SANDIA REPORT

SAND2016-9510
Unlimited Release
Printed June 2016

Cohesive phase-field fracture and a PDE constrained optimization approach to fracture inverse problems

Michael Tupek

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2016-9510
Unlimited Release
Printed June 2016

Cohesive phase-field fracture and a PDE constrained optimization approach to fracture inverse problems

Michael R. Tupek
Computational Solid Mechanics & Structural Dynamics
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-9999
mrtupek@sandia.gov

Abstract

In recent years there has been a proliferation of modeling techniques for *forward* predictions of crack propagation in brittle materials, including: phase-field/gradient damage models, peridynamics, cohesive-zone models, and G/XFEM enrichment techniques. However, progress on the corresponding *inverse* problems has been relatively lacking. Taking advantage of key features of existing modeling approaches, we propose a parabolic regularization of Barenblatt cohesive models which borrows extensively from previous phase-field and gradient damage formulations [1,2]. An efficient explicit time integration strategy for this type of nonlocal fracture model is then proposed and justified. In addition, we present a C++ computational framework for computing input parameter sensitivities efficiently for explicit dynamic problems using the adjoint method. This capability allows for solving *inverse* problems involving crack propagation to answer interesting engineering questions such as: 1) what is the optimal design topology and material placement for a heterogeneous structure to maximize fracture resistance, 2) what loads must have been applied to a structure for it to have failed in an observed way, 3) what are the existing cracks in a structure given various experimental observations, etc. In this work, we focus on the first of these engineering questions and demonstrate a capability to automatically and efficiently compute optimal designs intended to minimize crack propagation in structures.

Contents

Appendix

Figures

Acknowledgments

I acknowledge and appreciate the funding of this work from the Laboratory Directed Research and Development (LDRD) office and Engineering Sciences Research Foundation (ESRF). In addition, I thank John Dolbow, Tim Walsh, Christian Trott, Drew Kouri, Denis Ridzal, Kyle Starkey, Ivana Escobar, Brian Stevens, Steve Gomez, Scott Gampert, Ryan Coleman, the Sierra/DevOps team, and the Sierra/SolidMechanics team for providing critical ideas, technical assistance, and feedback.

1 Introduction

This report of comprised of two main sections. In section 2, existing phase-field fracture models are summarized and a numerical implementation based on an explicit time integration scheme is proposed. Initial results indicate that the approach is mesh convergent, efficient, and capable of capturing experimentally observed brittle fracture phenomena in both 2D and 3D. In section 3, an approach to inverse methods involving highly nonlinear dynamic fracture simulations is proposed. The adjoint *embedded sensitivity* equations are derived for a general explicit dynamic simulation, and specialized to the phase-field evolution equations of primary interest here. Using Sandia's Rapid Optimization Library (ROL), a prototype optimal design problem is demonstrated. Appendix A contains a summary of ***Springbok***, a new C++ library for computing adjoint sensitivities for dynamic problems. This library was critical for managing the complexity of the adjoint calculations required for the nonlinear explicit dynamic inverse problem demonstrated here.

2 Phase-field models for dynamic fracture

Classical *local* damage models are known to be ill-posed [3]. In practice, this results in non-convergence for many common numerical approaches to failure modeling. To demonstrate this issue, mode-I fracture simulations using a maximum principal stress failure criterion and element deletion to model the failure of brittle materials are shown in Figure 1. As expected, the predicted crack path is highly sensitive to the initial mesh size. Perhaps more importantly, the energy dissipated in each of these simulations varies substantially. It is well established that for fracture in brittle materials, the energy release rate associated with the creation of new surfaces is the critical factor for determining the propagation and propagation direction of cracks [4]. Brittle failure models that are successful at overcoming the limitations of local damage models are energetic and often nonlocal. Examples include linear elastic fracture mechanics (LEFM), cohesive zone models, peridynamics [5–7], and phase-field/gradient damage models [1, 2, 8–10]. Compared to the first two approaches, peridynamics and phase-field models have one critical advantage: they are energetic *and* nonlocal. This allows the prediction of crack branching and coalescence to be emergent, not prescriptive. In other words, crack branching occurs naturally and automatically in these theories, without having to impose any additional branching criterion. This is in contrast with, e.g., cohesive models, which are energetic and contain a length scale, but are essentially local and require external criterion to predict crack branching.[1] For this work, we pursue phase-field over peridynamics because the former is a more direct extension of classical damage modeling approaches, and it fits naturally in legacy finite element codes. The key contributions here are: 1) to highlight the advantage of cohesive phase-field models [2,12] over more standard phase-field models [1, 8–10], and 2) that phase-field models can be efficiently and accurately integrated in time using an explicit update rule.

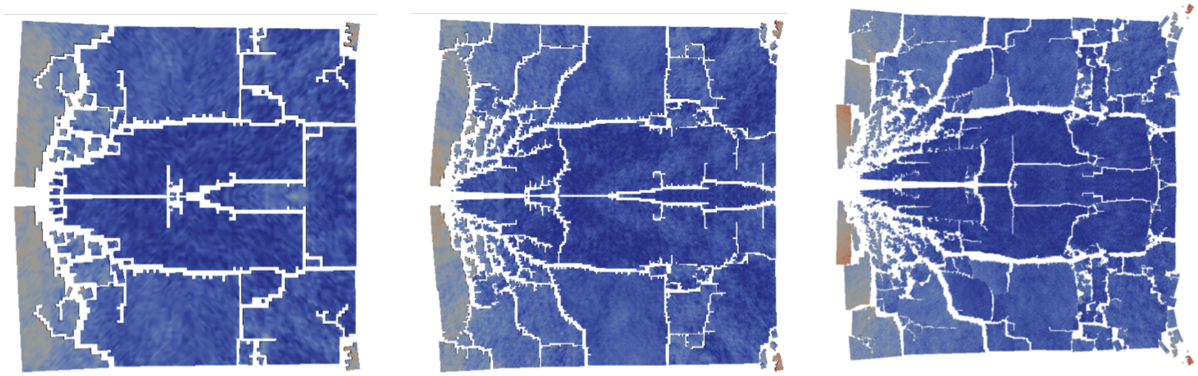


Figure 1: Non-convergence of failure models using a local criterion. The mesh resolution is increasing from left to right.

The use and development of phase-field models for brittle fracture has gained significant attention in recent years. This approach avoids the need for discretizing sharp crack

[1] For inter-element cohesive model discretizations [11], branching events are often determined primarily by the initial mesh!

discontinuities by smoothing the representation of a crack over a prescribed length scale, L. A schematic of the 'smoothed' phase-field regularization of a sharp crack is depicted in figure 2. The approach was initially justified as a mathematical regularization of variational fracture [4, 8] and approximates Griffith fracture/LEFM in the appropriate limits.

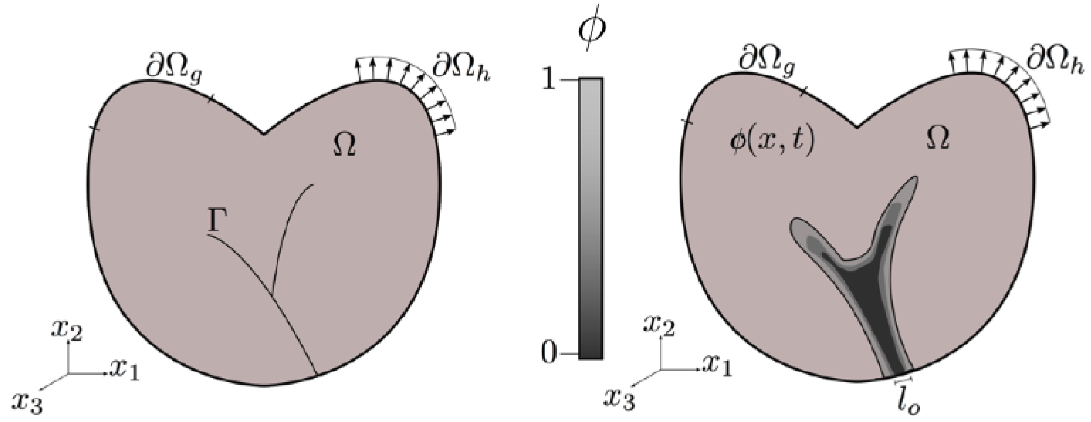


Figure 2: Schematic showing the phase-field approximation to a sharp crack.

Starting from an energetic point of view, phase-field models postulate a Helmholtz free-energy functional ψ which depends on the strain $\boldsymbol{\varepsilon}$ and the phase-field (a.k.a. damage field) $\phi \in [0,1]$:

$$\psi(\boldsymbol{\varepsilon},\phi) = g(\phi)\psi_e^+ + \psi_e^- + h(\phi) + \frac{c}{2}\nabla\phi\cdot\nabla\phi, \tag{1}$$

with strain energy $\psi_e = \frac{1}{2}\ \boldsymbol{\varepsilon} : \mathbb{E} : \boldsymbol{\varepsilon}$, strain $\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right)$, and displacement field $\mathbf{u}$. The $+$ and $-$ on the strain energy terms distinguishes between tensile and compressive contributions and is intended to prevent damage under pure compression and to maintain material resistance under compression [1]. The modeling choice of how to decompose the strain energy into compressive and tensile parts turns out to be critical for crack propagation predictions [13], but will not be discussed further in this report. The standard damage convention is used with $\phi = 0$ indicating no damage and $\phi = 1$ indicating a fully damaged state. The last two terms in equation (1) together form the so-called crack energy density $\gamma = h(\phi) + \frac{c}{2}\nabla\phi\cdot\nabla\phi$. The integral of this term over the domain forms an approximation to the energy associated with the creation of a new crack surface:

$$\int_\Omega \gamma\, d\mathbf{x} \approx G_c A_{crack}, \tag{2}$$

where A_{crack} is the crack surface area. Crack energy densities are typically parameterized by a length L, and the approximation (2) becomes exact in the limit $L \to 0$.

The momentum balance equation follows from equation (1) as

$$\rho\ddot{\mathbf{u}} = \nabla\cdot\boldsymbol{\sigma} + \mathbf{b}, \tag{3}$$

where $\boldsymbol{\sigma} = g(\phi)\frac{\partial\psi_e^+}{\partial\boldsymbol{\varepsilon}} + \frac{\partial\psi_e^-}{\partial\boldsymbol{\varepsilon}}$.

Assuming a strictly dissipative process, we take the phase evolution equation to be:

$$\eta\dot{\phi} = \langle -D_\phi \psi \rangle_+ = \langle -g'(\phi)\hat{\psi}_e^+ - h'(\phi) + c\nabla^2\phi \rangle_+, \tag{4}$$

where $\eta \geq 0$ is the phase viscosity and D_ϕ denotes a Frechét derivative with respect to the phase-field. Irreversibility of the phase evolution, $\dot{\phi} \geq 0$, is ensured by the inclusion of the Macauley brackets:

$$\langle x \rangle_+ := \begin{cases} x & \text{for } x > 0 \\ 0 & \text{for } x \leq 0. \end{cases}$$

2.1 Standard phase-field fracture model

For completeness, we describe the standard phase-field fracture model. (e.g., [1,9])[2] For this model the degradation function is

$$g(\phi) = (1-\phi)^2,$$

and the phase potential is

$$h(\phi) = \kappa\phi^2,$$

where κ is related to the length scale and energy release rate. Writing it out in full, the classic phase-field model has a potential given by:

$$\psi(\boldsymbol{\varepsilon}, \phi) = (1-\phi)^2\psi_e^+ + \psi_e^- + \frac{G_c}{2L}\phi^2 + \frac{G_c L}{2}\nabla\phi\cdot\nabla\phi.$$

This model is known to Γ-converge to Griffith fracture/LEFM as the length scale is decreased $L \to 0$. However, for any finite length scale $l > 0$, damage begins to evolve for any non-zero straining of the material. While this means there is no stress threshold for damage evolution, there is still a known maximum obtainable stress predicted by the model in 1D given by (see [9])

$$\sigma_{\max} = \frac{9}{16}\sqrt{\frac{EG_c}{6L}}.$$

This is often interpreted as the cohesive strength for a phase-field model with non-zero L.

2.2 Cohesive phase-field fracture model

The cohesive gradient damage model proposed in [2] is a special case of a phase-field fracture model where

$$g(\phi) = \frac{(1-\phi)^2}{1 + (m-2)\phi + (1+pm)\phi^2}$$

[2] At first glance the models from [1] and [9] appear different; however, a change of variables and a redefinition of L yields equivalent models.

and[3]

$$h(\phi) = k\phi,$$

where the parameters k, c and m are related to macroscopic cohesive fracture properties [2]:

$$k = \frac{3}{4}\frac{G_c}{L} \qquad c = \frac{3}{8}LG_c \qquad m = \frac{3}{2}\frac{EG_c}{\sigma_c^2 L}. \tag{5}$$

The parameter p influences the shape of the cohesive traction separation law following initial damage (and also the process zone size [12]), G_c is the fracture energy/critical energy release rate, L is the phase regularization length scale and σ_c is the critical stress in 1D. Lorentz et al. argue on physical grounds that $p \geq 1$ and $m \geq p + 2$. The first constraint ensures the traction separation law is a monotonically decreasing function of the opening displacement. The second constraint can be rewritten

$$L < \frac{3}{2(p+2)}\frac{EG_c}{\sigma_c^2}, \tag{6}$$

where we identify the right hand size as being the correct scaling for the process zone length in cohesive zone models. We can interpret this condition as constraining the regularization length scale to be small enough to resolve the process zone size.

Note that we have replaced ψ_e^+ in equation (1) with $\hat{\psi}_e^+$ to initialize the driving strain energy density to the threshold value implied by the Lorentz model. For an initially undamaged material, $\phi = 0$, $g'(\phi) = -m$ and $\nabla^2\phi = 0$, so

$$\eta\dot{\phi} = m\hat{\psi}_e^+ - k.$$

Requiring $\dot{\phi} \geq 0$, we redefine $\hat{\psi}_e^+$ as

$$\hat{\psi}_e^+(t) = \max\left(\psi_e^+(t), \frac{k}{m}\right).$$

In other words, we initialize $\hat{\psi}_e^+ = \frac{k}{m}$. Damage can only evolve once this initial strain energy barrier has been exceeded. This is in strong contrast to the standard phase-field models used in [1, 8–10] in which some damage evolution occurs for any strain energy $\psi_e^+ > 0$.

2.3 Crack surface energies with a critical stress threshold

To gain a little more insight into the cohesive phase-field model, we analyze the crack energy density term γ independently of the mechanics terms. Similar to the derivation of the standard phase-field model by Miehe, we start with the variational problem, approximating the crack surface energy in 1D by:

$$\mathcal{E}_{\text{crack}}(\phi) = \int_{-\infty}^{\infty} \gamma(\phi)\, dx, \tag{7}$$

[3] Lorentz, et al. formulate this term as a dissipation potential to include the effects of irreversibility, here we include this term as part of the stored energy associated with the phase to simplify the exposition.

where the crack energy density $\gamma(\phi)$ for this model is given by:

$$\gamma(\phi) = k\phi + \frac{c}{2}\nabla\phi \cdot \nabla\phi.$$

The functional (7) yields the Euler-Lagrange equation:

$$k = c\nabla^2\phi. \tag{8}$$

Using the constraint that the bar is fully broken at $x = 0$ and undamaged far away: $\phi(0) = 1$ and $\phi(\pm\infty) = 0$, a solution is

$$\phi(x) = \begin{cases} \frac{1}{L^2}(L - |x|)^2 & \text{for } |x| \leq L \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $L = \sqrt{\frac{2c}{k}}$.

To ensure that the functional (7) approximates the energy release rate associated with surface creation, we require

$$\int_{-\infty}^{\infty} \gamma(\phi)\, dx = \int_{-\infty}^{\infty} k\phi + \frac{c}{2}\nabla\phi \cdot \nabla\phi\, dx = G_c.$$

With this additional constraint, we can reproduce the parameters from equation (5): $c = \frac{3}{8}LG_c$, and $k = \frac{3}{4}\frac{G_c}{L}$.

This type of phase-field crack energy density results in a natural damage threshold, as implied by the $\phi(x) = 0$ for $|x| > L$ part of the analytic solution in (9). This feature is independent of the chosen degradation function $g(\phi)$ and has been used is conjunction with the standard phase-field degradation function [1, 9]. However, in order for the phase-field model to be truly cohesive, we require *both* a threshold crack energy density *and* a degradation function which ensures that the critical stress is independent of the regularization length scale. This property will be shown for the Lorentz model below.

2.4 Requirements for a cohesive phase-field model

A critical difference between the Lorentz gradient damage model and standard phase-field models is that the Lorentz model converges to a cohesive zone model as the regularization length scale goes to zero, $L \to 0$ [2], while the standard phase-field model Γ-converges to a Griffith model as $L \to 0$ [4, 14].[4] A proof of this is provided in [2]. Here we give a simplified

[4] Both Griffith and cohesive (a.k.a. Barenblatt) models of fracture are characterized by a release of energy, G_c, per unit area of crack growth. The distinction is that the Griffith model is purely energetic and effectively has an infinite critical stress, with a corresponding singular stress intensity factor at the crack tip. Cohesive models have a finite critical stress and correspondingly have finite stress intensities at the crack tip.

and hopefully more intuitive demonstration that the model is cohesive. Recalling that the standard phase-field model has a failure stress in 1D given by

$$\sigma_{\max} = \frac{9}{16}\sqrt{\frac{EG_c}{6L}},$$

and that the fracture process zone length, l_{fpz} (depicted schematically in figure 3), scales as

$$l_{fpz} \propto \frac{EG_c}{\sigma_c^2}$$

we find

$$L \propto l_{fpz}.$$

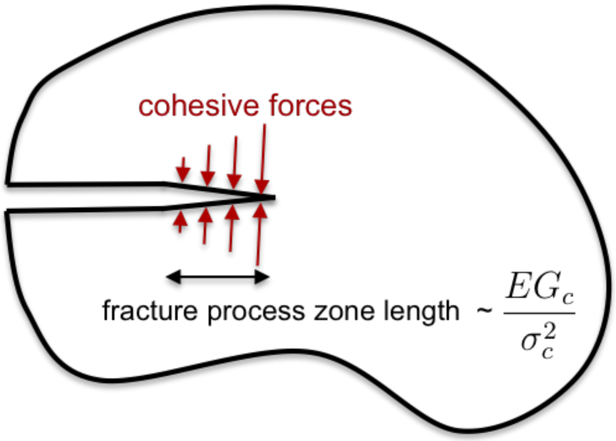


Figure 3: Schematic for a cohesive fracture process zone. Cohesive models are characterized by a process zone over which the surface tractions decay from the cohesive failure stress σ_c to 0 in the wake of the propagating crack. We refer to the distance along the propagating crack over which this decay occurs as the fracture process zone length.

In other words if we fix a critical stress to approximate a cohesive-like failure criterion, the standard phase-field model predicts that the regularization length scale, L, must scale proportionally to the fracture process zone length scale, l_{fpz}. In other words, these two length scales are directly tied together in the standard phase-field model. A direct consequence of this is shown on the left of figure 4. If the fracture process zone length is a significant fraction of the geometrical length scales in the problem,[5] the regularization provided by the phase-field model appears unphysically large. For a truly cohesive phase-field model, the regularization length scale should be able to be chosen independently of the fracture process zone length scale. From equation (6), we see that the Lorentz model satisfies this stricter definition of a cohesive phase-field model. Furthermore, in [2], it is demonstrated that this model predicts an exponential-like traction separation decay function, where shape of the

[5] Recall that the fracture process zone length is predominantly a function of material properties such as fracture toughness, stiffness, and critical stress.

decay can be controlled by the parameter p. Changing the shape of the traction-separation law also has the effect of modifying the process zone size [12].

The independence of the regularization length-scale from the fracture process zone length is shown in figure 4 for the case of a dynamic mode-I dynamic crack propagation problem, where the standard and Lorentz phase-field models are compared. The loading in both cases is a prescribed vertical velocity on the left side of the specimen. The top half has an applied velocity of 9 m/s in the positive y direction, while the bottom half has applied the same velocity in the negative direction. The maximum stress is given as $\sigma_c = 250e6$, the critical energy release rate is $G_c = 100e3$, the young's modulus is $E = 200e9$, the Poisson's ratio is $\nu = 0.3$, and the density is $\rho = 8000$. The presumed unit system is MKS. The problem is 2D, with the specimen having a height and width of 0.1. The cohesive shape parameter is taken to be $p = 3$. Not shown in the figures is the initial crack which extends from the left edge to the center of the specimen, located at the $y = 0$ plane. The damage profiles shown are propagating from this initial pre-crack.

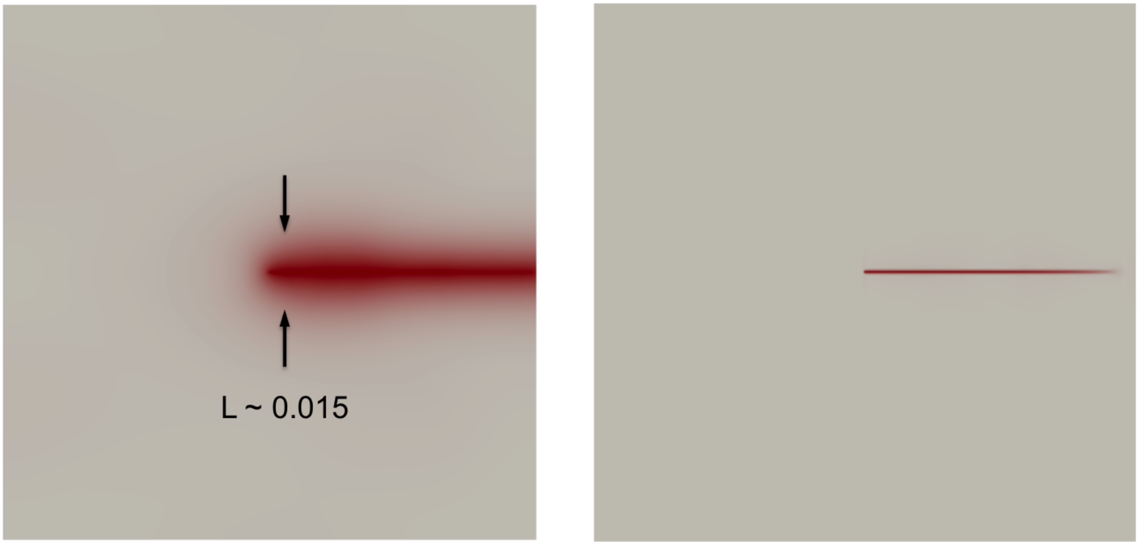


Figure 4: Griffith vs. cohesive phase-field model results. Simulation result for mode-I crack propagation problem with the standard phase-field model (left) and Lorentz's cohesive gradient damage model (right).

For a little more insight, we can determine the critical strain energy density predicted by the Lorentz model using equation (4). In an undamaged material, the model predicts some damage evolution when the following condition is satisfied:

$$-g'(0)\hat{\psi}^{+}_{crit} = k.$$

Due to dimensional scaling, the parameter k must be inversely proportional to the regularization length, L, for any physical phase-field fracture model. If $-g'(0)$ is independent of L (as with the standard degradation function, where $g(\phi) = (1-\phi)^2$), the critical strain-energy-density at failure must depend on L. However, the Lorentz model is carefully design in such a way that

$$\hat{\phi}^{+}_{crit} = \frac{k}{-g'(0)} = \frac{k}{m},$$

where $\frac{k}{m}$ is independent of L.

We see that in this model, the critical strain energy density (or critical stress) is completely independent of the regularization length scale, in contrast to the standard model. Figure 4 demonstrates the advantage of this flexibility. On the left, the standard phase-field solution has damage smoothed over a length similar to the process zone length, which is visibly large in this problem. On the right, we can choose to interpret the regularization length scale as a numerical parameter and have it be an order on magnitude smaller. This results in a significantly sharper effective crack.

A final property of the Lorentz model is that stress is a strictly decreasing function of the damage, so that the critical stress is also the maximum obtainable stress. Combine these properties, we find that the maximum stress in the Lorentz model is independent of the regularization length. These feature lead to the cohesive properties of the Lorentz model.

2.5 Parabolic regularization, explicit time integration, and stable time step

The model resulting from equations (3), (4) can be considered a parabolic regularization of brittle cohesive fracture. It is common for phase-field models to be considered elliptic regularization due to the inclusion of the diffusion/elliptic term $\nabla^2\phi$. With the addition of the phase viscosity term $\eta\dot{\phi}$, equation (4) become parabolic in nature.

One practical difficulty with equations (3) and (4) is that the first is a hyperbolic partial differential equation (PDE), while the second is a parabolic PDE. It is common in practice to numerically integrate the hyperbolic equation (3) in time using an explicit time-stepping algorithm [15, 16], especially for problems involving wave propagation, damage and failure. However, parabolic equations such as (3) (similar to the heat equation) are notorious for being difficult to integrate explicitly due to the restrictive time-steps required as the discretization is refined. In particular, the stable explicit time step for this class of PDEs scales as $\Delta t \propto (\Delta x)^2$. An alternative is to integrate (3) using an explicit scheme and (4) implicitly. However, this incurs a *significant* computation cost compared to just integrating (3) in the standard manner. Here we propose to break with convention and integrate the parabolic equation (4) explicitly as well, in spite of the time step restriction, which we believe is not the limiting factor for most phase-field fracture problems in practice.

A rough physical justification for how we can get away with this is as follows: in elastodynamics it is well know that information (i.e. waves) can travel no faster than the wave speeds in the material. This is true for crack propagation as well; the speed of propagation is bounded by a fraction of the wave-speeds of the surrounding material. Parabolic systems effectively propagate information instantaneously, which mean that equation (4) represents something of a contradiction because it allow for information about material damage to propagate faster than the material wave-speeds.[6]

Stable time step restrictions can be thought of as a constraint on how fast information is allowed to travel through a mesh. The implication of this is that the stable time step required to integrate the hyperbolic equation (3) should already be sufficiently small to integrate the parabolic equation (4) provided that the phase viscosity η is appropriately chosen, e.g., in physically reality, we don't expect to see damage propagating faster than the elastic wave-speeds which are traditionally responsible for restricting the explicit time step. A simplified analysis leads to the scaling:

$$\frac{\eta}{\Delta t} \gtrsim \frac{c}{(\Delta x)^2}.$$

Taking s to be the fastest wavespeed in the material, the stable explicit time step for equation (3) is already restricted by $\Delta t \leq \frac{\Delta x}{s}$ due to the mechanics, so if we choose $\eta^\star$ such that

$$\eta^\star \gtrsim \frac{c}{s\Delta x} \geq \frac{c\Delta t}{(\Delta x)^2},$$

the critical time step will not be impacted by the inclusion of the phase-field equation. Ideally, we'd like $\eta^\star$ as small as possible to minimize energy dissipation due to phase viscosity and to maximize the energy going into crack formation, so in practice we choose the phase viscosity to be

$$\eta^\star = \frac{fc}{s\Delta x} = \frac{f\frac{3}{8}LG_c}{s\Delta x} = f^\star N\frac{G_c}{s},$$

where $f^\star \leq 1$ is a safety factor, and $N = \frac{L}{\Delta x}$ is the approximate number of elements that are being used to resolve the length scale L. What this implies is that if we refine the mesh keeping N fixed, the stable time step for explicit time integrated phase-field fracture scales with the stable time step for elasto-dynamics.

Implementation details

The phase-field time integration is done using an explicit Euler time integration strategy:

$$\mathbf{M}_\phi \left(\boldsymbol{\phi}_h^{n+1} - \boldsymbol{\phi}_h^n\right) = \mathbf{f}_\phi^n \Delta t,$$

which steps at the same rate as the explicit Newmark time integrator for the mechanical system. The nonlinear phase-field force $\mathbf{f}_\phi^n = \mathbf{f}_\phi(\boldsymbol{\phi}_h^n, \mathbf{u}_h^n)$ is evaluated at the old time step and involves a consistent mass matrix for the $-g'(\phi)\hat{\psi}_e^+ - k$ term and a consistent stiffness matrix for the $c\nabla^2\phi$ term (computed explicitly without actually forming the matrices). The mass matrix $\mathbf{M}_\phi$ is the lumped mass matrix corresponding to a density of 1.

[6] While this seems unphysical, we have to realize that equation (1) is a *regularization* of the "true" physical problem which is the limit $L \to 0$.

Sierra input

The input syntax for using the explicit cohesive phase-field model in Sierra/SolidMechanics [17] is

```
begin property specification for material phase_field_fracture
  density          = 2450
  begin parameters for model gradient_damage_explicit
    Youngs modulus = 32e9
    Poissons ratio = 0.2
    fracture length scale = 2.5e-04
    fracture energy = 3.0
    critical stress = 1.0e7
    cohesive shape  = 2.0
    phase viscosity = 0.8
  end
end
```

It is also necessary to add a reaction diffusion command block:

```
begin reaction diffusion
  initial value = 0.0
  solve explicit = on
end reaction diffusion
```

The initial value $= 0.0$ sets the initial phase-field to 0, corresponding to no damage (this is in contrast to some phase-field fracture models which start the undamaged phase at 1 and damage it to 0). Most of the material properties are standard. The two notable exceptions are the "cohesive shape" which corresponds to p from [2,12] (we typically use 1–3) and "phase viscosity" which is currently poorly labeled as it actual corresponds to the numerical safety factor $f^\star$ and should be set to a value under 1.0. We also try to have at least 5-10 elements spanning the fracture length scale L.

2.6 2D examples

Here we present several examples in 2D. These examples demonstrate 1) convergence rates of the method, 2) the fracture mechanisms which are predicted by the model, and 3) the computational efficiency of the method.

Dynamic mode-I fracture

A common initial benchmark problem for brittle fracture is dynamic mode-I crack propagation. Considering figure 5, the problem setup has a prescribed velocity on the left side of the specimen. The upper side of the left edge has a positive velocity of 25 m/s applied to it, while the bottom part of the left side has a negative 25 m/s prescribed velocity. The mechanics parameters used are a stiffness $E = 190e9$, density $\rho = 8000$, energy release rate $G_c = 2e5$, Poisson's ration $\nu = 0.3$, cohesive shape $p = 3$, and a cohesive strength of $\sigma_c = 40e8$. The specimen is 0.1 in length and width. The simulation is 2D and similar to the simulation from section 2.4, the is a pre-crack starting at the left edge and ending in the middle of the specimen.

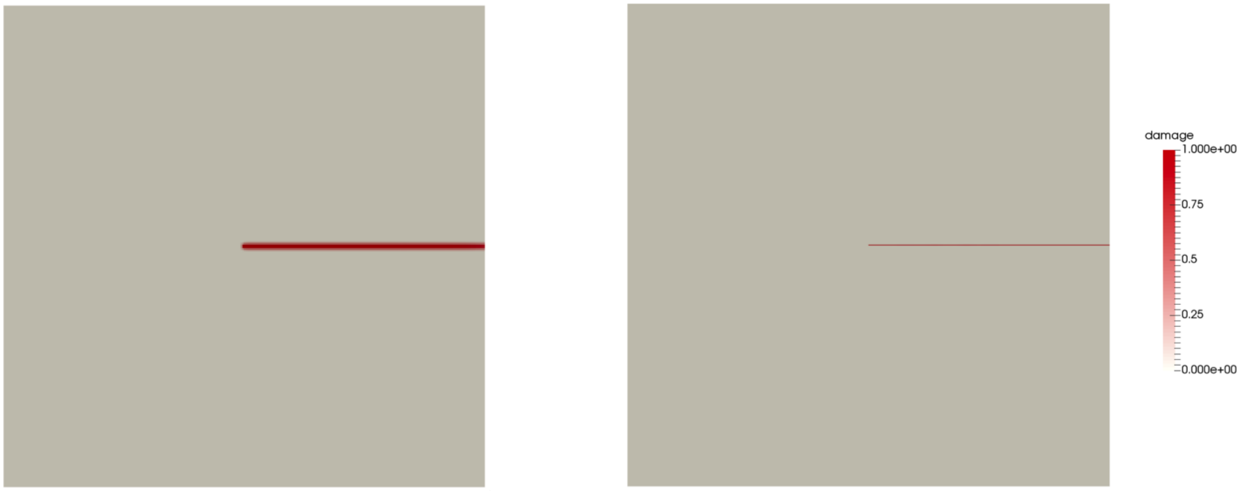


Figure 5: Mode-I crack propagation predictions for different regularization lengths.

For phase-field simulations, the notion of convergence can be a little subtle. Here we consider two kinds of convergences: one in which we fix the regularization length scale as the mesh is refined, and the other where we decrease the regularization length with the mesh size. We observe convergence in both cases. For the first case, we get the anticipated convergence rate of ~ 1. We are solving a parabolic PDE using linear finite elements, so we expect the energy norm to converge spatially at first order. In the second case the regularization length-scale is changing, so we are actually solving a different PDE for each refinement. Despite this complication, we find that we do indeed appear to be converging (albeit slowly, at half order) to a solution which dissipates the correct energy, and has the correct failure stress. In this sense it appears we are converging to a cohesive model solution. We note that this is only the case for a single propagating crack as cohesive models do not, in themselves, predict branching. As shown below, phase-field models are naturally capable of predicting branching without any additional constitutive assumptions. When this occurs, there is no corresponding cohesive model capable of capturing that behavior without imposing an external branch criteria, or allowing a discretization to artificially choose to branch based on initial mesh geometry.

Example mode-I crack propagation predictions for two different regularization length scales are shown in figure 5. We find that the dissipated energy in the two cases is similar. In fact, energy conservation using with this approach is nearly exact (provided that the time step is sufficiently small), as demonstrated in figure 6. Furthermore, the predicted energy dissipation due to crack propagation matches well with the expected analytic result which is

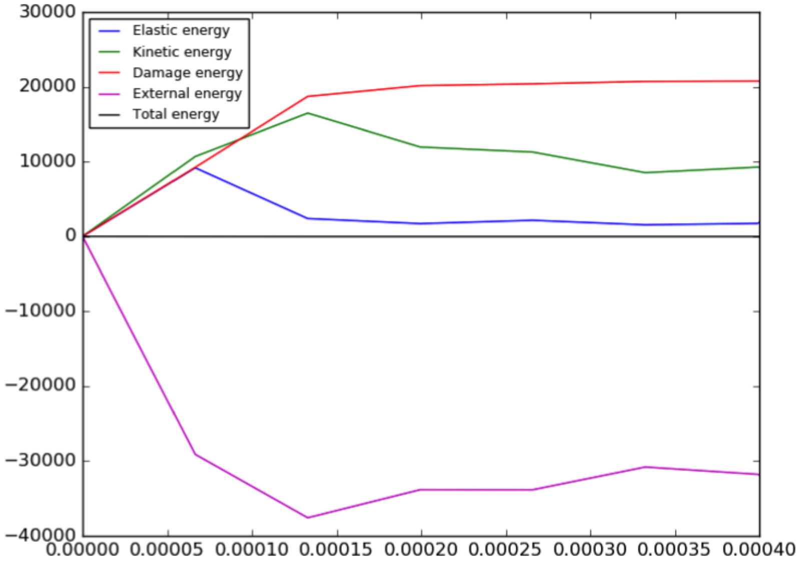


Figure 6: Energy vs. time for explicit phase-field fracture simulation subjected to dynamic mode-I loading.

$G_c A_{crack}$, where A_{crack} is the final crack surface area. A convergence plot for the two types of convergence mentioned above is shown in figure 7. The error in this plots in the difference between the energy dissipated numerically and the analytic energy dissipated, $G_c A_{crack}$.

Dynamic mode-I fracture: transition from a single crack to crack branching

An example demonstrating a transition from a single propagating crack to crack branching is shown in figure 8. For a given applied mode-I velocity, a bifurcation in the fracture pattern occurs as we lower the fracture energy. The way this is typically explained is that for a given loading, if the critical fracture energy G_c is high, a single crack is sufficient to dissipate the input energy. If instead G_c is low, then more than one crack is necessary to dissipate that same amount of input energy.

Dynamic mode-I fracture: mesh insensitivity of branch predictions

Qualitative convergence of dynamic branching under a nominally mode-I loading is shown in figure 9. This demonstrates that even in the presence of crack branching, phase-field fracture predictions appear to be very insensitive to the mesh refinement. While the results here are computed on a structured grid, a similar result is expected even for an unstructured initial mesh.

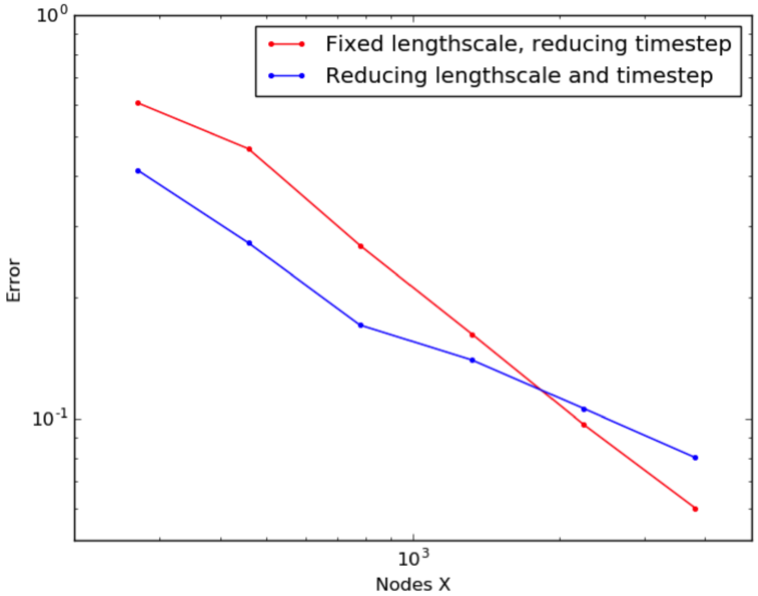


Figure 7: Convergence plots for the two different convergence strategies. The first way (in red) fixes the length scale and reduces the mesh resolution and the time step. The second way (in blue), has the time step and length scale reducing with the mesh size (though both at only half the rate). Converges rates of 1 and 0.5 are observed for these two cases.

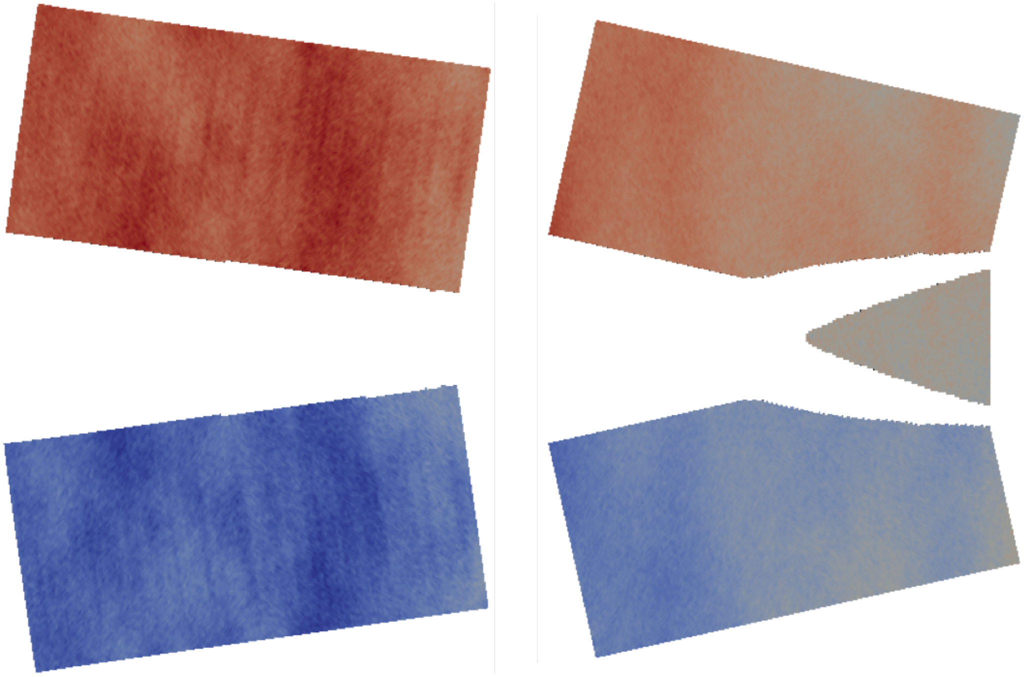


Figure 8: Transition from straight crack propagation to crack branching as the critical energy release rate is decreased.

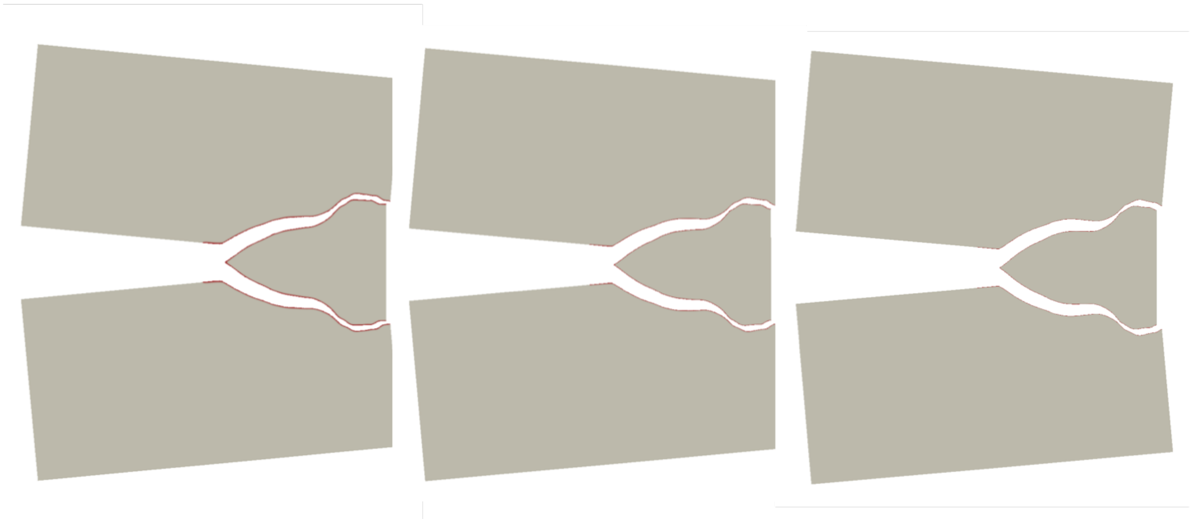


Figure 9: Dynamic mode-I crack propagation simulation for varying refinement levels. The number of elements per mesh from left to right are: 1.4 million, 4.1 million, 11.8 million. This result demonstrates that the predicted crack path is not very sensitive to the refinement level. Also note that due to a GPU implementation, the total run time for the finest simulation with 11.8 million elements was very reasonable. It only took around 5 hours on a single Nvidia Tesla K-40 GPU.

Kalthoff

The Kalthoff test, which is based on experimental results by Kalthoff and coworkers [18,19], has emerged as a benchmark problem for numerical approaches to brittle and ductile failure modeling [20,21]. The test consists of a plate with two edge notches, as depicted in figure 10, impacted by a projectile with initial velocity v_0. The two initial notches are equidistant from the center-line of the target and are separated by distance equal to the diameter of the projectile. For the simulations here, the notches are separated by a distance of 0.05 m, while the height of the target is 0.2 m and its width is 0.1 m. The material parameters used are a density $\rho = 8000\ kg/m^3$, Young's modulus $E = 190$ GPa, Poisson's ratio $\nu = 0.3$ and fracture energy $G_c = 22,000\ J/m^2$. We use an initial impact velocity of $v_0 = 16\ m/s$, which in experiments resulted in brittle crack propagation at an angle of around 70° relative to horizontal. A result using the explicit cohesive phase-field formulation is shown in figure 11. The predicted crack propagation angle is comparable to the experimentally observed angle, a result which is very competitive with alternative techniques [7,20,21].

Glass/steel interface problem

Recent experimental crack propagation results for a glass/steel interface problem [22] are shown in figure 12. The problem setup has a steel layer on top and a larger section of glass below. The temperature is dropped by around 470°. As the thermal contraction in the metal is larger than the contraction in the glass, this results in an effective moment being applied

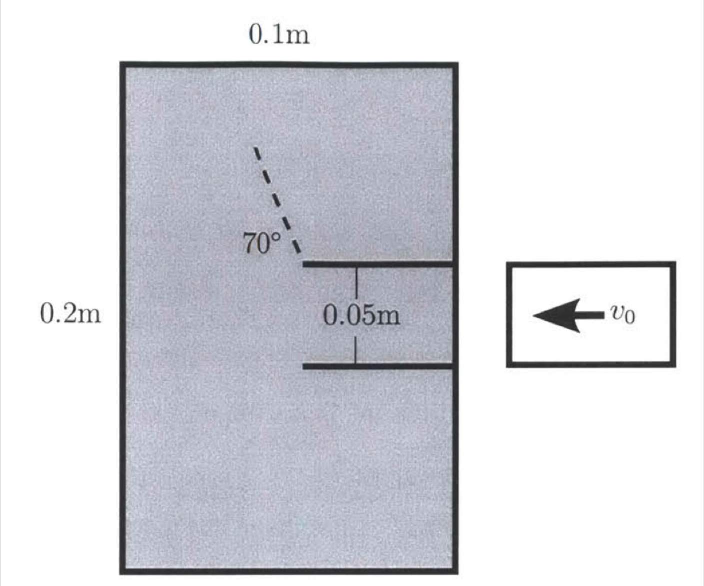


Figure 10: Schematic of the Kalthoff [19] impact problem setup.

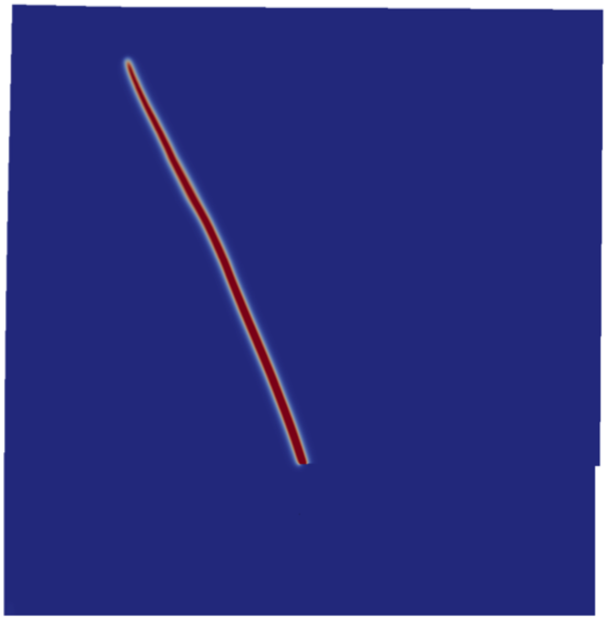


Figure 11: Predicted damage for Kalthoff simulation. Crack path matches well with the experimentally observed crack propagation angle of ∼ 70°. Only the top half of the specimen is modeled.

to the specimen. This in turn drives an initial vertical crack to begin propagating farther in the vertical direction. As the crack approaches the glass/metal interface (which is cannot easily cross), it turns to eventually run parallel to that interface. An example simulation result using phase-field is shown in figure 13.

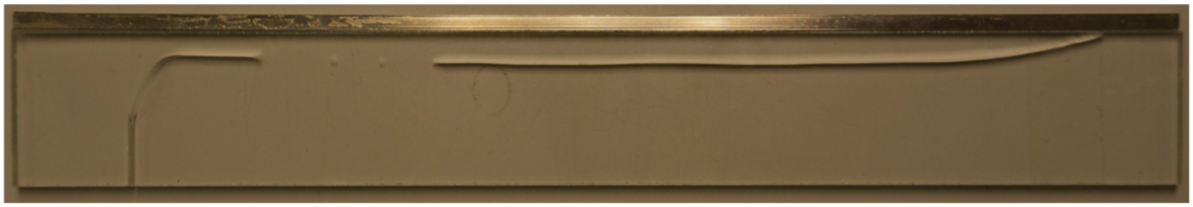


Figure 12: Experimentally observed crack propagation for glass/steel interface experiment [22].

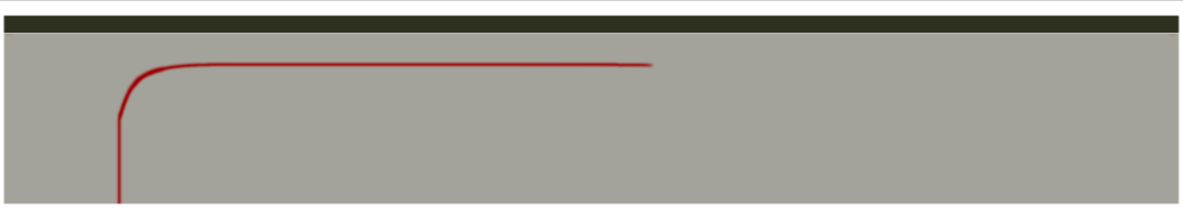


Figure 13: Simulated crack propagation for glass/steel interface problem using explicit cohesive phase-field approach.

The glass stiffness was taken as $E = 64e9$, the Poisson's ratio was $\nu = 0.2$, the energy release rate was $G_c = 400$, the critical stress was $\sigma_c = 8.0e7$, and the thermal expansion coefficient was $\alpha = 3.25e^{-}6$. The metal stiffness was taken as $E = 193e9$, the Poisson's ratio was $\nu = 0.29$, and the thermal expansion coefficient was $\alpha = 17.3e^{-}6$.

The experimentally observed results are qualitatively replicated. This simulation was actually run in explicit dynamics, but over a long enough time to replicate quasi-static loading. Approximately 1.5 million element were used for this result, but no expensive (and serializing) mesh modifications were necessary. Running on just a single Nvidia Tesla K-40 GPU, this result took less than 6 hours to compute. It is expected that a truly quasi-static implementation (e.g., using dynamic relaxation) could improve the efficiency of the approach significantly.

2.7 3D examples

The previous 2D example problems were all simulated using a newly developed, GPU-ready, adjoint-capable, structured grid finite element code, ***PhaseFieldMini***. In addition to developing the explicit phase-field capability in this miniapp, the model was also added to Sierra/SolidMechanics [17], leveraging the initial implicit phase-field implementation by Prof. Dolbow and the SolidMechanics team. This allows us to do large 3D brittle fracture simulations in parallel on unstructured meshes. Two example problem results will be briefly shown: 1) ceramic impact and 2) torsional fracture.

Ceramic impact

A demonstration ceramic impact simulation result is shown in figure 14. Without going into the details (as it is still work in progress), we will only comment that the approach appears to generalize well to 3D. This particular simulation is capable of capturing several key features of ceramic impact problems: spall planes, radial crack formation, and conical crack which form internal to the target at approximately 45°.

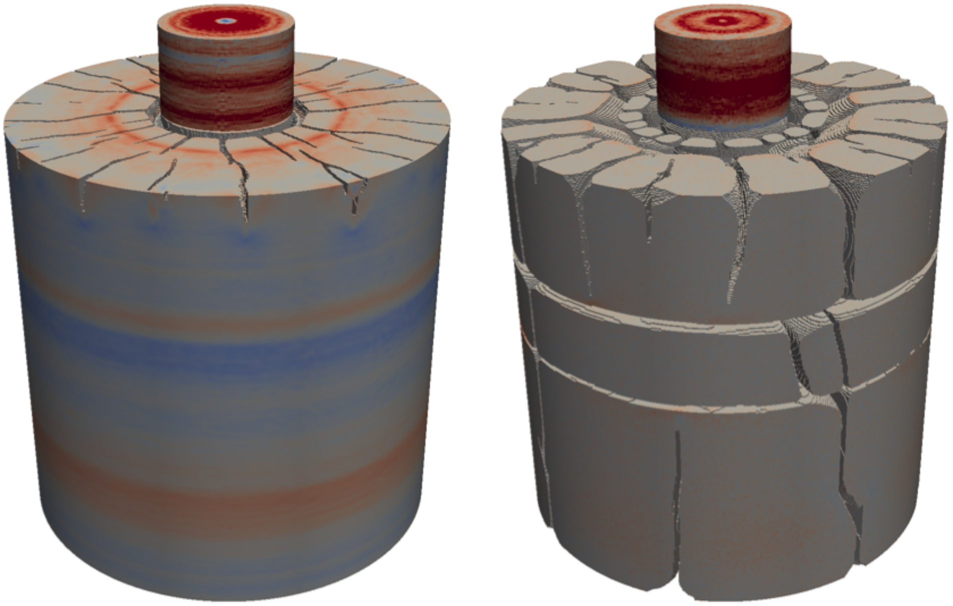


Figure 14: Simulated results for model ceramic impact problem. Spall planes, radial crack, and cone cracks can all be observed.

Torsional fracture

As a second 3D demonstration, a simple torsional fracture simulation was setup. The geometry is a simple cylinder, and the loading is a fixed displacement on the top plane, and a prescribed rotational displacement on the bottom plane. This induces torsion in the specimen, resulting in a max principal stress direction which is at $\sim 45°$ relative to the cylinder. A small defect (in the form of a gap between two elements) was put into the mesh in order to break the rotational symmetry of the problem and to get the problem to localize in a sharp band. The results are shown in figure 15. The expected spiral shaped crack propagation pattern is observed.

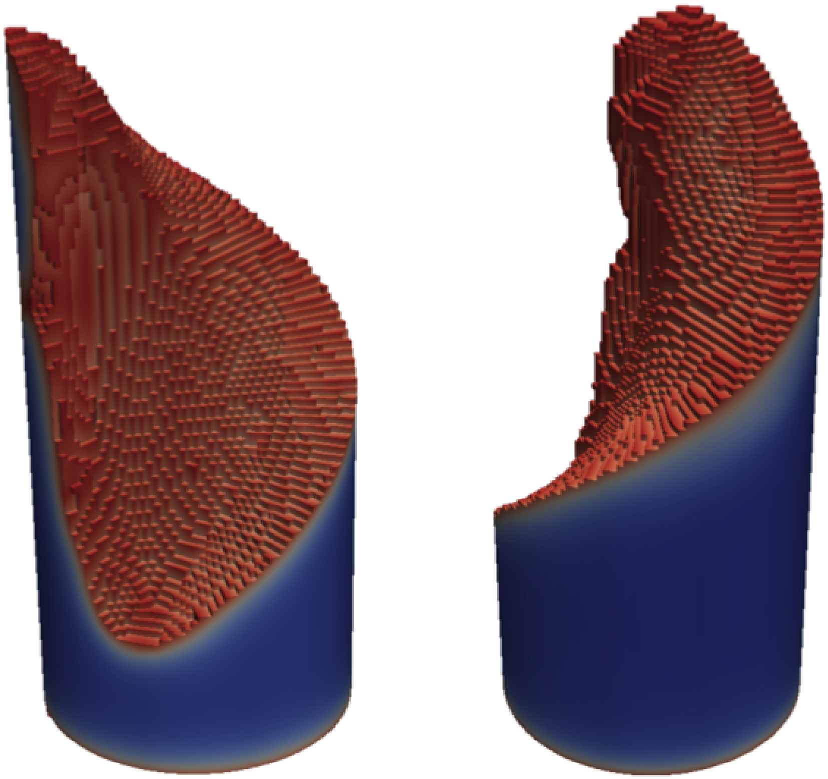


Figure 15: Results for torsion of a brittle cylinder. The expected helical fracture pattern is observed.

2.8 Thread scalability

In addition to the apparent fidelity of the proposed explicit phase-field method, the approach is massively thread scalable, as demonstrated by a GPU (and platform portable) implementation of ***PhaseFieldMini*** using Sandia's Kokkos [23] library. This approach to dynamic fracture simulations can be considered *future proof* in the sense that as the number of threads per processor continues to increase, the method proposed here will continue to see proportional performance gains.[7] Thread scalability results for the implementation using both Intel architectures and Nvidia GPUs are shown in figure 16.

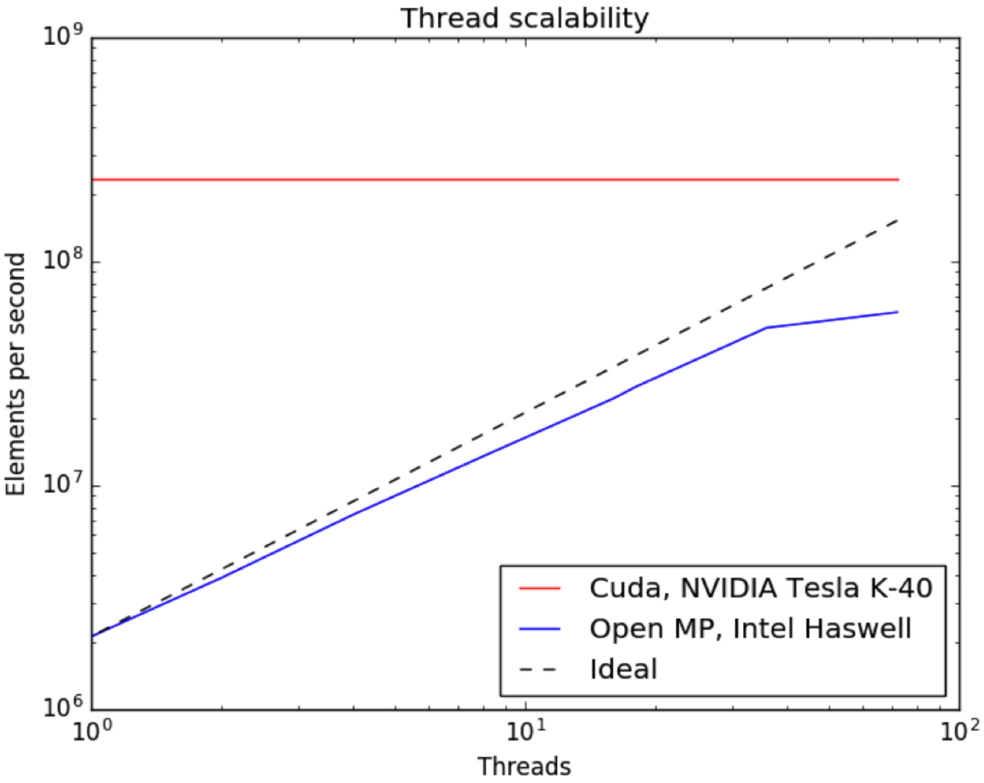
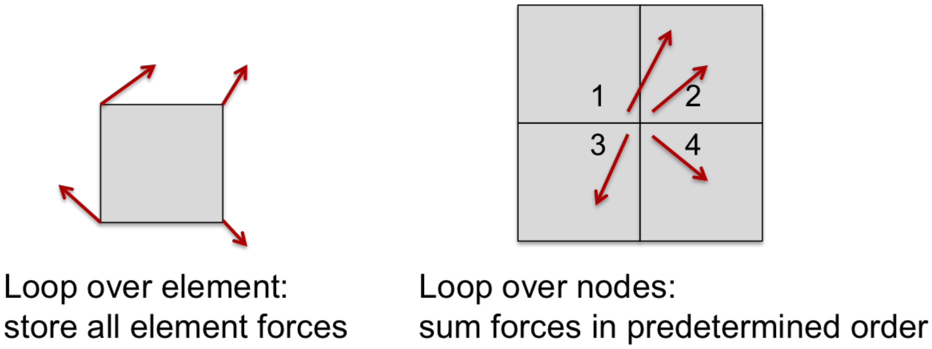


Figure 16: Thread scalability results for a Kokkos [23] implementation of an explicitly integrated cohesive phase-field model. The GPU execution results in over 100 X performance gains relative to a non-vectorized simulation run on a single Intel Haswell thread.

A final concern that is addressed in this implementation (and motivated further by the adjoint calculations in section 3) is that on threaded architectures, run-to-run machine precision repeatability is not ensured. This is especially true with the use of atomic operations. To address this issue, we used a data duplication approach to avoid atomics and to ensure that add orders for nodal force assembly is independent of the number of threads and thread execution order. A simple schematic showing how this works is provided in figure 17. The idea is that all the element forces are computed and then stored element by element on

[7] This is in contract to several alternatives for predicting crack propagation which require significant global communication, localized computation, data movement, and/or mesh modifications.

the mesh. In a separate loop, we loop over all the nodes and sum these stored forces in a predetermined order. The resulting simulations give the same result bit-for-bit independent of the number of threads or the thread execution order. This was found to be critical for reliably checkpointing and recomputing simulation results, as required for the adjoint implementation described in section 3.



Loop over element:
store all element forces

Loop over nodes:
sum forces in predetermined order

Figure 17: Schematic demonstrating an approach for maintaining determinacy in multi-threaded simulations. Element forces are stored globally and a second loop over nodes is used to assemble these element forces in a predetermined order.

3 Inverse methods for dynamic fracture simulations

Significant effort has gone into accurately and reliably predicting crack propagation in engineering materials. The previous section of this report describes some non-trivial improvements in this area. While there is still more work to be done in improving the predictive capabilities of fracture simulations, predictions in-and-of themselves are often not of ultimate interest. More often the real objective of simulation codes is to use predictions to make decisions, improve designs, or determine physical conditions/parameters which led to observed results in the real world. This is where inverse methods come in. At a basic level, inverse methods solve optimization problems with the constraint that the appropriate physical equations (say linear elasticity, or phase-field fracture) are satisfied. Inverse methods are behind technologies such as topology optimization, parameter calibration, automated design, optimal control etc. However, little work has been done in the area of applying inverse methods to problems involving crack initiation and propagation. This work begins closing that gap by developing techniques and expertise to perform these types of inverse problems where the physics of interest is transient and highly nonlinear. The application here focuses on the automated design of heterogeneous brittle materials to maximize crack resistance. However, the techniques developed here and the conclusions drawn are generally applicable to other highly nonlinear transient problems.

There are several inverse problems involving fracture of interest to the engineering community, including:

- **Non-destructive evaluation:** determine the existing cracks in a structure given observations of how waves propagate.
- **Crack forensics:** determine what loads were applied to a structure that caused it to damage/fracture in an observed way.
- **Nonlinear topology optimization:** determine the optimal design (shape and topology) of a structure to minimize the likelihood of crack propagation.
- **Heterogeneous material design:** design the layout of a heterogeneous micro-structure to maximize crack resistance.

There is a common theme in all the above mentioned inverse problems: these problems are all characterized by having a large number of optimization/design variables. Furthermore, the number of design variables scales with the resolution of the problem discretization. For example, the number of design variables in non-destructive evaluation, topology optimization and material design is proportional to the number of elements. For the crack forensics problem, the number of design variables scales with the number of time steps times the number of active boundary nodes. This presents a significant challenge to traditional simulation codes, as they are only designed to predict a single scenario at a time. In order for an optimization procedure to be computationally feasible for these problems (especially as the mesh resolution is increased), it is necessary to be able to efficiently compute the sensitivity of a quantity of interest with respect to a large number of design variables. The mathematical approach to overcome this limitation is called the adjoint method (also called backpropagation or reverse-mode automatic differentiation in different communities).

To motivate this method, first consider the alternatives: finite differencing or propagating sensitivities forward in simulation code. Both of these approaches are computationally infeasible as the number of design variables gets large. In particular if we take $\mathcal{T}$ to be the average time to run a single simulation and evaluate the quantity of interest, the standard approaches to compute sensitivities would require $t_{\text{sim}} = M\mathcal{T}$, where M is the number of design variables. For realistic problems of engineering interest, both M and $\mathcal{T}$ scale with the problems size, and can each be $> 1e8$. By contract, using the adjoint method these same sensitivities can be computed with the same (or increased) precision at a cost which scales as $t_{\text{sim}} = \alpha\mathcal{T} + M$, a cost reduction of up to 8 orders of magnitude (or more as problem sizes continue to increase). Depending on the physics and implementation, the time multiplication factor α can vary from $\alpha = 1$ to about $\alpha = 10$ in the worst case. For the physics considered here, an α of around 6 was required due to the fact that dynamic checkpointing and recomputation is necessary if the problem size (including every time step's results) is too large to fit entirely in memory [24]. In the proceeding, we provide a quick (and hopefully intuitive) derivation of the adjoint method for the case of an explicit update rule, and also for the case of an implicit update rule. In addition, we derive the basic equations necessary to compute adjoints for the case of an explicit phase-field model coupled to a linear-elastic continuum model. Finally, we show preliminary results using these efficient and accurate sensitivities for heterogeneous design of a brittle composite to maximize crack resistance.

3.1 Adjoint for explicit recursion relations

Here we provide a brief derivation of the adjoint equation for explicit recursion relations. This derivation is inspired by the one in [25], but differs in its use of Lagrange multipliers.[8]

Suppose we have a simulation defined by a recursion relation (e.g., any explicit dynamic code):

$$\mathbf{u}^n = \mathbf{h}^n(\mathbf{u}^{n-1}; \boldsymbol{\theta}), \tag{10}$$

with $\mathbf{u}^n \in \mathbb{R}^N$ and $\boldsymbol{\theta} \in \mathbb{R}^M$. It is assumed that $\mathbf{u}^0(\boldsymbol{\theta})$ is known, and an objective function / quantity of interest (qoi) is given as

$$f(\mathbf{u}, \boldsymbol{\theta}) = \sum_{n=1}^{N} g^n(\mathbf{u}^n; \boldsymbol{\theta}),$$

[8] The adjoints derived in this report are sensitivities for the discretized equations. This is in contrast to the PDE's adjoint, which can often be derived analytically and then discretized. The author's view is that the adjoint to the discretized equations is more useful in practice because it corresponds to the intuitive notion of a sensitivity in a simulation, i.e., it matches a finite difference approximation.

and we want to compute the sensitivity of the qoi, f, with respect to the simulation parameters $\boldsymbol{\theta}$:

$$\frac{df}{d\boldsymbol{\theta}} = \sum_{n=1}^{N} \frac{dg^n}{d\boldsymbol{\theta}}.$$

Using the chain rule and the notation $f_{,\boldsymbol{\theta}} := \frac{\partial f}{\partial \boldsymbol{\theta}}$ and $f_{,\mathbf{u}} := \frac{\partial f}{\partial \mathbf{u}}$ one finds:

$$\frac{df}{d\boldsymbol{\theta}} = \sum_{n=1}^{N} {g^n}_{,\boldsymbol{\theta}} + {g^n}_{,\mathbf{u}^n} \frac{d\mathbf{u}^n}{d\boldsymbol{\theta}},$$

where $\frac{d\mathbf{u}^n}{d\boldsymbol{\theta}}$ must be computed recursively using the chain rule. This results in a fairly complicated calculation that is computationally demanding. In particular, the terms $\frac{d\mathbf{u}^n}{d\boldsymbol{\theta}}$ require computing and storing NM values. For N and M large (as is typical in large PDE constrained optimization problems) this approach becomes computationally infeasible and scales as the cost of computing sensitivities numerically, e.g., using finite differences.

To avoid the expense of computing the sensitivities using this *forward propagation* method, we instead construct the Lagrangian

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}; \boldsymbol{\theta}) = \sum_{n=1}^{N} g^n\left(\mathbf{u}^n\right) + \langle \boldsymbol{\lambda}^n, \mathbf{h}^n(\mathbf{u}^{n-1}) - \mathbf{u}^n \rangle,$$

where $\langle a, b \rangle$ denotes an inner product. With the recursion relation equation (10) satisfied, the inner product terms in the Lagrangian are all zero for any choice of the Lagrange multipliers $\boldsymbol{\lambda}^n$, so we have

$$f(\mathbf{u}, \boldsymbol{\theta}) = \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}; \boldsymbol{\theta}) \quad \text{and therefore} \quad \frac{df}{d\boldsymbol{\theta}} = \frac{d\mathcal{L}}{d\boldsymbol{\theta}}.$$

Using the chain rule again, the sensitivity of the qoi is

$$\begin{aligned}\frac{df}{d\boldsymbol{\theta}} = \frac{d\mathcal{L}}{d\boldsymbol{\theta}} = &\sum_{n=1}^{N} {g^n}_{,\boldsymbol{\theta}} + \langle g^n_{,\mathbf{u}}, \frac{d\mathbf{u}^n}{d\boldsymbol{\theta}} \rangle + \langle \boldsymbol{\lambda}^n, {\mathbf{h}^n}_{,\boldsymbol{\theta}} + {\mathbf{h}^n}_{,\mathbf{u}} \frac{d\mathbf{u}^{n-1}}{d\boldsymbol{\theta}} - \frac{d\mathbf{u}^n}{d\boldsymbol{\theta}} \rangle \\ = &{g^N}_{,\boldsymbol{\theta}} + \langle \boldsymbol{\lambda}^N, {\mathbf{h}^N}_{,\boldsymbol{\theta}} \rangle + \langle g^N_{,\mathbf{u}} - \boldsymbol{\lambda}^N, \frac{d\mathbf{u}^N}{d\boldsymbol{\theta}} \rangle + \\ &\sum_{n=1}^{N-1} {g^n}_{,\boldsymbol{\theta}} + \langle \boldsymbol{\lambda}^n, {\mathbf{h}^n}_{,\boldsymbol{\theta}} \rangle + \langle g^n_{,\mathbf{u}} + \left(\mathbf{h}^{n+1}_{,\mathbf{u}}\right)^T \boldsymbol{\lambda}^{n+1} - \boldsymbol{\lambda}^n, \frac{d\mathbf{u}^n}{d\boldsymbol{\theta}} \rangle + \\ &\langle \left(\mathbf{h}^1_{,\mathbf{u}}\right)^T \boldsymbol{\lambda}^1, {\mathbf{u}^0}_{,\boldsymbol{\theta}} \rangle.\end{aligned}$$

As mentioned previously, terms like $\frac{d\mathbf{u}^n}{d\boldsymbol{\theta}}$ are hard and expensive to compute directly, but we are free to choose the adjoint variables $\boldsymbol{\lambda}^n$ in a convenient way. In particular, if we start with $\boldsymbol{\lambda}^N = g^N_{,\mathbf{u}}$, and use a recursion relation on the adjoint variables

$$\boldsymbol{\lambda}^n = g^n_{,\mathbf{u}} + \left(\mathbf{h}^{n+1}_{,\mathbf{u}}\right)^T \boldsymbol{\lambda}^{n+1}, \tag{11}$$

we find

$$f_{,\boldsymbol{\theta}} = \sum_{n=1}^{N} \left(g^{n}{}_{,\boldsymbol{\theta}} + \langle \boldsymbol{\lambda}^{n}, \mathbf{h}^{n}{}_{,\boldsymbol{\theta}} \rangle \right) + \langle \left(\mathbf{h}^{1}_{,\mathbf{u}} \right)^{T} \boldsymbol{\lambda}^{1}, \mathbf{u}^{0}{}_{,\boldsymbol{\theta}} \rangle. \tag{12}$$

Note that the adjoint variables are integrated backward in time, starting with a *final condition* on $\boldsymbol{\lambda}^N$. The computation of the previous adjoint variable $\boldsymbol{\lambda}^{n-1}$ depends on both $\boldsymbol{\lambda}^n$ and $\mathbf{u}^{n-1}$. Finally, we note that nothing in either the adjoint equation (11) or the sensitivity equation (12) scales as NM. In fact, the cost of computing M parameter sensitivities has a computational complexity similar to evaluating the forward recursion relation equation (10) once!

3.2 Adjoint for implicit recursion relations

For completeness, we generalize the results from the previous section to implicit recursion relations (essentially any physics simulation fits into this form):

$$\mathbf{h}^{n}(\mathbf{u}^{n-1}, \mathbf{u}^{n}; \boldsymbol{\theta}) = \mathbf{0}$$

with $\mathbf{u}^0(\boldsymbol{\theta})$ given, and an objective

$$\sum_{n=1}^{N} g^{n}(\mathbf{u}^{n}; \boldsymbol{\theta}).$$

The Lagrangian is:

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}; \boldsymbol{\theta}) = g^{n}\left(\mathbf{u}^{n}\right) + \langle \boldsymbol{\lambda}^{n}, \mathbf{h}^{n}(\mathbf{u}^{n-1}, \mathbf{u}^{n}) \rangle,$$

where the summation over n is implied here.

The sensitivities are

$$\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = g^{n}{}_{,\boldsymbol{\theta}} + \langle g^{n}_{,\mathbf{u}}, \frac{d\mathbf{u}^{n}}{d\boldsymbol{\theta}} \rangle + \langle \boldsymbol{\lambda}^{n}, \mathbf{h}^{n}{}_{,\boldsymbol{\theta}} + \mathbf{h}^{n}_{,\mathbf{u}} \frac{d\mathbf{u}^{n-1}}{d\boldsymbol{\theta}} + \mathbf{h}^{n}_{,\mathbf{v}} \frac{d\mathbf{u}^{n}}{d\boldsymbol{\theta}} \rangle$$

or

$$\begin{aligned} \frac{d\mathcal{L}}{d\boldsymbol{\theta}} =& g^{N}{}_{,\boldsymbol{\theta}} + \langle \boldsymbol{\lambda}^{N}, \mathbf{h}^{N}{}_{,\boldsymbol{\theta}} \rangle + \langle g^{N}_{,\mathbf{u}} + \left(\mathbf{h}^{N}_{,\mathbf{v}} \right)^{T} \boldsymbol{\lambda}^{N}, \frac{d\mathbf{u}^{n}}{d\boldsymbol{\theta}} \rangle + \\ & \sum_{n=1}^{N-1} g^{n}{}_{,\boldsymbol{\theta}} + \langle \boldsymbol{\lambda}^{n}, \mathbf{h}^{n}{}_{,\boldsymbol{\theta}} \rangle + \langle g^{n}_{,\mathbf{u}} + \left(\mathbf{h}^{n+1}_{,\mathbf{u}} \right)^{T} \boldsymbol{\lambda}^{n+1} + \left(\mathbf{h}^{n}_{,\mathbf{v}} \right)^{T} \boldsymbol{\lambda}^{n}, \frac{d\mathbf{u}^{n}}{d\boldsymbol{\theta}} \rangle + \\ & \langle \left(\mathbf{h}^{1}_{,\mathbf{u}} \right)^{T} \boldsymbol{\lambda}^{1}, \mathbf{u}^{0}{}_{,\boldsymbol{\theta}} \rangle, \end{aligned}$$

where we use the notation $\mathbf{h}_{,\mathbf{u}} := \frac{\partial \mathbf{h}(\mathbf{u},\mathbf{v})}{\partial \mathbf{u}}$ and $\mathbf{h}_{,\mathbf{v}} := \frac{\partial \mathbf{h}(\mathbf{u},\mathbf{v})}{\partial \mathbf{v}}$.

Terms like $\frac{d\mathbf{u}^n}{d\boldsymbol{\theta}}$ are hard to compute directly, so we choose the *adjoint* variables $\boldsymbol{\lambda}^n$ in a convenient way. In particular, if we start by solving

$$\left(\mathbf{h}^N_{,\mathbf{v}}\right)^T \boldsymbol{\lambda}^N + g^N_{,\mathbf{u}} = \mathbf{0},$$

and use an implicit recursion relation on the adjoint variables

$$\left(\mathbf{h}^n_{,\mathbf{v}}\right)^T \boldsymbol{\lambda}^n + \left(\mathbf{h}^{n+1}_{,\mathbf{u}}\right)^T \boldsymbol{\lambda}^{n+1} + g^n_{,\mathbf{u}} = \mathbf{0},$$

we find

$$\begin{aligned}\mathcal{L}_{,\boldsymbol{\theta}} =& g^N{}_{,\boldsymbol{\theta}} + \langle \boldsymbol{\lambda}^N, \mathbf{h}^N{}_{,\boldsymbol{\theta}} \rangle + \\ & \sum_{n=1}^{N-1} g^n{}_{,\boldsymbol{\theta}} + \langle \boldsymbol{\lambda}^n, \mathbf{h}^n{}_{,\boldsymbol{\theta}} \rangle + \\ & \langle \left(\mathbf{h}^1_{,\mathbf{u}}\right)^T \boldsymbol{\lambda}^1, \mathbf{u}^0{}_{,\boldsymbol{\theta}} \rangle.\end{aligned}$$

This completes the derivation.

3.3 Adjoint for an explicit dynamic time step update

For this work, we are primarily interesting in explicit dynamic continuum mechanics simulations. One very common integration scheme in this field is the so-called explicit Newmark or central difference time integration algorithm [16]. Given a material and mass model for computing accelerations $\mathbf{a}$ from deformed nodal coordinates $\mathbf{x}$:

$$\mathbf{a}^n = \boldsymbol{\pi}^n(\mathbf{x}^n),$$

the explicit Newmark algorithm can we written

$$\begin{aligned}\mathbf{x}^{n+1} &= \mathbf{x}^n + \Delta t\, \mathbf{v}^n + \tfrac{\Delta t^2}{2}\mathbf{a}^n \\ \mathbf{v}^{n+1/2} &= \mathbf{v}^n + \tfrac{\Delta t}{2}\mathbf{a}^n \\ \mathbf{a}^{n+1} &= \boldsymbol{\pi}^{n+1}(\mathbf{x}^{n+1}) \\ \mathbf{v}^{n+1} &= \mathbf{v}^{n+1/2} + \tfrac{\Delta t}{2}\mathbf{a}^{n+1},\end{aligned} \tag{13}$$

where $\mathbf{v}$ are the velocities and Δt is the time step increment. This is an explicit update rule for the simulation's state (displacement, velocity and acceleration) at step n given the state at step $n-1$. To compute the corresponding explicit adjoint using equation (11), we identify the state $\mathbf{u}$ as

$$\mathbf{u}^n := \begin{bmatrix} \mathbf{x}^n \\ \mathbf{v}^n \\ \mathbf{a}^n \end{bmatrix}.$$

The state update rule $\mathbf{u}^n = \mathbf{h}^n(\mathbf{u}^{n-1}, \boldsymbol{\theta})$ is given by the explicit update equations (13). The adjoint state is

$$\boldsymbol{\lambda}^n = \begin{bmatrix} \hat{\mathbf{x}}^n \\ \hat{\mathbf{v}}^n \\ \hat{\mathbf{a}}^n \end{bmatrix},$$

where

$$\hat{\mathbf{x}}^n = \frac{\partial \langle \mathbf{h}^{n+1}, \hat{\mathbf{x}}^{n+1} \rangle}{\partial \mathbf{x}^n}$$

$$\hat{\mathbf{v}}^n = \frac{\partial \langle \mathbf{h}^{n+1}, \hat{\mathbf{x}}^{n+1} \rangle}{\partial \mathbf{v}^n}$$

$$\hat{\mathbf{a}}^n = \frac{\partial \langle \mathbf{h}^{n+1}, \hat{\mathbf{x}}^{n+1} \rangle}{\partial \mathbf{a}^n}.$$

For the explicit Newmark case, the updates on the Lagrange multipliers can be simplified to

$$\begin{aligned} \hat{\mathbf{a}}^{n+1/2} &= \hat{\mathbf{a}}^{n+1} + \tfrac{\Delta t}{2} \hat{\mathbf{v}}^{n+1} \\ \hat{\mathbf{x}}^n &= \hat{\mathbf{x}}^{n+1} + \hat{\mathbf{a}}^{n+1/2} \cdot \boldsymbol{\pi}^{n+1}_{,\mathbf{x}} \\ \hat{\mathbf{v}}^n &= \hat{\mathbf{v}}^{n+1} + \Delta t \, \hat{\mathbf{x}}^n \\ \hat{\mathbf{a}}^n &= \tfrac{\Delta t}{2} \hat{\mathbf{v}}^n. \end{aligned} \tag{14}$$

In addition to the adjoint relation (11), we must also supply the sensitivity of the state updates to the parameters:

$$\frac{\partial \langle \mathbf{h}^{n+1}, \hat{\mathbf{x}}^{n+1} \rangle}{\partial \boldsymbol{\theta}}, \tag{15}$$

and also the sensitivities of the qoi to each state and the parameters.

A new C++ computational framework ***Springbok*** was developed to manage all of these incremental sensitivity equations. As described in Appendix A, a user can supply a forward recursion relation (e.g. equation (13)), the corresponding adjoint recursion relation (e.g. equation (14)), the state update parameter sensitivities (e.g. equation (15)) and the qoi sensitivities. With this, the library will manage the evolution of the simulation and adjoint calculations, and will return with both the final qoi and its sensitivity with respect to all the input parameters. This framework was essential for achieving the results presented in section 3.5.

3.4 Adjoint for an explicit dynamic time step update with damage evolution

In addition to the standard coordinate updates used in classical continuum mechanics models, we are also interested in a coupled phase-field damage evolution. Assuming an explicit Euler update rule for the phase field and explicit Newmark for the mechanics results in the following explicit update rule:

$$\begin{aligned}
\mathbf{x}^{n+1} &= \mathbf{x}^{n} + \Delta t\, \mathbf{v}^{n} + \tfrac{\Delta t^2}{2}\mathbf{a}^{n} \\
\mathbf{d}^{n+1} &= \mathbf{d}^{n} + \Delta t\, \mathbf{w}^{n} \\
\mathbf{v}^{n+1/2} &= \mathbf{v}^{n} + \tfrac{\Delta t}{2}\mathbf{a}^{n} \\
\mathbf{a}^{n+1} &= \boldsymbol{\pi}^{n+1}\left(\mathbf{x}^{n+1}, \mathbf{d}^{n+1}\right) \\
\mathbf{w}^{n+1} &= \boldsymbol{\omega}^{n+1}\left(\mathbf{x}^{n+1}, \mathbf{d}^{n+1}\right) \\
\mathbf{v}^{n+1} &= \mathbf{v}^{n+1/2} + \tfrac{\Delta t}{2}\mathbf{a}^{n+1},
\end{aligned}$$

where $\mathbf{d}$ is the nodal damage field and $\mathbf{w} \approx \dot{\mathbf{d}}$ is the discretized damage-rate.

Similar to the derivation of the explicit Newmark update above, we compute the adjoint update rules for the coupled phase-field/mechanics problem as

$$\begin{aligned}
\hat{\mathbf{a}}^{n+1/2} &= \hat{\mathbf{a}}^{n+1} + \tfrac{\Delta t}{2}\hat{\mathbf{v}}^{n+1} \\
\hat{\mathbf{x}}^{n} &= \hat{\mathbf{x}}^{n+1} + \hat{\mathbf{a}}^{n+1/2} \cdot \boldsymbol{\pi}^{n+1}_{,\mathbf{x}} + \hat{\mathbf{w}}^{n+1} \cdot \boldsymbol{\omega}^{n+1}_{,\mathbf{x}} \\
\hat{\mathbf{v}}^{n} &= \hat{\mathbf{v}}^{n+1} + \Delta t\, \hat{\mathbf{x}}^{n} \\
\hat{\mathbf{a}}^{n} &= \tfrac{\Delta t}{2}\hat{\mathbf{v}}^{n} \\
\hat{\mathbf{d}}^{n} &= \hat{\mathbf{d}}^{n+1} + \hat{\mathbf{a}}^{n+1/2} \cdot \boldsymbol{\pi}^{n+1}_{,\mathbf{d}} + \hat{\mathbf{w}}^{n+1} \cdot \boldsymbol{\omega}^{n+1}_{,\mathbf{d}} \\
\hat{\mathbf{w}}^{n} &= \Delta t\, \hat{\mathbf{d}}^{n+1}.
\end{aligned}$$

The computation of the parameter sensitivities is similar to that in section 3.3. These equations were implemented in ***Springbok*** and fully tested. Because of the huge number of time steps required to simulate these dynamic problems ($\gg 1000$), it is impossible to store all the states $\mathbf{u}^n$, which are are required to compute the adjoint recursion relation. As described in Appendix A, dynamic checkpointing is the solution. This necessary involves some duplication of computational effort to recompute states which were unable to be stored. When timing problems involving approximately $120,000$ elements and $10,000$ time steps, we found that the cost of computing both the qoi and adjoint sensitivities (including this checkpointing and recomputation effort), was at most 7 times the cost of a single forward solve. This was using 30 checkpointed states that required about 15 times the memory that a forward simulation alone would have. As these problems involve over $100,000$ sensitivities, this results in a time savings of over $10,000$ relative to finite differencing or forward sensitivities![9] This relative benefit only increases as the problem size gets larger.

[9] Admittedly, this is not as large a benefit as adjoints for linear implicit problems which only take ~ 2 times the effort to compute qoi and sensitivities over qoi alone, or nonlinear implicit adjoint problems which can be $1 + \epsilon$ the cost of a forward solve alone. Nevertheless, the approach results in *significant* savings over alternative sensitivity methods for dynamic problems.

3.5 Heterogeneous material design inverse problem

To demonstrate the potential provided by adjoint-based embedded sensitivities, we provide a demo nonlinear inverse problem. We focus on the heterogeneous material design problem mentioned in the introduction to this section. Recent simulation work using phase-field models of fracture have demonstrated it is possible to significantly improve the effective fracture toughness of a heterogeneous material relative to the individual constituents [26]. For example [26] showed that alternating stiff/compliant layers can increase fracture resistance. In addition, they showed that designed compliant pathways through a stiffer material can encourage the crack to follow a tortuous path, minimizing the total crack extent.

Inspired by this, we set up an inverse optimization problem to try and find a locally optimized solution to this heterogeneous material design problem to minimize the extent of crack propagation.

The formulation is as follows: minimize the total damage evolution by changing the initial material stiffness, element by element:

$$\begin{aligned} \min_{E_e} \sum_{n}^{num\_nodes} & d_n(t_{final}), \\ \text{s.t. } E_{min} \leq & E_e \leq E_{max}, \end{aligned}$$

where we are constraining the stiffness by lower and upper bounds, and where the sum here is a sum over the nodal damage field evaluated at the end of the simulation. There is an additional constraint that the physics satisfies the standard dynamic phase-field fracture model from section 2.1. This objective (to minimize the extent and magnitude of damage evolution in the material) is a surrogate to the real objective of maximizing fracture toughness, but in practice the results are likely to be similar.

The material properties for this simulation assumes that the element by element material stiffness vary between $E_{max} = 47.5e9$ and $E_{min} = 190e9$. The density is $\rho = 8000$ and the fracture energy is $G_c = 4e5$, and the Poisson's ratio is $\nu = 0.3$.

This optimization problem likely has many local minima (as it is not yet sufficiently regularized, a concern for future work), so the choice of initial guesses is important. For the demonstration here, we simply use an initial guess for the stiffness that is randomly chosen, element-by-element between E_{min} and E_{max}. A plot of this initial stiffness field is shown in figure 18.

Applying a dynamic mode-I loading (fixed opposing vertical velocities on the left side of the specimen) results in the crack propagation shown in figure 18.

This constitutes the so called *forward problem.* To compute the inverse design, we use the adjoint calculation described in previous sections. The resulting adjoint sensitivity field for each element's stiffness (the sensitivity of the sum of the damage at the end of the simulation with respect for this initial element stiffness guess) is shown in figure 20. An optimization algorithm will try to minimize the extent of damage and will also tend to drive this adjoint

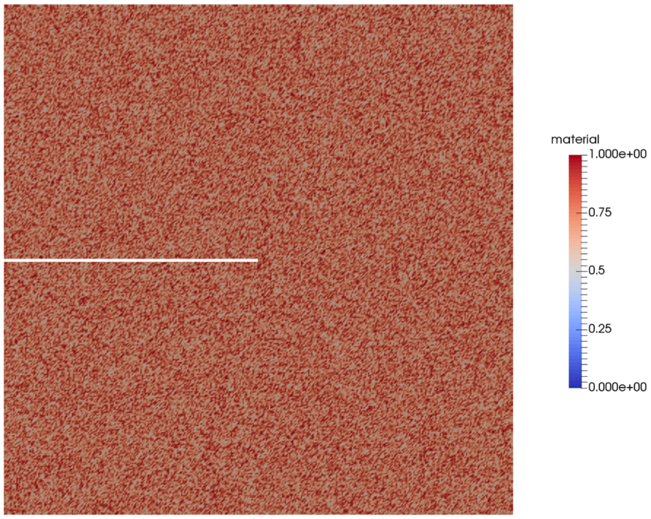


Figure 18: Random initial guess stiffnesses for heterogeneous crack inverse problem.

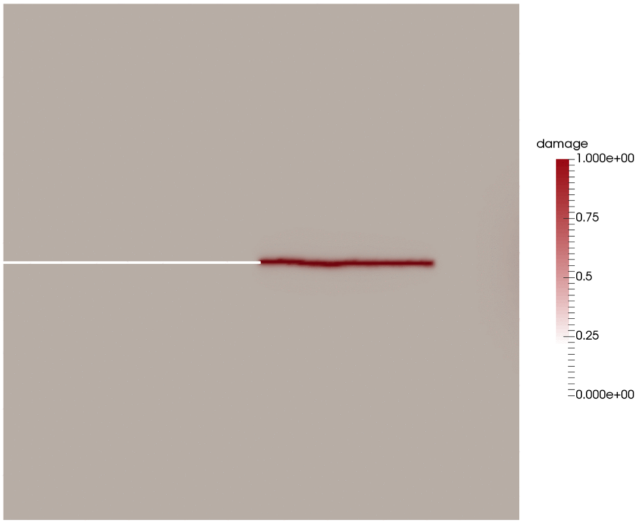


Figure 19: Crack propagation for dynamic mode-I loading of the heterogeneous material from figure 18.

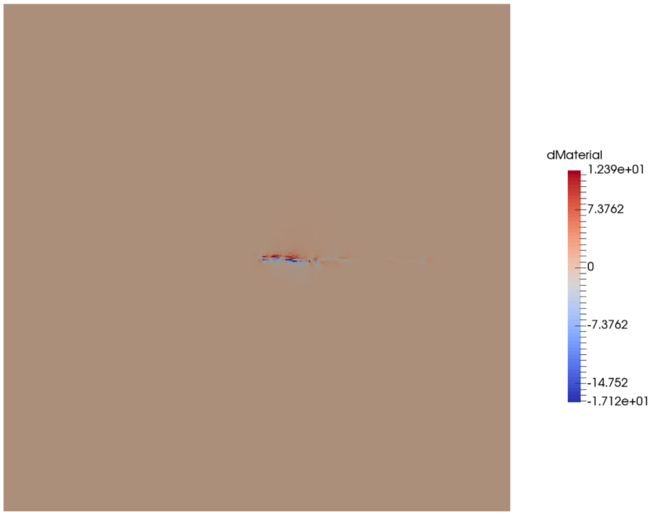


Figure 20: Initial adjoint field: the sensitivity of the total damage with respect to each element's stiffness.

field to as close to zero as possible (it cannot completely drive the derivatives to zero because of the constraint that the element stiffness are bounded).

With a qoi and design sensitivities in hand, we interface to Sandia Rapid Optimization Library (ROL) to perform the optimization. ROL is provided the objective function, the gradients of the objective function and the bound constraints on the element stiffness. The result from this optimization procedure is shown in figures 21 and 22. Figure 21 shows the optimized crack propagation result. Comparing this result to the one in Figure 19, it is clear that the optimization procedure has found a way to reduce the crack propagation extend by almost a factor of 2. In addition, the final crack path is curved, similar to the imagined designs suggested in [26]. The corresponding optimized stiffness design is shown in figure 22. Stiffer elements are red, more compliant ones are gray. Here is it clear that the design has chosen a curved complaint section, embedded in a stiffer matrix. This forces the crack to curve and therefore dissipate more energy. In addition, we note the alternating stiff/compliant layers at the trailing end of the crack. This is a residue of the optimization procedure, which found that these alternating layers could mitigate the extent of cracking. These layers are reminiscent of seashell structures, which are similarly designed to maximize toughness.

These results should only be taken as a demonstration of the incredible potential for these methods. While qualitatively the resulting design appears reasonable, in reality a few critical approximations were make which makes it extremely unlikely that the design will be optimal in reality. Among other issues, perhaps the biggest if the fact that the evolution of the phase-field had to be smoothed in order to have a qoi which varied continuously as a function of the input parameters. This issue is discussed in more detail in the next section.

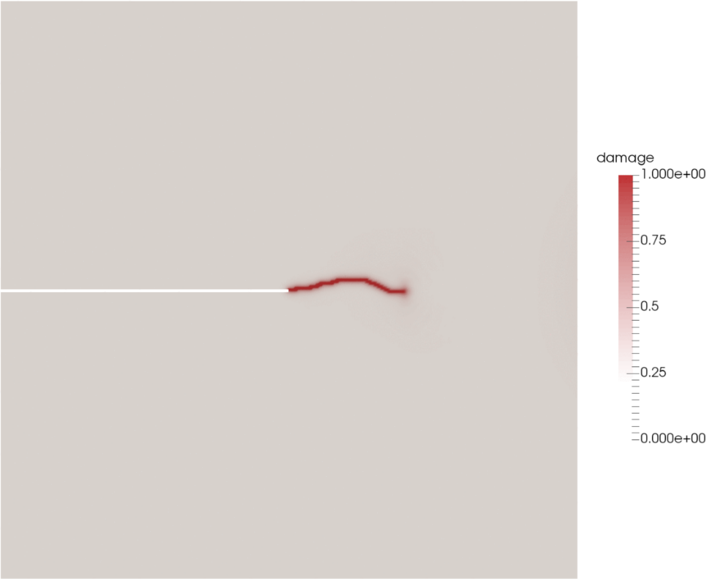


Figure 21: Local optimal solution: extent of crack propagation. Compare this result to the initial guess prediction in Figure 19.

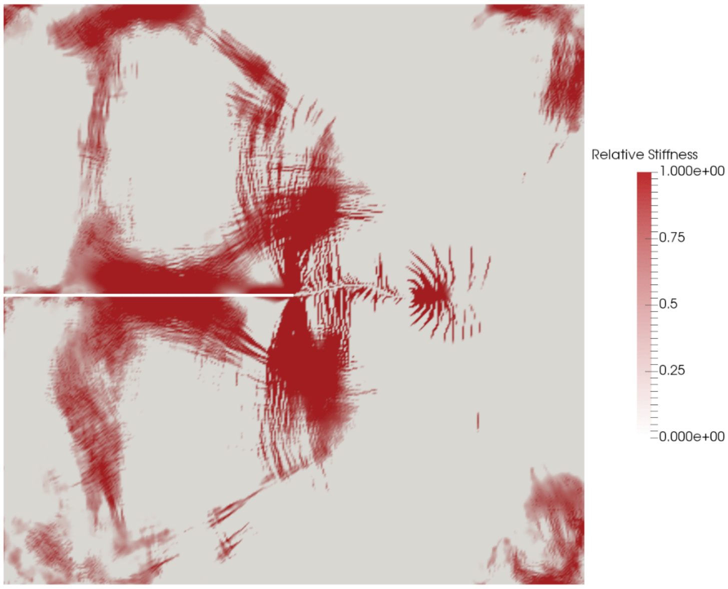


Figure 22: Local optimal solution: material stiffness which minimizes crack propagation for mode-I loading.

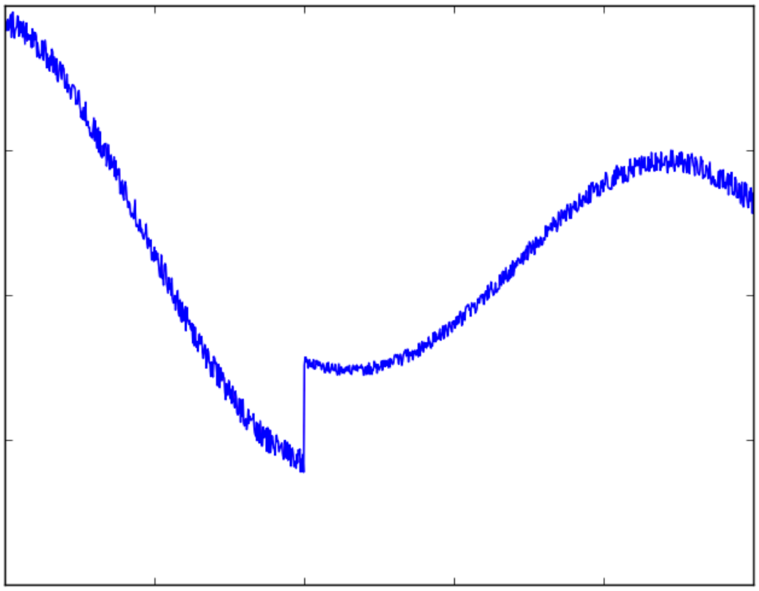


Figure 23: Example of an everywhere non-smooth response function. Plot shows qoi vs. a design variable.

3.6 The smoothness problem

A significant challenge was identified when trying to do inverse problems involving highly nonlinear material models that have a threshold (e.g., a yield stress or an energy-based failure criterion). This includes the phase-field fracture model used here, where there is a constraint that damage evolution can only be positive. It was found that whenever there is a material model threshold which results in a discontinuity, the parameter sensitivities are effectively infinite (in code, the floating point values overflow) for any value of those parameters. This is because when there are enough elements and time step in a simulation, there will inevitably be some material evaluation which is close to a threshold discontinuity. This is true even if the sensitivities are computed using adjoint or finite differences. In the finite difference case, the parameter sensitivities were found to be entirely dependent on the finite difference step size, indicating that these derivatives are undefined to within machine precision. If efficient automated design it to be achieved using large deformation finite element codes, it appears that many of our existing material models (and probably contact algorithms) will need to be reformulated. Follow on work will seek to address these concerns.

To give a more visual representation of this issue, figure 23 shows a made-up function which is effectively discontinuous everywhere (and where the x-axis scale is a small multiple of machine precision). In finite element simulations with a material model involving a threshold at every element (or on element faces for contact), this is a typical resulting response for a qoi vs. input parameters plot. This is worrying, as it means that the local sensitivities are not well defined for these nonlinear solid mechanics simulations. This significantly reduces the usefulness of the adjoint method and challenges UQ and V&V efforts. It is true that a global sensitivity might still be meaningful in these simulations, but global sensitivities are

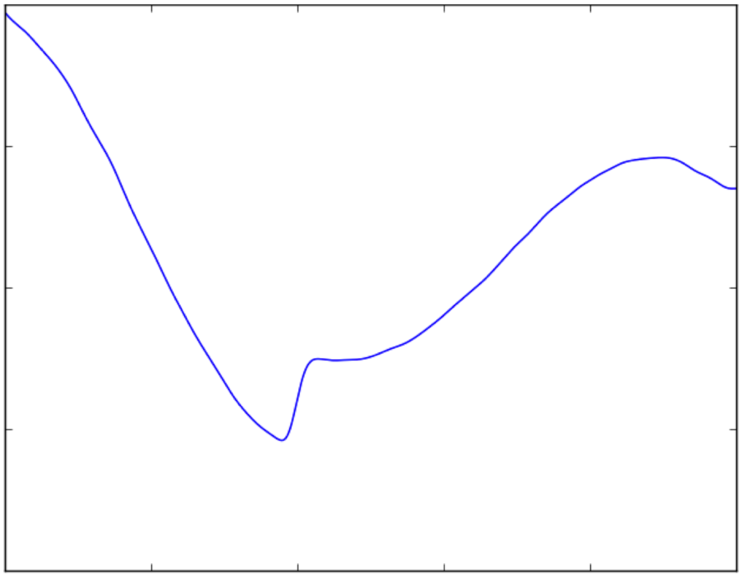


Figure 24: Smoothed version of the response function in figure 23.

very expensive to compute, especially for a large number of parameters. Ideally, we want to smooth the response in our simulation to something like that shown in figure 24. The essential features of the model are still there, as ideally we are only smoothing discontinuities as machine precision.

For the case of the phase-field inverse problem above, we had to smooth the damage evolution constraint. This constraint says

$$\dot{\phi} \geq 0.$$

An example of such as smoothing is shown in figure 25. While the hope is that this approximation would have little impact of the simulation results, that does not appear to be the case. Improvement in this area is critical for getting more realistic and verifiable optimal designs. Going forward, it appears that the biggest challenge now for applying inverse methods to nonlinear continuum mechanics problems will be overcoming this smoothness issue. This problem does not typically exist in other inverse formulations, such as topology optimization, because the responses are smooth (at least almost everywhere). In the nonlinear case, we may have to learn to deal with function that are nowhere smooth.

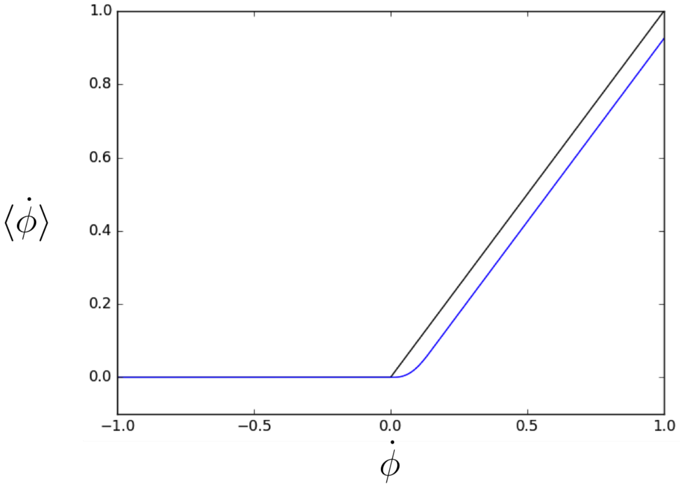


Figure 25: Smoothed approximation to the $\dot{\phi} \geq 0$ constraint.

4 Conclusion

In this report, two distinct developments were presented. The first was an implementation and demonstration of a cohesive phase-field model [2] for brittle fracture simulations. It was argued that a parabolic regularization of the model and an explicit dynamic implementation results in a convergent damage model for both quasi-static and dynamic crack propagation problems. The approach is able to reproduce dynamic fracture benchmark problems with reasonable efficiency. In addition, it was shown to be massively thread scalable, as demonstrated by a GPU implementation using the Kokkos [23] library. This approach to dynamic fracture simulations can be considered *future proof* in the sense that as the number of threads per processor continues to increase, the method proposed here will continue to see proportional performance gains. A version of this explicit gradient damage model has been implemented in Sierra/SolidMechanics. Follow on work will extend the method to more material varieties, e.g., geomaterial failure and ductile failure.

The second significant outcome was the development and demonstration of adjoint-based embedded sensitivities for highly nonlinear dynamic fracture simulations. A C++ computational framework was developed to assist in computing adjoint-based parameter sensitivities, test these sensitivities, and checkpoint the forward simulations to make the adjoint calculations both computationally and memory efficient. It was demonstrated that the sensitivity of a quantity of interest with respect to hundreds of thousands of input parameters can be computed at the cost of less than 7 *forward* crack propagation simulations. This capabilities was interfaced to Sandia's Rapid Optimization Library (ROL) to perform a demonstration PDE constrained optimization problem involving crack propagation: the automated design of a heterogeneous micro-structure to minimize crack propagation. However, a significant challenge was identified when trying to do inverse problems involving nonlinear material models that have a threshold (e.g., a yield stress or an energy-based failure criterion). It was found that whenever there is an element level threshold which results in a slope discontinuity, the simulation's parameter sensitivities are effectively infinite for any value of those parameters. This is true if the sensitivities are computed using the adjoint approach or finite differences. If efficient automated design it to be achieved using large deformation finite element codes, it appears that many of our existing material models (and possibly contact algorithms) will need to be reformulated with smoothness in mind. Follow on work will seek to address these concerns.

References

[1] 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*, vol. 83, pp. 1273–1311, 2010.

[2] E. Lorentz, S. Cuvilliez, and K. Kazymyrenko, "Convergence of a gradient damage model toward a cohesive zone model," *Comptes Rendus Mecanique*, vol. 339, pp. 20–26, 2011.

[3] Z. Bazant and M. Jirasek, "Nonlocal integral formulations of plasticity and damage: Survey of progress," *Journal of Engineering Mechanics*, vol. 128, pp. 1119–1149, 2002.

[4] B. Bourdin, G. Francfort, and J.-J. Marigo, "The variational approach to fracture," *Journal of elasticity*, vol. 91, pp. 5 –148, 2008.

[5] S. Silling, M. Epton, . Weckner, J. Xu, and E. Askari, "Peridynamic states and constitutive modeling," *Journal of Elasticity*, vol. 88, pp. 151–184, 2007.

[6] M. Tupek and R. Radovitzky, "An extended constitutive correspondence formulation of peridynamics based on nonlinear bond-strain measures," *Journal of the Mechanics and Physics of Solids*, vol. 65, pp. 82–92, 2014.

[7] M. Tupek, *Extension of the peridynamic theory of solids for the simulation of materials under extreme loadings.* PhD thesis, Massachusetts Institute of Technology, 2014.

[8] B. Bourdin, G. Francfort, and J.-J. Marigo, "Numerical experiments in revisited brittle fracture," *Journal of the Mechanics and Physics of Solids*, vol. 48, pp. 797–826, 2000.

[9] M. Borden, C. Verhoosel, M. Scott, T. Hughes, and C. Landis, "A phase-field description of dynamic brittle fracture," *ICES REPORT 11-14, The Institute for Computational Engineering and Sciences, The University of Texas at Austin*, 2011.

[10] M. Borden, T. Hughes, C. Landis, and C. 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*, vol. 273, pp. 100–118, 2014.

[11] R. Radovitzky, A. Seagraves, M. Tupek, and L. Noels, "A scalable 3D fracture and fragmentation algorithm based on a hybrid, discontinuous Galerkin, cohesive element method," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, pp. 326–344, 2011.

[12] E. Lorentz, S. Cuvilliez, and K. Kazymyrenko, "Modelling large crack propagation: from gradient damage to cohesive zone models," *International Journal of Fracture*, vol. 178, pp. 85–95, 2012.

[13] T. Li, J.-J. Marigo, D. Guilbaud, and S. Potapov, “Gradient damage modeling of brittle fracture in an explicit dynamics context,” *International Journal for Numerical Methods in Engineering*, 2016.

[14] L. Ambrosio and V. M. Tortorelli, “Approximation of functional depending on jumps by elliptic functional via Γ-convergence,” *Communications on Pure and Applied Mathematics*, pp. 999âĂŞ–1036, 1990.

[15] H. Hilber, T. Hughes, and R. Taylor, “Improved numerical dissipation for time integration algorithms in structural dynamics,” *Earthquake Engineering and Structural Dynamics*, vol. 5, pp. 283–292, 1977.

[16] T. Belytschko and T. Hughes, *Computational Methods for Transient Analysis.* Elsevier Science, North-Holland, 1983.

[17] Sierra/SM Development Team, “Adagio: A 3-D nonlinear solid mechanics finite element application for quasistatic, implicit transient dynamics, and explicit transient dynamics,” Tech. Rep. SAND2014-3257, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California, 2014.

[18] W. Bohme and J. Kalthoff, “The behavior of notched bend specimens in impact testing,” *International Journal of Fracture*, vol. 20, pp. 139–143, 1982.

[19] J. Kalthoff and S. Winkler, “Failure mode transition at high rates of shear loading,” *In C.Y. Chiem, H.D. Kunze and L.W. Meyer, editor, International Conference on Impact Loading and Dynamic Behavior of Materials*, pp. 185–195, 1987.

[20] Z. Liu, T. Menouillard, and T. Belytschko, “An xfem/spectral element method for dynamic crack propagation,” *International Journal of Fracture*, vol. 169, pp. 183–198, 2011.

[21] Z. Zhang and G. Paulino, “Cohesive zone modeling of dynamic failure in homogeneous and functionally graded materials,” *International Journal of Plasticity*, vol. 21, pp. 1195–1254, 2005.

[22] S. Grutzik and E. Reedy, “Crack path selection in thermally loaded borosilicate/steel bibeam specimen,” *In proceedings of the society for experimental mechanics XIII international congress and exposition on experimental and applied mechanics, Orlando, FL: Springer*, 2016.

[23] H. Edwards, C. Trott, and D. Sunderland, “Kokkos: Enabling manycore performance portability through polymorphic memory access patterns,” *Journal of Parallel and Distributed Computing*, vol. 74, pp. 3202–3216, 2014.

[24] Q. Wang, P. Moin, and G. Iaccarino, “Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation,” *SIAM Journal on Scientific Computing*, vol. 31, pp. 2549–2567, 2009.

[25] S. Johnson, "Adjoint methods and sensitivity analysis for recurrence relations," *Course notes for MIT's 18.335*, 2007–2011.

[26] M. Hossain, C.-J. Hsueh, B. Bourdin, and K. Bhattacharya, "Effective toughness of heterogeneous media," *Journal of the Mechanics and Physics of Solids*, vol. 71, pp. 15–32, 2014.

[27] M. Bhardwaj, K. Pierson, G. Reese, T. Walsh, D. Day, K. Alvin, J. Peery, C. Farhat, and M. Lesoinne, "Salinas: A scalable software for high-performance structural and solid mechanics simulations," *In proceedings of Supercomputing, ACM/IEEE*, 2002.

A ***Springbok***: a C++ framework for computing adjoint sensitivities with checkpointing in dynamic simulations

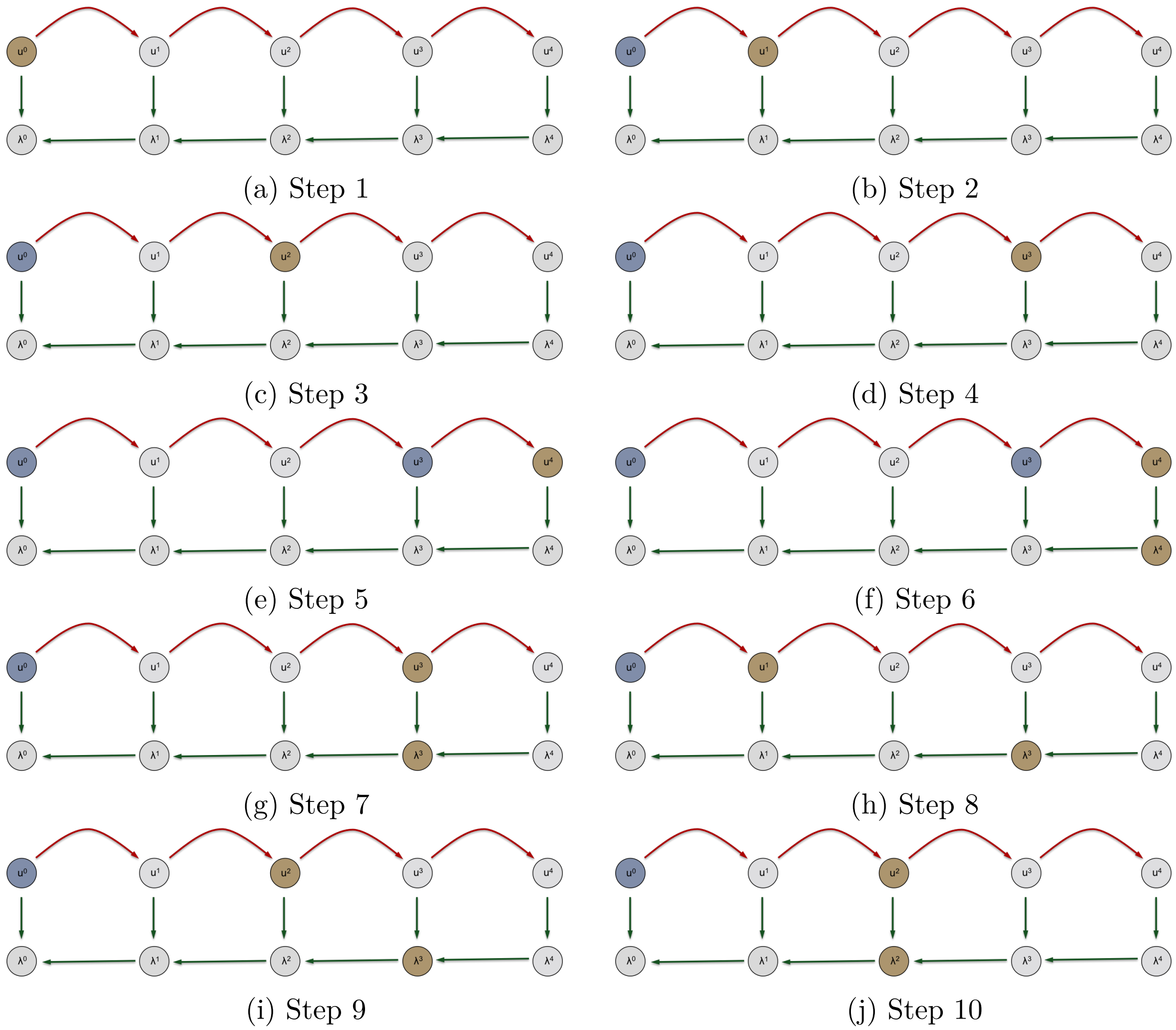
Computing adjoint sensitivities for nonlinear transient simulations is a complicated procedure to efficiently automate. At a minimum, a code which seeks to compute embedded sensitivities using the adjoint method would have to implement:

- A objective function, also called a quantity of interest (qoi). Examples: maximum deflection, temperature at a given location.
- A clear definition of the simulation's state $\mathbf{u}^n$ at step iteration n. The state evolves as the simulation progresses. Examples: displacements, velocities, damage.
- A clear definition of the input parameters $\boldsymbol{\theta}$ to the simulation. Parameters are fixed throughout the simulation. Examples: stiffness, initial density at each material point.
- The sensitivity of the objective function with respect to both design parameters, $\boldsymbol{\theta}$, and the simulation's state $\mathbf{u}^n$ at each step iteration n.
- A rule for updating the simulation's state from n to $n+1$: $\mathbf{u}^{n+1} = \mathbf{h}(\mathbf{u}^n, \boldsymbol{\theta})$.
- The sensitivity of the state update rule with respect to both the previous state $\mathbf{u}^n$ and the parameters $\boldsymbol{\theta}$.
- A condition for determining when the simulation should end.
- A mechanism for retrieving the previous state $\mathbf{u}^{n-1}$. Either each state must be stored or a certain subset of states can be checkpointed and the state $\mathbf{u}^{n-1}$ might have to be recomputed from some earlier state.[10] An example of this checkpointing procedure is shown in figure A.1.

In addition, there are many places in this flow where simple coding mistakes can make the adjoint sensitivity calculation *very* wrong. We have found it necessary to test each term of the adjoint individually and then build up the total sensitivity.

To aid with the implementation and testing of computing adjoints in dynamic simulations, an extensible C++ library ***Springbok*** was designed to efficiently encapsulate these requirements and to help automate the incremental testing of state and objective sensitivities.

[10] The requirement to recompute the same state multiple times when calculating adjoints in dynamics means it is important that the forward simulation is computationally repeatable. This puts a strict requirement on simulations running in a multi-threaded environment, where the use of atomic operations means that bit-by-bit reproducibility in a forward simulation is not guaranteed. For the work presented in this report, a fairly standard data duplication procedure was used to ensure simulation results were independent of the number of threads and thread execution order.



(a) Step 1

(b) Step 2

(c) Step 3

(d) Step 4

(e) Step 5

(f) Step 6

(g) Step 7

(h) Step 8

(i) Step 9

(j) Step 10

Figure A.1: Example dynamic checkpointing algorithm. When computing adjoint solutions going backward in time, λ_n depends on both the adjoint solution one step ahead in time λ_{n+1}, and also on the state u_n at that time. In steps 1–5, we are simulating the state (tan) going forward in time, and occasionally checkpointing (blue). At step 6, we begin to backpropagate the adjoint solution. At first because we checkpointed u_4, we can directly compute the previous adjoint solution. However, when we get to the end of step 7, we do not have access to the current state u_3, so we must *recompute* u_3 from the last checkpointed state, u_0, as shown in steps 8–10.

This framework consists of a few key classes that need to have derived implementations in order to compute adjoint sensitivities for a new discretized PDE or recursion relation:

- **`State`**
- **`Parameters`**
- **`StateUpdateRule`**
- **`Qoi`**
- **`TerminationCondition`**

We'll briefly describe the functionality of each of these classes.

A.1 State

The **`State`** contains all the data in the simulation that is updated from one step iteration to the next. A derived implementation of a state can have any number of variables and fields internal to it. The only essential overrides necessary are a **`double& operator[](int)`** method which takes an integer and returns a floating point value for each index, and a **`int num_dof()`** method which returns the *total* number of floating point values which make up that state.

A.2 Parameters

The **`Parameters`** class contains all the inputs to the simulation that a user wants to compute sensitivities with respect to. It can have arbitrary size, but must provide override methods: **`double& operator[](int)`** and **`int num_parameters()`**. Parameters can not change during a forward simulation, but will change during the course of an optimization procedure when solving the PDE constrained optimization problem.

A.3 StateUpdateRule

The **`StateUpdateRule`** specifies the recursion relation. For explicit dynamics, it is the procedure which updates the current solution to the next step iteration. A **`StateUpdateRule`** is constructed with a **`Parameters`** instance and must implement a **`void update(State& state_old, State& state_new)`** method which defines how **`state_new`** is determined from the parameters and **`state_old`**. In addition, two sensitivity methods must be implemented:

```
update_adjoint(State old, State& rhs, State& dState)
```

update_params_adjoint(State& old, State& rhs, Parameters& dPars).

The sensitivity of the update rule with respect to **state_old** is provided in **update_adjoint**. In this case, **dState** = $\boldsymbol{\lambda}$, must be implemented as

$$\boldsymbol{\lambda} := \frac{\partial \langle \mathbf{h}(\mathbf{u}; \boldsymbol{\theta}), \mathbf{rhs} \rangle}{\partial \mathbf{u}},$$

where $\mathbf{u}$ is **state_old**, $\mathbf{h}$ is the update rule function, and $\mathbf{rhs}$ is an arbitrary input vector which has the size and type of a **State**.

The sensitivity of the update rule with respect to the parameters $\boldsymbol{\theta}$ is provided in **update_params_adjoint**. Here **dPars** = $\boldsymbol{\mu}$, must be implemented as

$$\boldsymbol{\mu} := \frac{\partial \langle \mathbf{h}(\mathbf{u}; \boldsymbol{\theta}), \mathbf{rhs} \rangle}{\partial \boldsymbol{\theta}}.$$

Templated helper functions have been implemented for testing that all of these sensitivities are evaluated correctly for the provided update rule. They are designed so that a new **State** and **StateUpdateRule** can be added and immediately have its sensitivity operators tested with minimal additional coding effort. It is also possible to use automatic differentiation to compute the sensitivity operators directly from the update rule. This has not been done for the work demonstrated here, but might make extensions easier and more maintainable.

A.4 Qoi

The **Qoi** class specifies the quantity of interest (objective function). It is constructed with a **Parameters&** instance. The override methods for this class are **double evaluate(State& state)**, **void dPars(State& state, Parameters& params_deriv)** and **void dState(State& state, State& state_deriv)**. the **evaluate** method returns the quantity of interest as a function of the **State** and **Parameters**. The sensitivity of the quantity of interest with respect to the **Parameters** and **State** are implemented in **dParams** and **dState** respectively. If a quantity of interest only depends on the state at certain step iterations, the implementation of this class must check and only return a non-zero value when appropriate. Generic helper functions have also been implemented to fully test the **Qoi** sensitivity implementations.

A.5 TerminationCondition

The **TerminationCondition** class determines when the simulation ends. It is constructed with a **Parameters** instance and implements a **bool evaluate(State& state)** method which returns true only when the simulation should end.

A.6 Implementation Classes

In addition to the user provided classes described above. Several hidden helper classes are critical for performing the adjoint calculations. Of note are the classes:

- **`StateContainer`**
- **`StateManager`**
- **`Physics`**

The **`StateContainer`** class has derived classes which are templated on the **`State`** type and simply stores a given number of states. In particular, it is allocated to hold the maximum number of states a user thinks can be stored safely in memory. This number is defaulted to 50. The checkpointing algorithm reads and writes to states as necessary from this fixed sized container of states.

The **`StateManager`** implements key parts of the dynamic checkpointing algorithm from [24]. A **`StateManager`** is constructed with a preallocated **`StateContainer`** and a **`StateUpdateRule`**.

The public methods are

- **`void update_state()`**
- **`void reverse_state()`**
- **`void update_adjoint_state()`**
- **`State& get_current_state()`**
- **`State& get_previous_state()`**

The **`void update_state()`** method updates the internal state using a **`StateUpdateRule`**. The **`void reverse_state()`** method fetches the previous state at $n-1$. If that state is checkpointed, it simply returns a reference to it. Otherwise it goes back to the most recently checkpointed state and uses that that to recompute up to the $n-1$ state and then returns a reference to that newly recomputed state. A demonstration of why this is necessary is provided in figure A.1. The method **`void update_adjoint_state()`** uses the **`StateUpdateRule`** to compute the previous adjoint solution from the current one. The methods **`State& get_current_state()`** and **`State& get_previous_state()`** return the current and previous states, respectively.

The **`Physics`** class owns an instance of the following classes: **`Parameter`**, **`State`**, **`StateUpdateRule`**, **`Qoi`**, **`TerminationCondition`**. It uses these classes to allocate and own both a **`StateContainer`** and a **`StateManager`**. The physics is responsible for implementing the part of the dynamic checkpointing algorithm [24] that drives the simulation from the initial state to the final state, determines when to stop, evaluates a qoi, and then reverses to compute the initial adjoint field from the final adjoint. This is in contrast to the **`StateManager`**, which knows nothing about **`Qoi`**, **`Parameter`** or **`TerminationCondition`**. In fact, the **`StateManager`**'s key job is to be able to fetch new and previous states for the **`Physics`** (using the **`StateUpdateRule`**). The algorithms internal to the **`Physics`** and **`StateManager`** classes don't need to be implemented or rederived by users of the library. The only things that change when adding a new recursion relation (e.g., physics equation) are encapsulated in the public classes from the previous subsections.

This design results in a powerful, relatively easy to use, and extensible framework for automatically calculating highly complex explicit update rules and their adjoint sensitivities. It is driven at a fairly high level and is compatible with both MPI and thread-parallel implementations of the underlying physics. In fact, the results presented in this report ran predominantly on Nvidia's Telsa K-40 GPUs. The only significant overhead associated with ***`Springbok`*** is the necessary checkpointing and computational duplication required for computing dynamic adjoints on computers with limited memory space. In fact, if the number of checkpoints is set to be higher than the number actually used in the simulation, there should be negligible overhead in using ***`Springbok`*** relative to a non-checkpointed adjoint implementation.

DISTRIBUTION:

1	MS 0845	Michael Skroch, 1542
1	MS 0845	Timothy Walsh, 1542
1	MS 0845	Jesse Thomas, 1542
1	MS 0845	Julia Plews, 1542
1	MS 0845	Kendall Pierson, 1542
1	MS 0845	Miguel Aguilo, 1542
1	MS 0845	Martin Heinstein, 1542
1	MS 0845	John Dolbow, 1542
1	MS 0845	Wilkins Aquino, 1542
1	MS 0897	Ted Blacker, 1543
1	MS 0828	Walt Witkowski, 1544
1	MS 0845	Kyran Mish, 1544
1	MS 0557	Peter Coffin, 1553
1	MS 0840	Eliot Fang, 1554
1	MS 0840	Joseph Bishop, 1554
1	MS 0840	Pania Newell, 1554
1	MS 0840	Kevin Long, 1554
1	MS 0840	James Cox, 1554
1	MS 0889	Earl David Reedy, Jr., 1556
1	MS 0346	John Emery, 1556
1	MS 0346	Kurtis Ford, 1556
1	MS 1318	Christian Trott, 1426
1	MS 1318	James Stewart, 1441
1	MS 1320	Drew Kouri, 1441
1	MS 1320	Denis Ridzal, 1441
1	MS 1320	Richard Lehoucq, 1442
1	MS 1322	David Littlewood, 1444
1	MS 1322	Stewart Silling, 1444
1	MS 1320	Scott Collis, 1440
1	MS 0845	Scott Hutchinson, 1540
1	MS 1318	Timothy Trucano, 1400
1	MS 0840	Stephen Attaway, 1500
1	MS 0744	John Bignell, 6233
1	MS 9042	Jakob Ostien, 8343
1	MS 9042	James Foulk III, 8343
1	MS 9042	Alejandro Mota, 8343
1	MS 0359	D. Chavez, LDRD Office, 1911
1	MS 0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories