched crack

In this example, we consider a phase field d associated to a branched crack, see Figure 5.4. This crack consists of three straight segments with the junction at $j = (0.5, 0.5)$, and the three crack tips at $a = (0.3, 0.5)$, $b = (0.7, 0.7)$, and $c = (0.7, 0.3)$.

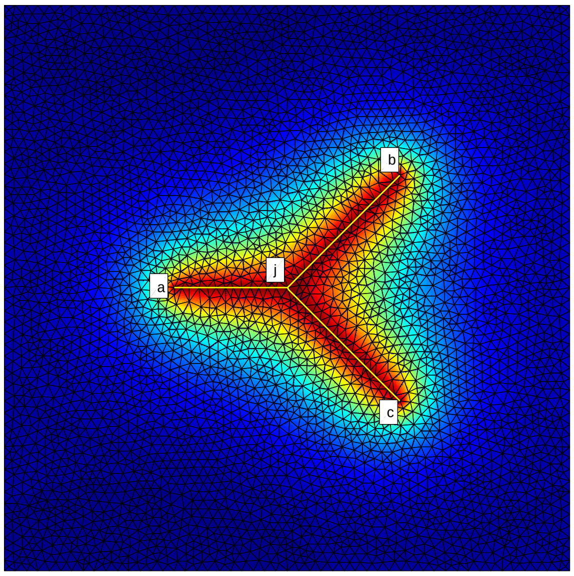


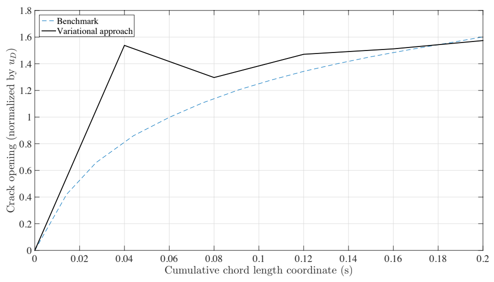
Figure 5.4: A branched crack. Phase field diagram and Γ^*, the "exact solution" in yellow, are shown. There is a conforming mesh to Γ^*. We labeled the junction and the three crack tips.

Figure 5.5 depicts the openings of the three segments for a set of knots spacing $2h$. Also shown is the openings for the benchmark. Readers are referred to 5.2.4 for further explanations about the special issues for the case of branching.

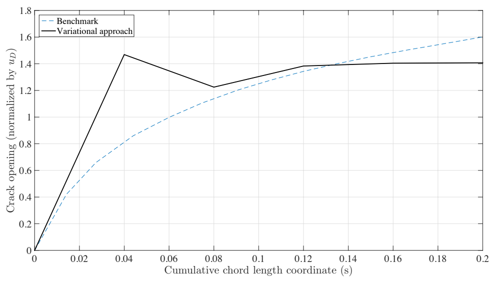
5.3.3 Multiple cracks

In this example, we calculate the opening for two non-intersecting cracks with a certain distance. Phase field topology is shown with Γ and Γ^*, see Figure 5.6.

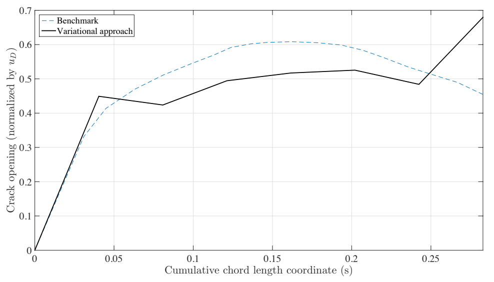
Figure 5.7 illustrates the openings for the horizontal (left) and vertical crack (right), compared to ω^*. Three sets of knots are shown for each segment. It can be seen that the ones with smaller knot spacings tends more to the benchmark.



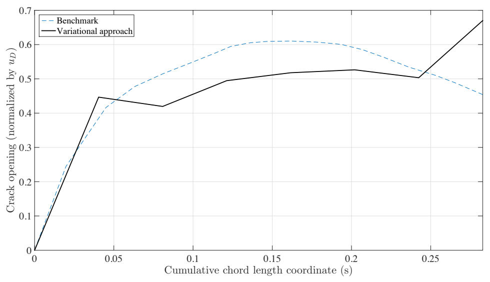
(a)



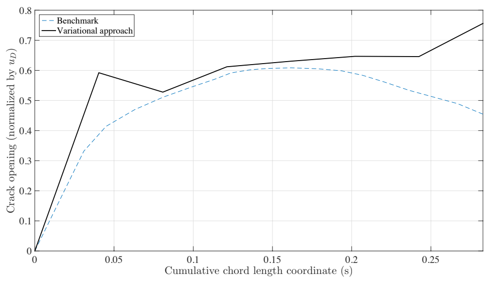
(b)



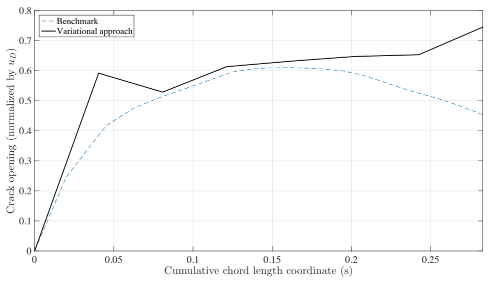
(c)



(d)



(e)



(f)

Figure 5.5: A branched crack. The openings for the segments aj, bj, and cj are subsequently shown in (left) for model B, and (right) for model C. Curves in blue are for the benchmark.

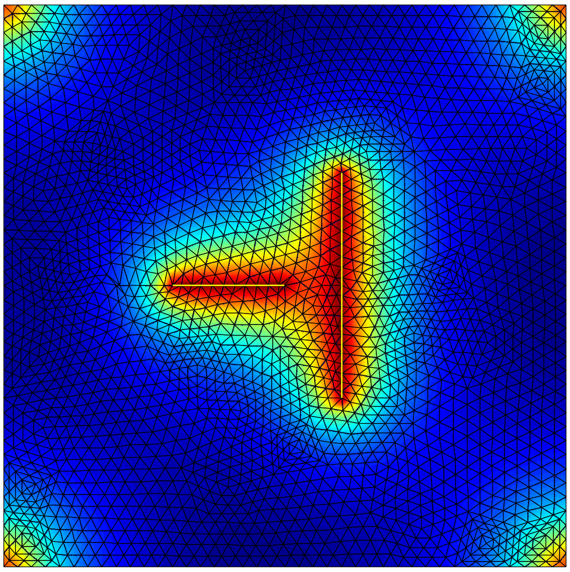


Figure 5.6: Multiple cracks. Phase field diagram and Γ^*, the "exact solution" in yellow, are shown. There is a conforming mesh to Γ^*.

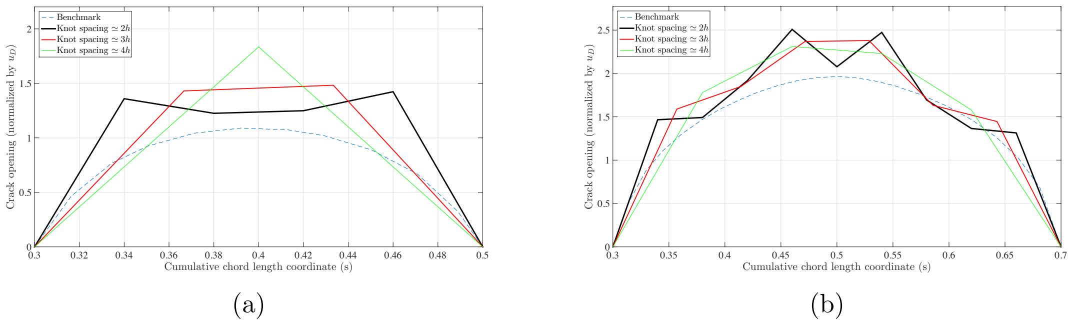


Figure 5.7: Multiple cracks. The openings along the horizontal and vertical cracks are shown in (a) and (b), accordingly. Also shown is ω^* in blue dashed lines. The results are presented for three sets of knots with different knot spacings for each segment.

5.4 Conclusion

We are motivated to calculate the crack opening in some applications like the study of durability in concrete structures, or computationally to solve for lubrication equation in hydraulic fracturing.

We have designed a variational way to calculate the opening from a smeared crack representation. The idea is to stabilize the effect of both the smeared crack and its equivalent explicit one by means of a virtual work equilibrium. This would be attractive since there is no need to consider the explicit description of a discontinuous displacement field in the computational model.

The method works fine even for the case of a crack with high curvature. For a branched case, for sake of more accuracy, we slightly modify the basic equilibrium.

We also output the results for different knot spacings and two phase field models.

Chapter 6

Conclusion

- We designed a GPU algorithm for the phase field approach to fracture, where the displacement field is a solution of an elastodynamic problem, while the phase field is updated according to a rate-dependent formulation, so that we obtain a fully explicit scheme. The explicit method fits nicely with the GPU architecture, especially in terms of the thread and memory hierarchies. This is a strong motivation for adopting a fully explicit method.

- To ensure stability, we designed a time adaptivity strategy to account for the decreasing critical time step during the evolution of the field. The time step does not decrease until the latest stages of fracture when it becomes significantly smaller. As will be seen in the scalability study in A.2, this time adaptivity scheme maintains the overall scaling of the method, which will be as good as that of an implicit scheme, if not better.

- We did a comprehensive study to compare the three popular phase field models undergoing tension and shear loadings with high and low rates to see interesting phenomena such as branching and crack nucleation in the simulations. In conclusion, generally model B and model C provide more physical predictions than model A for most materials. But there is no clear winner between models B and C in terms of the phenomena they are able to capture.

- We designed a crack detection algorithm to identify the crack path in a variational way from the phase field approach to fracture. The algorithm is capable of identifying not only a simple crack but also multiple cracks and a branched crack, as demonstrated in the numerical examples. We have also provided a performance study for each example in terms of a comparison of optimization algorithms available in MATLAB.

- We proposed a variational way to calculate the crack openings from a smeared crack representation. We verified our calculations with a proper benchmark.

Future work

Currently, we are working on a new phase field formulation of hydraulic fracture. For the solid phase, we use the phase field approaches. For the liquid phase, we first identify the crack geometry, the path and the openings, using our variational approach. Next, we solve the lubrication equation for the liquid phase. As an advantage, since the crack surface is identified, there would not be a challenge to impose the pressure loading on the solid phase while we benefit from the phase field approaches for brittle fracture.

Bibliography

F. Amiri, D. Millán, Y. Shen, T. Rabczuk, and M. Arroyo. Phase-field modeling of fracture in linear thin shells. *Theoretical and Applied Fracture Mechanics*, 69: 102–109, 2014.

H. Amor, J.-J. Marigo, and C. Maurini. Regularized formulation of the variational brittle fracture with unilateral contact: Numerical experiments. *Journal of the Mechanics and Physics of Solids*, 57:1209–1229, 2009.

J. F. Babadjian and V. Millot. Unilateral gradient flow of the Ambrosio-Tortorelli functional by minimizing movements. *Annales de l'Institut Henri Poincare (C) Non Linear Analysis*, 31(4):779–822, 2014.

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. *Computer Methods in Applied Mechanics and Engineering*, 217–220:77–95, 2012.

M. J. Borden, T. J.R. Hughes, C. M. Landis, and C. V. Verhoosel. A higher-order phase-field model for brittle fracture: Formulation and analysis within the isogeometric analysis framework. *Computer Methods in Applied Mechanics and Engineering*, 273:100–118, 2014.

M. Bottoni, F. Dufour, and C. Giry. Topological search of the crack pattern from a continuum mechanical computation. *Engineering Structures*, 99:346–359, 2015.

B. Bourdin, G. A. Francfort, and J.-J. Marigo. The variational approach to fracture. *Journal of Elasticity*, 91:5–148, 2008.

B. Bourdin, C. J. Larsen, and C. Richardson. A time-discrete model for dynamic fracture based on crack regularization. *International Journal of Fracture*, 168(2): 133–143, 2011.

B. Bourdin, C. P. Chukwudozie, and K. Yoshioka. A variational approach to the numerical simulation of hydraulic fracturing. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2012.

C. Cecka, A. J. Lew, and E. Darve. Assembly of finite element methods on graphics processors. *International Journal for Numerical Methods in Engineering*, 85(5): 640–669, 2011.

A. Chambolle, G. A. Francfort, and J.-J. Marigo. When and how do cracks propagate? *Journal of the Mechanics and Physics of Solids*, 57(9):1614–1622, 2009.

M. Charlotte, J. Laverne, and J.-J. Marigo. Initiation of cracks with cohesive force models: A variational approach. *European Journal of Mechanics - A/Solids*, 25(4): 649–669, 2006.

C. Chukwudozie, B. Bourdin, and K. Yoshioka. A variational approach to the modeling and numerical simulation of hydraulic fracturing under in-situ stresses. In *Proceedings of the 38th Workshop on Geothermal Reservoir Engineering*, 2013.

Milton N. da Silva, Jr., F. P. Duda, and E. Fried. Sharp-crack limit of a phase-field model for brittle fracture. *Journal of the Mechanics and Physics of Solids*, 61(11): 2178–2195, 2013.

G. Dal Maso and G. Lazzaroni. Quasistatic crack growth in finite elasticity with non-interpenetration. *Annales de l'Institut Henri Poincare (C) Non Linear Analysis*, 27(1):257–290, 2010.

G. Del Piero, G. Lancioni, and R. March. A variational model for fracture mechanics: Numerical experiments. *Journal of the Mechanics and Physics of Solids*, 55(12): 2513–2537, 2007.

F. Dufour, G. Legrain, G. Pijaudier-Cabot, and A. Huerta. Estimation of crack opening from a two-dimensional continuum-based finite element computation. *International Journal for Numerical and Analytical Methods in Geomechanics*, 36(16): 1813–1830, 2012.

X. Feng and A. Prohl. Analysis of gradient flow of a regularized Mumford-Shah functional for image segmentation and image inpainting. *ESAIM: M2AN*, 38(2): 291–320, 2004.

G. A. Francfort and J.-J. Marigo. Revisiting brittle fracture as an energy minimization problem. *Journal of the Mechanics and Physics of Solids*, 46(8):1319–1342, 1998.

S. Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et naturelles*, 6:749–754, 1931.

H. Gomez, A. Reali, and G. Sangalli. Accurate, efficient, and (iso)geometrically flexible collocation methods for phase-field models. *Journal of Computational Physics*, 262:153–171, 2014.

V. Hakim and A. Karma. Laws of crack motion and phase-field models of fracture. *Journal of the Mechanics and Physics of Solids*, 57(2):342–368, 2009.

M. Hofacker and C. Miehe. Continuum phase field modeling of dynamic fracture: Variational principles and staggered FE implementation. *International Journal of Fracture*, 178(1–2):113–129, 2012a.

M. Hofacker and C. Miehe. A phase field model of dynamic fracture: Robust field updates for the analysis of complex crack patterns. *International Journal for Numerical Methods in Engineering*, 93:276–301, 2012b.

T. J.R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

M. J. Hunsweck, Y. Shen, and A. J. Lew. A finite element approach to the simulation of hydraulic fractures with lag. *International Journal for Numerical and Analytical Methods in Geomechanics*, 37(9):993–1015, 2013.

F. Irzal, J.C. Remmers, J. M. Huyghe, and R. de Borst. A large deformation formulation for fluid flow in a progressively fracturing porous material. *Computer Methods in Applied Mechanics and Engineering*, 256:29 – 37, 2013.

M. Jirásek and T. Zimmermann. Embedded crack model. Part II: Combination with smeared cracks. *International Journal for Numerical Methods in Engineering*, 50 (6):1291–1305, 2001.

A. Karma, D. A. Kessler, and H. Levine. Phase-field model of mode III dynamic fracture. *Phys. Rev. Lett.*, 87:045501, 2001.

G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

D. E Keyes, D. R Reynolds, and C. S Woodward. Implicit solvers for large-scale nonlinear problems. *Journal of Physics: Conference Series*, 46(1):433–442, 2006.

D. B. Kirk and W. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach.* Elsevier, Amsterdam, 2010.

C. Kuhn and R. Müller. A continuum phase field model for fracture. *Engineering Fracture Mechanics*, 77(18):3625–3634, 2010.

C. Larsen. Models for dynamic fracture based on Griffith's criterion. In *IUTAM Symposium on Variational Concepts with Applications to the Mechanics of Materials*, volume 21 of *IUTAM Bookseries*, pages 131–140. Springer Netherlands, 2010.

J.-J. Marigo. Initiation of cracks in Griffith's theory: An argument of continuity in favor of global minimization. *Journal of Nonlinear Science*, 20(6):831–868, 2010. ISSN 0938-8974.

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(45–48):2765–2778, 2010a.

C. Miehe, F. Welschinger, and M. Hofacker. Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations. *International Journal for Numerical Methods in Engineering*, 83:1273–1311, 2010b.

C. Miehe, S. Mauthe, and S. Teichtmeister. Minimization principles for the coupled problem of Darcy-Biot-type fluid transport in porous media linked to phase field modeling of fracture. *Journal of the Mechanics and Physics of Solids*, 82:186–217, 2015.

A. Mikelić, M. F. Wheeler, and T. Wick. A phase field approach to the fluid filled fracture surrounded by a poroelastic medium. *ICES Report*, 13–15, 2013a.

A. Mikelić, M. F. Wheeler, and T. Wick. A quasi-static phase-field approach to the fluid filled fracture. *ICES Report*, 13–22, 2013b.

A. Mikelić, M. F. Wheeler, and T. Wick. Phase-field modeling of a fluid-driven fracture in a poroelastic medium. *Computational Geosciences*, 19:1171–1195, 2015a.

A. Mikelić, M. F. Wheeler, and T. Wick. A phase-field method for propagating fluid-filled fractures coupled to a surrounding porous medium. *Multiscale Modeling and Simulation*, 13(1):367–398, 2015b.

N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46(1):131–150, 1999.

K. Pham, J.-J. Marigo, and C. Maurini. The issues of the uniqueness and the stability of the homogeneous response in uniaxial tests with gradient damage models. *Journal of the Mechanics and Physics of Solids*, 59(6):1163–1190, 2011.

G. Pijaudier-Cabot, F. Dufour, and M. Choinska. Permeability due to the increase of damage in concrete: From diffuse to localized damage distributions. *Journal of Engineering Mechanics*, 135(9):1022–1028, 2009.

C. Pozrikidis. Leakage through a permeable capillary tube into a poroelastic tumor interstitium. *Engineering Analysis with Boundary Elements*, 37(4):728–737, 2013.

R. Rangarajan, M. M. Chiaramonte, M. J. Hunsweck, Y. Shen, and A. J. Lew. Simulating curvilinear crack propagation in two dimensions with universal meshes. *International Journal for Numerical Methods in Engineering*, 102(3–4):632–670, 2015.

A. Schlüter, A. Willenbücher, C. Kuhn, and R. Müller. Phase field approximation of dynamic brittle fracture. *Computational Mechanics*, 54(5):1141–1161, 2014.

B. A. Schrefler, S. Secchi, and L. Simoni. On adaptive refinement techniques in multifield problems including cohesive fracture. *Computer Methods in Applied Mechanics and Engineering*, 195(46):444 – 461, 2006.

Y. Shen and A. Lew. An optimally convergent discontinuous Galerkin-based extended finite element method for fracture mechanics. *International Journal for Numerical Methods in Engineering*, 82(6):716–755, 2010a.

Y. Shen and A. Lew. Stability and convergence proofs for a discontinuous-Galerkin-based extended finite element method for fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 199(37–40):2360–2382, 2010b.

Y. Shen and A. J. Lew. A locking-free and optimally convergent discontinuous-Galerkin-based extended finite element method for cracked nearly incompressible solids. *Computer Methods in Applied Mechanics and Engineering*, 273:119–142, 2014.

E. Tamayo-Mas and A. Rodríguez-Ferran. A new continuous-discontinuous damage model: Cohesive cracks via an accurate energy-transfer process. *Theoretical and Applied Fracture Mechanics*, 69:90–101, 2014.

E. Tamayo-Mas and A. Rodríguez-Ferran. A medial-axis-based model for propagating cracks in a regularised bulk. *International Journal for Numerical Methods in Engineering*, 101(7):489–520, 2015.

H. Ulmer, M. Hofacker, and C. Miehe. Phase field modeling of fracture in plates and shells. *Proceedings in Applied Mathematics and Mechanics*, 12(1):171–172, 2012.

H. Ulmer, Ma. Hofacker, and C. Miehe. Phase field modeling of brittle and ductile fracture. *Proceedings of Applied Mathematics and Mechanics*, 13(1):533–536, 2013.

C. V. Verhoosel and R. de Borst. A phase-field model for cohesive fracture. *International Journal for Numerical Methods in Engineering*, 96(1):43–62, 2013. ISSN 1097-0207.

G. Z. Voyiadjis and N. Mozaffari. Nonlocal damage model using the phase field method: Theory and applications. *International Journal of Solids and Structures*, 50(20–21):3136–3151, 2013.

Y. Wang and H. Waisman. From diffuse damage to sharp cohesive cracks: A coupled XFEM framework for failure analysis of quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 299:57–89, 2016.

M. F. Wheeler, T. Wick, and W. Wollner. An augmented-Lagrangian method for the phase-field approach for pressurized fractures. *Computer Methods in Applied Mechanics and Engineering*, 271:69–85, 2014.

D. Xu, Z. Liu, X. Liu, Q. Zeng, and Z. Zhuang. Modeling of dynamic crack branching by enhanced extended finite element method. *Computational Mechanics*, 54(2): 489–502, 2014.

P. Zhao, L. Xie, C. Yang, and Y. Guo. Research on hydraulic fracturing based on smeared crack model. *Chinese Journal of Rock Mechanics and Engineering*, 34(S1): 2593–2600, 2015.

Appendix A

Convergence and performance study of the GPU-implemented algorithm

A.1 Convergence study

This appendix devotes to a verification of the GPU implementation of the phase field models. To this end, we present two examples with exact solutions to study the convergence rates in space and in time, respectively.

A.1.1 Spatial convergence

To study the spatial convergence, we consider an exact solution for $(\boldsymbol{x}, t) \in (0,1)^2 \times (0, t_f)$ where $t_f = 1.0 \times 10^{-5}$s:

$$\begin{aligned} u_x(\boldsymbol{x}, t) &= \left[x - (y-0.5)^2\right] t, \quad u_y(\boldsymbol{x}, t) = x\,(y-0.5)\,t, \\ d(\boldsymbol{x}, t) &= 0.5\left[x + (y-0.5)^2\right] t. \end{aligned} \tag{A.1}$$

We construct an initial boundary value problem whose exact solution is (A.1). To this end, we (a) let $\partial_D\Omega = \partial\Omega$ and impose appropriate initial conditions (1.5), boundary conditions (1.1) and (1.4d), and body force $\mathbf{b}$ in (1.4a) according to (A.1), and (b) modify (1.4b') in the case of model A to be

$$\dot{d} = \frac{1}{\eta}\left\langle 2(1-d)\psi(\boldsymbol{\varepsilon}) - \frac{g_c}{l}\left(d - l^2\Delta d\right) + Q\right\rangle_+, \tag{A.2}$$

with $Q = Q(\boldsymbol{x}, t)$ determined from (A.1).

Here we have chosen the solution in (A.1) such that it only linearly depends on t, and it can be proved that in this case all discretization errors come from the spatial discretization.

Figure A.1 depicts the convergence curves for $\boldsymbol{u}$ and d as functions of h computed at $t = t_f$ with $\Delta t = 10^{-8}$s, which is much smaller than the critical time step. The results from the two finest meshes give rise to a convergence rate of 1.99, and 1.39 for $\boldsymbol{u}$ and d, respectively. The reason why d has a lower convergence rate can be attributed to the term ψ in (A.2), which depends on $\boldsymbol{\varepsilon}$, resulting in a one-order reduction in convergence rate than that of $\boldsymbol{u}$.

Hence, with these convergence rates, the implementation is verified.

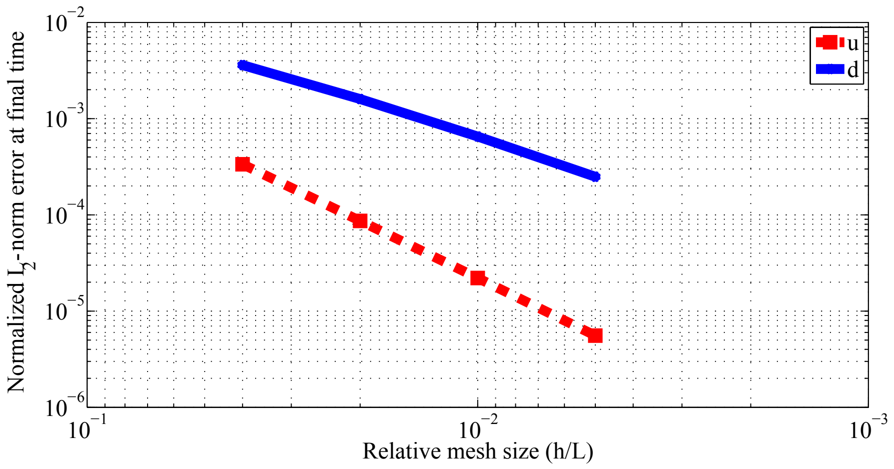


Figure A.1: Spatial convergence curves for $\boldsymbol{u}$ (red dashed line) and d (blue solid line). Here, for both fields, the L_2-norms of error normalized by the L_2-norm of the respective exact solutions are shown. From the two finest meshes, the convergence rates are estimated to be 1.99 and 1.39 for $\boldsymbol{u}$ and d, respectively. The reason why d has a lower convergence rate is explained in the text.

A.1.2 Time convergence

We follow the same idea to study the time convergence, using the following exact solution for $(\boldsymbol{x}, t) \in (0,1)^2 \times (0, t_f)$ where $t_f = 1.0 \times 10^{-4}$s:

$$u_x(\boldsymbol{x}, t) = u_y(\boldsymbol{x}, t) = d(\boldsymbol{x}, t) = t^5. \tag{A.3}$$

This exact solution is chosen to be constant in space, hence the only discretization error comes from time discretization. It is reminded that $Q(\boldsymbol{x}, t)$ as well as the initial and boundary conditions has to be determined from (A.3) accordingly.

Figure A.2 depicts the convergence curves for $\boldsymbol{u}$ and d. Here, we set $\Delta t = 1.0 \times 10^{-7}$s which is much smaller than the critical time step. The convergence rates obtained from the two smallest time steps are 2.00 for both fields.

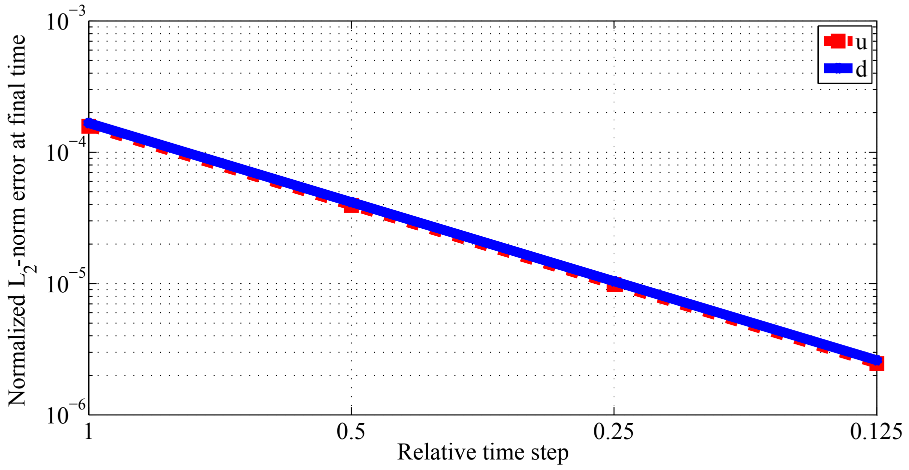


Figure A.2: Time convergence curves for $\boldsymbol{u}$ (red dashed line) and d (blue solid line). As in Figure A.1, the normalized L_2-norm error is shown. The error values for both fields are close to each other by coincidence. The convergence rates obtained from the two smallest time steps are 2.00 for both fields.

This example further verifies our implementation.

A.2 Performance study

In this section, we study the performance of the GPU implementation by measuring the wall time. Figure A.3 plots the wall time versus the physical time for the tension test with model C which has 105,545 DOFs (see Section 3.1 for more details about the example to be run). The simulation was run for 582,262 time steps. As can be seen, the slope increases near the end of the simulation, which indicates the effect of the decreasing critical time step (see Figure 3.5).

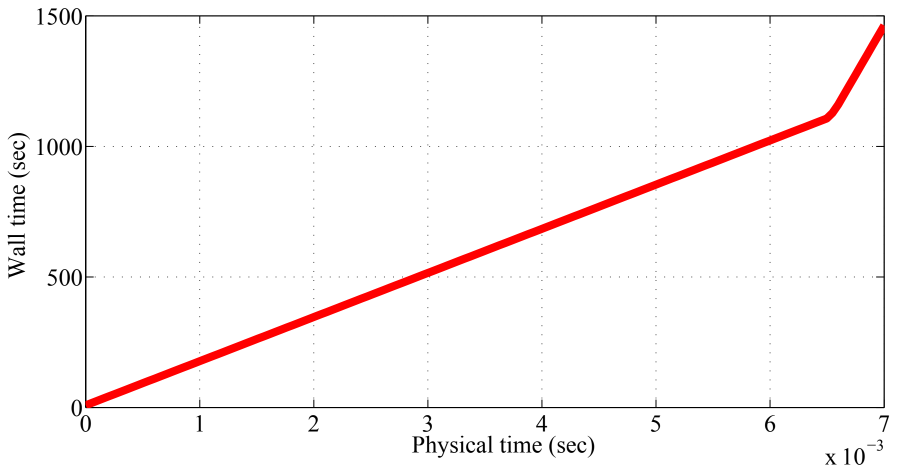


Figure A.3: Performance study. The wall time versus the physical time for the tension test with model C. The example has 105,545 DOFs, and 582,262 steps have been followed for the explicit solution. The change of slope in the graph is due to the decreasing critical time step.

Figure A.4 plots the wall time per physical time for the problem at hand against the number of DOFs. A linear scaling is observed, which may be the best possible scaling since all DOFs have to be updated. In other words, an implicit scheme is no better than the presented explicit one.

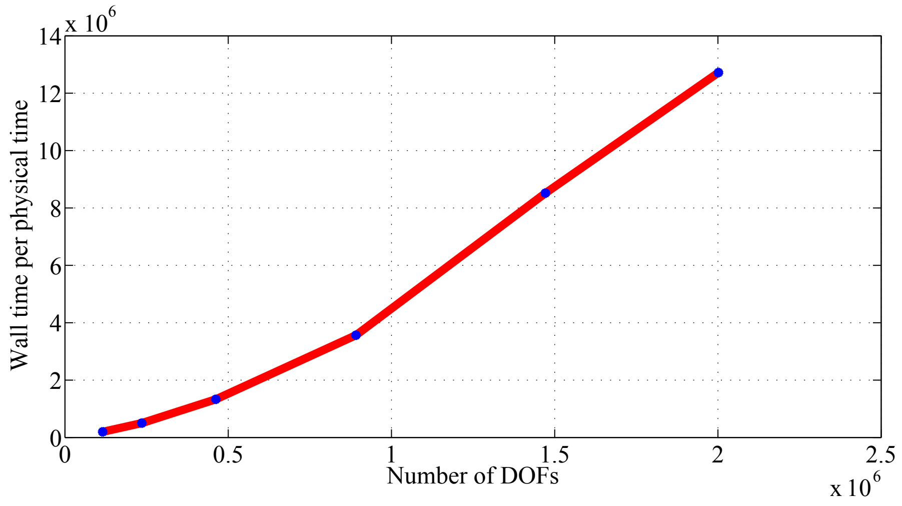


Figure A.4: Performance study. The wall time per physical time for the tension test with model C. The linearity of the curve shows the desired linear scaling.