An open-source implementation of a phase-field model for brittle fracture using Gridap in Julia

Mohammad Masiur Rahaman
School of Infrastructure, Indian Institute of Technology Bhubaneswar, Bhubaneswar, India

Received 9 August 2021; accepted 17 December 2021

Abstract
This article proposes an open-source implementation of a phase-field model for brittle fracture using a recently developed finite-element toolbox, Gridap in Julia. This work exploits the advantages of both the phase-field model and Gridap toolbox for simulating fracture in brittle materials. On one hand, the use of the phase-field model, which is a continuum approach and uses a diffuse representation of sharp cracks, enables the proposed implementation to overcome such well-known drawbacks of the discrete approach for predicting complex crack paths as the need for re-meshing, enrichment of finite-element shape functions, and an explicit tracking of the crack surfaces. On the other hand, the use of Gridap makes the proposed implementation very compact and user-friendly that requires low memory usage, and provides a high degree of flexibility to the users in defining weak forms of partial differential equations. Tests on a single-edge notched plate under tension, an L-shaped panel, a notched plate with a hole, a notched beam under symmetric three-point bending and a notched beam with three holes under asymmetric three-point bending are considered to demonstrate how the proposed Gridap-based phase-field Julia code can be used to simulate fracture in brittle materials.

Keywords
Phase-field, open-source, Gridap, Julia, brittle fracture, continuum approach

1. Introduction
To design structures with high reliability, failure analysis of structures is of great importance in engineering applications. As fracture due to crack initiation and propagation is one of the most often encountered failure modes in engineering materials and structures, modeling of fracture in solids has always been one of the most intriguing topics of research interests. Numerical modeling of fracture in solids has mainly been done either by using a discrete or a continuum approach. In the discrete approach, cracks in the material body are modeled as the discontinuity of the displacements in the domain whereas in the continuum approach a diffused approximation of cracks is used to model fracture as a continuum damage process for which displacements are continuous but the material stiffness gradually degrades. Linear elastic fracture mechanics (LEFM) [1–4] and cohesive zone model (CZM) [5,6] are the notable theories in the category of the discrete approach for fracture modeling. Although LEFM and CZM are very
popular, their implementation requires an explicit tracking of the discontinuity in the displacement field that poses difficulty in modeling an arbitrary complex crack path.

Knowing the well-known drawbacks of the discrete approach for modeling complicated crack paths, researchers generally refer to the continuum approach that provides the crack paths as part of the solutions of the governing partial differential equations (PDEs). One of the most popular theories in the category of continuum approach is the phase-field model (PFM) [7]. There are of course several phase-field approaches to brittle fracture that have been independently developed in the mechanics community [8–16] and in the physics community [17–22] as well. In this article, a PFM proposed by Dhas et al. [16] is adopted as the model provides a thermodynamically consistent way of accommodating dissipative energy effects whenever needed. In all the PFMs, a diffused approximation of sharp cracks is used by introducing a length-scale parameter and an internal variable called phase field. The accuracy of the diffused approximation depends on the value of the length-scale parameter and may represent the original crack problem if the length-scale parameter is chosen sufficiently small. Although this feature of PFMs imposes a highly efficient implementation of the model while using the finite-element method as very fine meshes are required for regularizing the sharp cracks with a small value of length-scale parameter, the model has gained huge popularity in the research community, as it can be incorporated in commercial finite-element software such as Abaqus [23–28]. However, to make the PFM available for a wider class of practitioners and researchers, there are also attempts toward open-source implementation of PFMs by using finite-element method [29] and machine learning–based approaches [30,31].

In this article, a new open-source implementation of a PFM is proposed using a recently developed finite-element toolbox Gridap available in the programming language Julia [32,33] that shares the advantages of both the static and dynamic languages. The programming language Julia is computationally efficient as the static languages such as Fortran, C++, and so on, and also easy to use as the dynamic languages like MATLAB, Python, Mathematica, and so on. Gridap is an extensible finite-element toolbox [34,35] in Julia that can be used to solve a wide range of physical problems modeled mathematically using PDEs. In contrast to other finite-element libraries written in Julia, such as FinEttools, JuAFEM, and JuliaFEM, [36] Gridap uses a novel software design (for example, high-level application programming interface (API) calls) that enables one to compute the value for a specific cell on the fly and never store the values for all cells in the mesh simultaneously and thus essentially requires very low memory usage. Moreover, Gridap provides a high degree of flexibility to the users as they can implement any PDEs-based mathematical model such as a PFM using a very compact syntax without explicitly writing any for-loop for assembly over elemental matrices. To develop a Gridap-based open-source program for phase-field modeling of brittle fracture, a thermodynamically consistent PFM is briefly described first in Section 2. Then, the derivation of the weak form corresponding to the governing PDEs of the PFM and the finite-element implementation in Julia are provided in Section 3. Successful implementation of the PFM using Gridap is demonstrated in Section 4 through numerical simulations of brittle fracture in a single-edge notched plate under tension, an L-shaped panel, a notched plate with a hole, a notched beam under symmetric three-point bending, and a notched beam with three holes under asymmetric three-point bending. Finally, the outcomes of this work are summarized, and concluding remarks are accordingly made in Section 5.

2. PFM

In this section, a brief description of a thermodynamically consistent phase-field approach [16] to brittle fracture in elastic solids, under small strains and isothermal conditions, is provided.

Consider an open-set \( \Omega \) to be the reference configuration of a deformable body in the three-dimensional Euclidean space \( \mathbb{R}^3 \). Let \( \partial \Omega \) and \( \overline{\Omega} = \Omega \cup \partial \Omega \) be the smooth boundary and the closure of \( \Omega \), respectively. The displacement field may be defined as \( u(\cdot, t) : \Omega \to \mathbb{R}^3 \) at any instant of time \( t \in \mathbb{R}^+ \). Within a small deformation setup, the strain tensor \( \epsilon \) is given by

\[
\epsilon(u) = \frac{1}{2} (\nabla u + \nabla u^T)
\]

where \( (\cdot)^T \) denotes transpose of a tensor and \( \nabla \) the gradient operator. In the PFM, sharp cracks are approximated using a diffused representation of the cracks via a length-scale parameter \( l \), and an
internal variable \( s \) called phase field (see Figure 1). Using the values of the phase-field variable, one can describe the damaged, undamaged, or partially damaged states of matter as follows: \( s = 1 \) for the undamaged state, \( s = 0 \) for the fully damaged state and \( 0 < s < 1 \) for a partially damaged state. Considering fully damaged states as the fracture, crack set may be defined as

\[
\Omega_s = f_x 2 \Omega_j (x) = 0 \quad g.
\]

Here, the damage process is considered to be irreversible that is if \( x_2 \Omega_s \) at time \( t_0 \) then \( x_2 \Omega_s \) for all \( t \geq t_0 \). Thus, the deformed and damaged states of the material body may be described by considering phase-field variable \( s \) as an additional kinematic descriptor along with the displacement vector \( u \).

To describe the deformation of a material body under external loading, one can derive the force balances from a virtual power principle. In addition to the conventional stress tensor \( \sigma \), traction vector \( t(n) \), and body force \( b \), a scalar traction \( \chi(n) \), vector stress \( \xi \), and a scalar stress \( \pi \) are introduced to define the force system. One can identify \( \dot{u} \) as the power conjugate of \( t(n) \) and \( b \), \( \dot{s} \) as the power conjugate of \( \chi(n) \) and express the external power \( P^\text{ext} \) for any arbitrary part of the body \( P \subset \Omega \) as

\[
P^\text{ext} = \int_{\partial P} [t(n) \cdot \dot{u} + \chi(n) \cdot \dot{s}] dA + \int_P b \cdot \dot{u} dV
\]

where \( \dot{()} \) denotes the time derivative of a variable, \( n \) the unit normal vector to the boundary \( \partial P \), \( dA \), and \( dV \) are, respectively, measures on \( \partial P \) and \( P \). The internal power \( P^\text{int} \) can be defined by the summation of power expenditure of \( \sigma \) over \( \nabla \dot{u} \), \( \xi \) over \( \nabla \dot{s} \) and \( \pi \) over \( \dot{s} \), and can be given by

\[
P^\text{int} = \int_P (\sigma : \nabla \dot{u} + \xi \cdot \nabla \dot{s} + \pi \dot{s}) dV
\]

Invoking the virtual power principle and the localization theorem (see Dhas et al. [16] for details), one can obtain the conventional linear momentum equation with the traction condition as given by

\[
\nabla \cdot \sigma + b = 0, \quad t(n) = \sigma n
\]

and an additional balance equation with an additional traction condition corresponding to the phase-field variable as given by

\[
\nabla \cdot \xi - \pi = 0, \quad \chi(n) = \xi \cdot n
\]

respectively.
From the second law of thermodynamics for an isothermal condition, one may find an inequality (see Dhas et al. [16] for details) given by

\[ \dot{\psi} - (\sigma : \dot{\epsilon} + \xi \cdot \nabla \dot{s} + \pi \dot{s}) \leq 0 \]  

(6)

where \( \psi \) is the Helmholtz free-energy of the system. The inequality must be satisfied while determining or postulating the constitutive relations for the thermodynamic fluxes \( \sigma, \xi, \) and \( \pi \) in terms of the kinematic quantities \( \epsilon, \nabla s, \) and \( s \). Let the free energy of the system \( \psi \) be function of \( \epsilon, \nabla s, s \) and may be written as

\[ \psi = \dot{\psi}(\epsilon, \nabla s, s) \]  

(7)

Using the chain rule in equation (7), one can get the rate of free energy as

\[ \dot{\psi} = \partial_{\epsilon} \psi : \dot{\epsilon} + \partial_{\nabla s} \psi \cdot \nabla \dot{s} + \partial_{s} \psi \dot{s} \]  

(8)

where \( \partial_{(\cdot)} \) with a suffix represents the derivative of a function with respect to the argument in the suffix while keeping others fixed. Substituting equation (8) in equation (6), one can get that

\[ (\sigma - \partial_{\epsilon} \psi) : \dot{\epsilon} + (\xi - \partial_{\nabla s} \psi) \cdot \nabla \dot{s} + (\pi - \partial_{s} \psi) \dot{s} \geq 0 \]  

(9)

Applying the Coleman–Noll procedure [37] to equation (9), one can arrive at the constitutive relations for the thermodynamics fluxes as

\[ \sigma = \partial_{\epsilon} \psi \]  

(10)

\[ \xi = \partial_{\nabla s} \psi \]  

(11)

\[ \pi = \partial_{s} \psi \]  

(12)

To quantify the thermodynamic forces, one need to specialize the constitutive relations by postulating an explicit expression of the Helmholtz free energy \( \psi \) in terms of \( \epsilon, \nabla s, \) and \( s \). Let the Helmholtz free energy \( \psi(\epsilon, \nabla s, s) \) be a sum of elastic energy \( \psi_{\text{elas}}(\epsilon, s) \) and fracture energy \( \psi_{\text{frac}}(\nabla s, s) \) as

\[ \psi(\epsilon, \nabla s, s) = \psi_{\text{elas}}(\epsilon, s) + \psi_{\text{frac}}(\nabla s, s) \]  

(13)

It is assumed that crack cannot propagate under pure compression and imposed by considering an additive decomposition of \( \epsilon \) into a volumetric part \( \epsilon_{\text{vol}} \) and a deviatoric part \( \epsilon_{\text{dev}} \), that is

\[ \epsilon = \epsilon_{\text{vol}} + \epsilon_{\text{dev}} \]  

(14)

where

\[ \epsilon_{\text{vol}} = \mathbb{P}_{\text{vol}} \epsilon; \quad \epsilon_{\text{dev}} = \mathbb{P}_{\text{dev}} \epsilon \]  

(15)

In equation (15), volumetric and deviatoric parts of a second-order tensor are obtained by introducing fourth-order projection tensors \( \mathbb{P}_{\text{vol}} \) and \( \mathbb{P}_{\text{dev}} \), respectively. Defining \( p = (1/3)I : \mathbb{P}_{\text{vol}} C \epsilon \) with \( C \) denoting the fourth-order elasticity tensor, the elastic part of the free energy may be postulated as

\[ \psi_{\text{elas}}(\epsilon) = s^2 \psi_{+}(\epsilon) + \psi_{-}(\epsilon) \]  

(16)

where \( \psi_{+}(\epsilon) \) is the elastic energy part due to a combination of pure tension and shear

\[ \psi_{+}(\epsilon) = \frac{1}{2} (p : I : \epsilon_{\text{vol}} + \mathbb{P}_{\text{dev}} C \epsilon : \epsilon_{\text{dev}}) \]  

(17)

and \( \psi_{-}(\epsilon) \) is the elastic energy part due to pure compression.
\begin{equation}
\psi_{\text{clas}}(\epsilon) = \frac{1}{2} (p) : \epsilon_{\text{vol}}
\end{equation}

In equation (16), a degradation function \( s^2 \) is introduced to account for the reduced stiffness of the material due to damage. Note that the degradation function \( s^2 \) is only associated with the so-called positive part of the elastic energy \( \psi_{\text{clas}}(\epsilon) \) to impose the condition that cracks cannot propagate under pure compression. In equation (17), \( \langle p \rangle_+ = (1/2)(p + |p|) \) and in equation (18) \( \langle p \rangle_- = (1/2)(p - |p|) \). The fracture energy \( \psi_{\text{frac}}(\nabla s, s) \) may be postulated as

\begin{equation}
\psi_{\text{frac}}(\nabla s, s) = G_c \left( \frac{(1-s)^2}{2l_s} + \frac{l_s}{2} \nabla s \cdot \nabla s \right)
\end{equation}

where \( G_c \) is the critical energy release rate and \( l_s \) is the phase field length-scale parameter. Employing equations (10), (11), (12), (17), (18), and (19), explicit expressions for the thermodynamic fluxes \( \alpha, \xi, \) and \( \pi \) may be derived as

\begin{equation}
\alpha = \partial_s \psi = s^2 (\langle p \rangle_+ I + \mathbb{P}_{\text{dev}} \epsilon) + \langle p \rangle_- I
\end{equation}

\begin{equation}
\xi = \partial_{\nabla s} \psi = G_c l_s \nabla s
\end{equation}

and

\begin{equation}
\pi = \partial_s \psi = 2s \psi_{\text{clas}}(\epsilon_{\text{clas}}) - \frac{G_c}{l_s} (1-s)
\end{equation}

respectively. Substituting the above constitutive relations in the force balances, one may express the governing PDEs in terms of the kinematic descriptors \( u \) and \( s \). One can express the stress tensor \( \alpha \) in terms of kinematic variables as \( \mathbb{C}_{\text{mod}} \epsilon \) by defining \( \mathbb{C}_{\text{mod}} \) as

\begin{equation}
\mathbb{C}_{\text{mod}} = \begin{cases} 
s^2 \mathbb{C}, & \text{for } p \geq 0 \\
s^2 (\mathbb{P}_{\text{dev}} \mathbb{C}) + \mathbb{P}_{\text{vol}} \mathbb{C}, & \text{for } p < 0
\end{cases}
\end{equation}

Using \( \alpha = \mathbb{C}_{\text{mod}} \epsilon \), equation (4) may be re-written as

\begin{equation}
\nabla \cdot (\mathbb{C}_{\text{mod}} \epsilon) + b = 0
\end{equation}

Similarly, using the expressions of \( \xi \) and \( \pi \), equation (5) may be re-written as

\begin{equation}
\nabla \cdot (G_c l_s \nabla s) - 2s \psi_{\text{clas}}(\epsilon) + \frac{G_c}{l_s} (1-s) = 0
\end{equation}

To account for the irreversibility condition on damage, a history function \( \mathcal{H}(\mathcal{E}) \), where \( \mathcal{H}(f) = \max_{\tau \in [0, f]} f(\tau) \) for any input argument \( f \), of the so-called positive part of elastic energy \( \mathcal{E} = \psi_{\text{clas}}(\epsilon) \) is employed [14]. Using the history function \( \mathcal{H}(\mathcal{E}) \), equation (25) may be expressed as

\begin{equation}
\nabla \cdot (G_c l_s \nabla s) - 2s \mathcal{H}(\mathcal{E}) + \frac{G_c}{l_s} (1-s) = 0
\end{equation}

The governing PDEs (24) and (26) are coupled and subject to boundary conditions, such as prescribed displacement \( u = \bar{u} \) and applied traction \( t(n) = t(n) \) on \( \partial \Omega_u \) and \( \partial \Omega_t \), respectively. Equations (24) and (26) together with the boundary conditions are called the strong form of the governing PDEs. In absence of body force, that is, \( b = 0 \), the strong form for phase-field modeling of brittle fracture in an elastic solid defined by domain \( \Omega \) subjected to displacement boundary condition \( u = \bar{u} \) on the boundary \( \partial \Omega_u \) can be given in a compact form as

\begin{equation}
\nabla \cdot \alpha = 0 \quad (\text{where } \alpha = \mathbb{C}_{\text{mod}} \epsilon) \quad \text{in } \Omega
\end{equation}
\[ u = \bar{u} \text{ on } \partial \Omega_u \]
\[ \sigma n = 0 \text{ on } \partial \Omega / \partial \Omega_u \]

and

\[ \nabla \cdot (G_c I_s \nabla s) - 2s \mathcal{H}(\mathcal{E}) + \frac{G_c}{I_s} (1 - s) = 0 \text{ in } \Omega \]

\[ \nabla s \cdot n = 0 \text{ on } \partial \Omega \]

In this study, considered test specimens are made of isotropic materials for which components of the fourth-order elasticity tensor may be expressed as

\[ C_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}) \]

where \( \lambda \) and \( \mu \) are the Lamé parameters and \( \delta \) denotes the Kronecker delta. The Lamé parameters are related to Young’s modulus \( E \) and Poisson’s ratio \( \nu \) by

\[ \lambda = E \nu / ((1 + \nu)(1 - 2\nu)) \text{ and } \mu = E / (2(1 + \nu)). \]

3. The weak form and an outline for finite-element implementation in Julia

In this section, first the weak form for the strong form given by equations (27) and (28) is derived and then an outline for the finite-element implementation is provided through a numerical example. Let \( v \) and \( f \) be the test functions corresponding to displacement \( u \) and phase-field \( s \), respectively. The trial spaces with the given displacement boundary condition may be given by

\[ H_u = \{ u \in H^1(\Omega)^d; \ u = \bar{u} \text{ on } \partial \Omega_u \} \]
\[ H_s = \{ s \in H^1(\Omega) \} \]

where \( \bar{u} \) is a prescribed displacement on \( \partial \Omega_u \) and \( d \) corresponds to the dimension of displacement vector. The test spaces may be defined as

\[ V_u = \{ v \in H^1(\Omega)^d; \ v = 0 \text{ on } \partial \Omega_u \} \]
\[ V_s = \{ \phi \in H^1(\Omega) \} \]

One can obtain the following weak form: find \( u \in H_u \) and \( s \in H_s \) such that for all \( v \in V_u \) and \( \phi \in V_s \)

\[ a_{\text{Disp}}(u, v) = b_{\text{Disp}}(v) \]
\[ a_{\text{PF}}(s, \phi) = b_{\text{PF}}(\phi) \]

where

\[ a_{\text{Disp}}(u, v) = \int_{\Omega} [\varepsilon(v) : \sigma(u)] d\Omega, \quad b_{\text{Disp}}(v) = 0 \]

\[ a_{\text{PF}}(s, \phi) = \int_{\Omega} \left( G_c I_s \nabla s \cdot \nabla \phi + 2s \phi \mathcal{H}(\mathcal{E}) + \frac{G_c}{I_s} s \phi \right) d\Omega, \quad b_{\text{PF}}(\phi) = \int_{\Omega} \frac{G_c}{I_s} \phi d\Omega \]

In this study, a staggered scheme originally proposed by Miehe et al. [14] is employed to solve for the unknown displacement vector and phase-field from the weak form defined by equations (33) and (34) using Gridap.

One of the salient features of Gridap is that one can directly use the weak form in Julia for the finite-element implementation using Gridap. All the steps associated with the finite-element simulations such as creating the mesh file, implementing the weak form, application of boundary conditions, solutions for
the unknown field variables, and post-processing for the output files are described through a Julia code on numerical simulation of a test on a notched beam under symmetric three-point bending (see Section 4.4 for numerical results). One may readily apply the developed phase-field-based Julia code for simulating other brittle fracture problems with appropriate modifications. For reproducing the results presented in section 4, one needs to first load the following Julia packages in the script file which is presently written in a jupyter notebook.

```julia
using GridapGmsh
using Gridap
using Gridap.Geometry
using Gridap.TensorValues
using PyPlot
```

One can define the input parameters associated with the elastic and fracture material properties of the notched beam by writing the following lines.

```julia
const E_mat = 20.8
const ν_mat = 0.3
const Gc = 5e-4
const ls = 0.03
const η = 1e-15
```

For the finite-element simulations, one needs to have the discretization of the computational domain which can be generated by writing a mesh file in Julia (see Julia code for meshing of notched beam under symmetric three-point bending) and load that mesh file to build an instance of “DiscreteModel” by the following lines.

```julia
model = GmshDiscreteModel("BeamWithNotchSymThreePtBending.msh")
writevtk(model, "BeamWithNotchSymThreePtBending")
```

For an isotropic material, the constitutive tensor \( C \) can be defined for two-dimensional plane stress and plane strain problems by writing the following function.

```julia
function ElasFourthOrderConstTensor(\( E \), \( ν \), PlanarState)
    # 1 for Plane Stress and 2 Plane Strain Condition
    if PlanarState == 1
        C1111 = \( E / (1 - ν^2) \)
        C1122 = \( (ν^2 E) / (1 - ν^2) \)
        C1112 = 0.0
        C2222 = \( E / (1 - ν^2) \)
        C2212 = 0.0
        C1212 = \( E / (2*(1 + ν)) \)
    elseif PlanarState == 2
        C1111 = \( (E*(1 - ν*ν)) / ((1 + ν) - (1 - ν - 2*ν*ν)) \)
        C1122 = \( (ν^2 E) / (1 - ν - 2*ν*ν) \)
        C1112 = 0.0
        C2222 = \( (E*(1 - ν)) / (1 - ν^2) \)
        C2212 = 0.0
        C1212 = \( E / (2*(1 + ν)) \)
    end
    C_ten = SymFourthOrderTensorValue(C1111, C1112, C1122, C1112,
                                       C1212, C2212, C1122, C2212, C2222)
    return C_ten
end
```
In this study, plane strain condition is assumed for which the constitutive tensor is computed by calling
the above Julia function as given below.

```julia
const C_mat = ElasFourthOrderConstTensor(E_mat, ν_mat, 2)
```

To satisfy the assumption that crack cannot propagate under pure compression, stress, and strain ten-
sors are decomposed into a volumetric and a deviatoric part by introducing the projection operators $P_{\text{vol}}$ and $P_{\text{dev}}$, respectively, which are defined by the following lines.

```julia
I2 = SymTensorValue{2, Float64}(1.0, 0.0, 1.0)
I4 = I2 ⊗ I2
I4_sym = one(SymFourthOrderTensorValue{2, Float64})
P_{\text{vol}} = (1.0/3) * I4
P_{\text{dev}} = I4_sym - P_{\text{vol}}
```

To express the stress tensor in terms of kinematics variables, that is, $\sigma = C_{\text{mod}} \epsilon$, where the expression of $C_{\text{mod}}$ is given by equation (23), the following function is defined in Julia.

```julia
function σ fun(ε, ε_in, s_in)
    σ_elas = C_mat ⊙ ε
    if tr(ε_in) ≥ 0
        σ = (s_in ^ 2 + η) * σ_elas
    elseif tr(ε_in) < 0
        σ = (s_in ^ 2 + η) * P_dev ⊙ σ_elas + P_vol ⊙ σ_elas
    end
    return σ
end
```

One can determine the so-called positive part of elastic free energy, $\psi_{+}^{\text{elas}}(\epsilon)$ which is given by equation (17), by writing the following function in Julia.

```julia
function ψ Pos(ε_in)
    σ_elas = C_mat ⊙ ε_in
    if tr(ε_in) ≥ 0
        ψ_Pos = 0.5 * (ε_in ⊙ σ_elas)
    elseif tr(ε_in) < 0
        ψ_Pos = 0.5 * ((P_dev ⊙ σ_elas) ⊙ (P_dev ⊙ ε_in))
    end
    return ψ_Pos
end
```

One needs to generate a discrete approximation of the finite-element test and trial spaces of the problem
on the discretized computational domain. Approximation of the finite-element spaces associated with
the phase-field variable can be done by the following lines in Julia.

```julia
order = 1
reffe_PF = ReferenceFE(lagrangian, Float64, order)
V0_PF = TestFESpace(model, reffe_PF;
    conformity =: H1)
U_PF = TrialFESpace(V0_PF)
sh = zero(V0_PF)
```

Similarly, one can generate the approximation of the finite-element spaces associated with the displace-
ment variable by writing the following lines in Julia.
To compute the integrals in the weak form given by equation (34) numerically, one needs to define an integration mesh along with an integration rule (for example, Gauss quadrature) in each of the cells in the triangulation. Using Gridap, one can easily define the integration mesh and the corresponding Lebesgue measure by using the built-in functions “Triangulation” and “Measure,” respectively. For instance, one can use the following lines for integrating the weak form given by equation (34) defined on the domain $\Omega$ using a quadrature rule of degree two times the order of interpolation in the cells of the triangulation.

```plaintext
degree = 2 * order
$\Omega = \text{Triangulation}(\text{model})$
d $\Omega = \text{Measure}(\Omega, \text{degree})$
```

One can determine the applied load on a part of the boundary of the domain by determining boundary integral using the following built-in functions available in Gridap.

```plaintext
labels = \text{get\_face\_labeling}(\text{model})
LoadTagId = \text{get\_tag\_from\_name}(labels, ’’LoadLine’’)
$\Gamma_{Load} = \text{BoundaryTriangulation}(\text{model}, \text{tags} = \text{LoadTagId})$
d $\Gamma_{Load} = \text{Measure}(\Gamma_{Load}, \text{degree})$
n $\Gamma_{Load} = \text{get\_normal\_vector}(\Gamma_{Load})$
```

To find the values of variables that are defined by using the built-in function “CellState” at the Gauss points, one need to use the “project” function as defined below.

```plaintext
function project(q, model, d $\Omega$, order)
    reffe = \text{ReferenceFE}(\text{lagrangian}, \text{Float64}, \text{order})
    V = \text{FESpace}(\text{model}, \text{reffe}, \text{conformity} = :L2)
    a(u, v) = \int (u*v) * d $\Omega$
    b(v) = \int (v*q) * d $\Omega$
    op = \text{AffineFEOperator}(a, b, V, V)
    qh = \text{solve}(op)
    return qh
end
```

In this study, a staggered scheme is used to update the solution from the pseudo time $t_n$ to $t_{n+1}$ [14]. Given the displacement vector, phase-field and the history function at the time $t_n$, one can update the phase field at the time $t_{n+1}$ by using the following function.

```plaintext
function stepPhaseField(uh_in, $\Psi$ PlusPrev_in)
a_PF(s, $\phi$) = \int (Gc*ls*\nabla(\phi) \cdot \nabla(s) + 2*\Psi \text{PlusPrev\_in}*s*\phi + (Gc/ls)*s*\phi) * d $\Omega$
b_PF(\phi) = \int ((Gc/ls)*\phi) * d $\Omega$
op_PF = \text{AffineFEOperator}(a_PF, b_PF, U_PF, V0_PF)
sh_out = \text{solve}(op_PF)
return sh_out
end
```
Using the values of displacement vector and the history function at time \( t_n \), and the computed value of phase-field at time \( t_{n+1} \), one can update the displacement vector at time \( t_{n+1} \) by calling a function as given below.

\[
\text{function stepDisp(uh_in, sh_in, vApp)}
\begin{align*}
    uApp1(x) &= \text{VectorValue}(0.0, 0.0) \\
    uApp2(x) &= \text{VectorValue}(0.0, 0.0) \\
    uApp3(x) &= \text{VectorValue}(0.0, -vApp) \\
    U_{\text{Disp}} &= \text{TrialFESpace}(V0_{\text{Disp}}, (uApp1, uApp2, uApp3)) \\
    a_{\text{Disp}}(u, v) &= \int \left( (\varepsilon(v)) \cdot (\sigma \text{fun}^{\delta}(\varepsilon(u)), \varepsilon(u)) \right) d\Omega \\
    b_{\text{Disp}}(v) &= 0.0 \\
    \text{op}_{\text{Disp}} &= \text{AffineFEOperator}(a_{\text{Disp}}, b_{\text{Disp}}, U_{\text{Disp}}, V0_{\text{Disp}}) \\
    uh_{\text{out}} &= \text{solve}(\text{op}_{\text{Disp}}) \\
    \text{return } uh_{\text{out}}
\end{align*}
\]

Once both displacement and phase field are determined at time \( t_{n+1} \), one can update the energy history function at time \( t_{n+1} \) by defining it as

\[
\mathcal{H}_{n+1} = \begin{cases} 
\varepsilon_{n+1} & \text{for } \varepsilon_{n+1} > \varepsilon_n \\
\varepsilon_n & \text{otherwise,}
\end{cases}
\tag{35}
\]

where the history function \( \mathcal{H}(\varepsilon) \) of elastic free energy \( \varepsilon = \psi_{\text{elas}}^{\text{cl}}(\varepsilon) \) needs to be defined in Julia (see Section 2), which can be achieved by writing the following lines.

\[
\text{function new_EnergyState(\psi PlusPrev_in, \psi hPos_in)}
\begin{align*}
    \psi \text{ Plus_in} &= \psi \text{ hPos_in} \\
    \text{if } \psi \text{ Plus_in} &> = \psi \text{ PlusPrev_in} \\
        \psi \text{ Plus_out} &= \psi \text{ Plus_in} \\
    \text{else} \\
        \psi \text{ Plus_out} &= \psi \text{ PlusPrev_in} \\
    \text{end} \\
    \text{true, } \psi \text{ Plus_out} \\
\end{align*}
\]

Finally, one can simulate the brittle fracture in the notched beam by applying a monotonic displacement control loading and solve for the unknown displacement and phase field at each loading step by writing the main routine in Julia that uses the above-defined functions as listed below.

\[
\begin{align*}
    \text{vApp} &= 0 \\
    \text{delv} &= 1e-3 \\
    \text{const vAppMax} &= 0.1 \\
    \text{innerMax} &= 10 \\
    \text{count} &= 0 \\
    \text{Load} &= \text{Float64[]} \\
    \text{Displacement} &= \text{Float64[]} \\
    \text{push!(Load, 0.0)} \\
    \text{push!(Displacement, 0.0)} \\
    \text{sPrev} &= \text{CellState}(1.0, d\Omega) \\
    \text{sh} &= \text{project(sPrev, model, d\Omega, order)} \\
    \psi \text{ PlusPrev} &= \text{CellState}(0.0, d\Omega) \\
    \text{while vApp < vAppMax} \\
        \text{count} &= \text{count} + 1 \\
        \text{if vApp} &> = 3e-2
\end{align*}
\]
delv = 1e-4
end
vApp = vApp + delv
print(' Entering displacement step:', float(vApp))
for inner = 1: innerMax
    ψhPlusPrev = project (ψ PlusPrev, model, d Ω, order)
    RelErr = abs (\( \sum (Gc/ls * \nabla \cdot \nabla (sh) + 2*ψ hPlusPrev * sh*sh + (Gc/ls)*sh*sh)*d Ω - \int (Gc/ls)*sh*sh)*d Ω) / abs (\( \sum (Gc/ls)*sh)*d Ω) )
    sh = stepPhaseField(uh, ψ hPlusPrev)
    uh = stepDisp (uh, sh, vApp)
    ψ hPos in = ψ Pos°(ε(uh))
    update_state! (new_EnergyState, ψ PlusPrev, ψ hPos in)
    if RelErr < 1e-8
        break
    end
end
Node_Force = sum (\( n_\Gamma_Load \cdot (\sigma fun°(\varepsilon(uh), \varepsilon(uh), sh))) * d \Gamma_{Load})
push! (Load, -Node_Force (2))
push! (Displacement, vApp)
end

One can create output files at each loading step that can be viewed in ParaView. For instance, one can create a “.vtu” file to save data for the solution of the displacement vector and phase field at each loading step by using the following lines in Julia.

```julia
writevtk (Ω, 'results_SymThreePtBendingTest',
          cellfields = [ "uh" => uh, "s" => sh])
```

One can generate the load–displacement curve by using the plot command as given below.

```julia
plot (Displacement, Load)
```

### 4. Numerical simulations

Proposed phase-field-based Julia codes are validated against tests on a single-edge notched plate under tension, an L-shaped panel, a notched plate with a hole, a notched beam under symmetric three-point bending, and a notched beam with three holes under asymmetric three-point bending. The effect of length-scale parameter value on the fracture response is well understood in the literature. However, for completeness of this article, the effect of length-scale parameter on the fracture response is demonstrated through simulations of brittle fracture in a single-edge notched plate subjected to tension for different values of \( l_s \). The width of a diffused crack increases with the increase in the value of the length-scale parameter, and thus a higher value of \( l_s \) may not give a reasonable approximation of the original sharp crack problems. One can easily show the same observation for all the other numerical examples and hence not presented here. Instead, the results obtained using the proposed Julia implementation are directly compared for one particular value of the length-scale parameter \( l_s \) which is mentioned in the literature for the given problem. In the same spirit, the convergence of results to mesh refinement is only shown for the single-edge notched plate under tension. For all the numerical simulations, non-uniform
finite-element meshes with a finer mesh (length of the largest side of triangular elements is less than half of \( l_s \) value) in regions where cracks may propagate is used.

4.1. Single-edge notched plate under tension

Modeling of brittle fracture in a single-edge notched plate subjected to tension using a PFM has been considered by several studies in the literature [7,14,38,39]. The geometry of the specimen and the boundary conditions considered for the test is shown in Figure 2(a). For simulations of brittle fracture in the single-edge notched plate, a finite-element mesh file “SquarePlateWithEdgeNotch.msh” is generated by writing a Julia code (see Julia code for meshing of single-edge notched plate under tension). Using the mesh file “SquarePlateWithEdgeNotch.msh” with the required modifications in the Julia implementation outlined in Section 3, brittle fracture in the notched plate is simulated (see Julia code for fracture simulation in a single-edge notched plate under tension). A typical finite-element mesh used for the simulation is shown in Figure 2(b). Material properties for the notched plate are taken as \( E = 210 \times 10^3 \) MPa, \( \nu = 0.3 \), and \( G_c = 2.7 \) N/mm. Displacement-controlled loading is considered with an increment of \( \Delta u_2 = 1 \times 10^{-4} \) mm up to \( 5 \times 10^{-3} \) mm and \( \Delta u_2 = 1 \times 10^{-5} \) mm till failure of the specimen. Crack propagation in the specimen at three different load steps is shown in Figure 3. From the notch-tip, crack propagates straight to the right edge of the specimen. Convergence of the finite-element solution is shown through load–displacement curves for three different meshing in Figure 4. To show the effect of length-scale parameter value on the response, three different values of \( l_s \) are considered. As can be seen from Figure 5, diffused crack is observed to be lesser diffused (that represents the sharp crack in a better way) with the decrease in the value of \( l_s \). From the load–displacement curve shown in Figure 6, one can observe that the fracture load increases with the decrease in the value of \( l_s \). For a particular length-scale value \( l_s = 0.0075 \) mm, the load–displacement curve (see Figure 7) obtained from the proposed Julia implementation shows an excellent agreement with the results reported in the literature [14].

4.2. L-shaped panel

Modeling of mixed-mode brittle fracture in an L-shaped panel is considered as one of the benchmark problems and has been solved by using PFM [7,40] and using screened Poisson equation [41,42]. For
numerical simulation, the setup for the test on the L-shaped panel is shown in Figure 8(a). For brittle fracture simulations in the L-shaped panel, a finite-element mesh file “LShapedPanel.msh” is generated by writing a Julia code (see Julia code for meshing of L-shaped panel). Using the mesh file “LShapedPanel.msh” with the necessary modifications in the Julia implementation outlined in Section 3, simulation of brittle fracture in the L-shaped panel is accomplished (see Julia code for simulation of fracture in an L-shaped panel). Non-uniform finite-element mesh with finer mesh where crack may propagate, as shown in Figure 8(b), is used for the simulation. The length scale considered for the problem is $l_s = 1.1875$ mm as mentioned by Ambati et al. [7]. Material properties for the problem are taken as $E = 25.8423 \times 10^3$ MPa, $\nu = 0.18$, and $G_c = 0.089$ N/mm. Displacement control loading with an increment of $\Delta \bar{u}_2 = 10^{-3}$ mm has been considered. The applied displacement is increased to 0.3 mm gradually, then unloaded till the origin, and then again loaded till failure. As can be seen from Figure 9, the load–deformation curve obtained from the proposed Julia implementation is in good agreement with the results reported in the literature [7]. Simulated crack patterns at different load steps are demonstrated in Figure 10 which are also very similar to the results reported in the literature [7,41,42].

Figure 3. Damage profiles for a single-edge notched plate under tension for a characteristics length scale $l_s = 0.0075$ mm at applied displacement (a) 0.00549 mm, (b) 0.00589 mm, and (c) 0.0062 mm.

Figure 4. Convergence of finite-element solution for a single-edge notched plate under tension to mesh-refinement. In figure, three load–displacement curves for three different meshing is shown, where $N_n$ and $N_e$ denotes number of nodes and number of elements, respectively.
Figure 5. Damage profiles for a single-edge notched plate under tension at an applied displacement of 0.062 mm and length-scale value of (a) $l_s = 0.030 \text{ mm}$, (b) $l_s = 0.015 \text{ mm}$, and (c) $l_s = 0.0075 \text{ mm}$.

Figure 6. Load–displacement curves for a single-edge notched plate under tension for different characteristic length scale, $l_s$ values.

Figure 7. Comparison of load–displacement curve (for same length-scale value $l_s = 0.0075 \text{ mm}$) obtained from the proposed Julia implementation with the results reported by Miehe et al. [14] for a single-edge notched plate subjected to tension.
4.3. Notched plate with a hole

Modeling of brittle fracture in a notched plate with a hole at an offset from the central axis is studied in the literature by using fenics implementation of PFM [29] and strain gradient continuum damage model [43]. The geometry of the specimen and the boundary conditions considered for the test are shown in Figure 11(a). The characteristic length-scale value is chosen as $l_s = 0.45\text{mm}$ as specified by Hirshikesh et al. [29]. For brittle fracture simulations in the notched plate with a hole using the proposed Julia implementation, a finite-element mesh file “NotchedPlateWithHole.msh” is generated by writing a Julia code (see Julia code for meshing of the notched plate with hole). Using the mesh file “NotchedPlateWithHole.msh” and the associated modifications in the Julia implementation outlined in Section 3, the brittle fracture in the notched plate with a hole is simulated (see Julia code for simulation of fracture in a notched plate with a hole). Non-uniform finite-element mesh with finer mesh near the notch and hole where crack may propagate, as shown in Figure 11(b), is used for simulation. The material properties of the notched plate with hole are chosen as: $E = 5.9827 \times 10^3\text{MPa}$, $\nu = 0.22$, and
\[ G_c = 2.28 \text{ N/mm} \]. Displacement-controlled loading is considered with an increment of \( \Delta \bar{u}_2 = 1 \times 10^{-2} \text{ mm} \) up to 0.38 mm and \( \Delta \bar{u}_2 = 5 \times 10^{-4} \text{ mm} \) till failure of the specimen. Crack profiles at two different applied displacement are shown in Figure 12. Crack propagation starts from the notch and traverses toward the hole. After the failure of a section near the notch, the load again started increasing and developed a secondary crack that propagates from the hole to the right-side edge of the specimen. As can be seen from Figure 13, the load–displacement curve obtained from the proposed Julia implementation matches well with the available results in the literature [29].

### 4.4. Symmetric three-point bending test

Modeling of brittle fracture in a simply supported notched beam under symmetric three-point bending is one of the classical benchmark problems which has frequently been analyzed in the literature.

---

**Figure 10.** Damage profiles for the L-shaped panel at (a) 300th load step, (b) 900th load step, and (c) 1300th load step.

**Figure 11.** Test setup for a notched plate with a hole under tension and a finite-element mesh for the geometry of the notched plate with a hole (all dimensions are in millimeters (mm)). Sub-figure (a) shows the geometry and boundary conditions. Sub-figure (b) shows a representative mesh using triangular elements for the finite-element simulation.
All the steps associated with the simulations of brittle fracture in the notched beam are outlined in Section 3, which is also available as a jupyter notebook file in Julia code for simulation of fracture in a notched beam under symmetric bending. The three-point bending test setup and a finite-element mesh used for the simulation are demonstrated in Figure 14. For the numerical simulation, material properties for the notched beam are taken as \( E = 20.8 \times 10^3 \) MPa, \( \nu = 0.3 \), \( G_c = 5.0 \times 10^{-3} \) kN/mm, and \( l_s = 0.03 \) mm. Displacement control loading (monotonic displacement \( \Delta u_2 \)) is applied in small increments \( \Delta u_2 \) is considered and the damage profiles for the notched beam at different stages of applied displacement are presented in Figure 15. For applied displacement up to \( u_2 = 0.025 \) mm, monotonic increment of \( \Delta u_2 = 10^{-3} \) mm and from \( u_2 = 0.025 \) mm to until failure

**Figure 12.** Damage profiles for a notched plate with a hole (at an offset from the central axis) under tensile loading at different deformation stages are shown in sub-figures at applied displacement (a) 0.438 mm and (b) 0.56 mm.

**Figure 13.** Comparison of the load–displacement curve for the notched plate with a hole obtained from the proposed Julia implementation with the results reported by Hirshikesh et al. [29].
Figure 14. Three-point bending test setup and a finite-element mesh for the geometry of the notched beam (all dimensions are in millimeters (mm)). Sub-figure (a) shows the geometry and boundary conditions for the test. Sub-figure (b) shows the mesh using triangular elements for the finite-element simulation.

Figure 15. Damage profiles for a notched beam under symmetric three-point bending at different deformation stages shown in sub-figures at applied displacement (a) 0.0405 mm, (b) 0.041 mm, (c) 0.05 mm, and (d) 0.1 mm.

Figure 16. Comparison of load–displacement curve obtained from the proposed Julia implementation for a symmetric three-point bending test with the results reported by Miehe et al. [14].
4.5. Asymmetric notched three-point bending test

The developed Julia code for the PFM is validated against a set of tests on a notched beam with three holes under asymmetric three-point bending, which were carried out by Ingraffea and Grigoriu [44] and numerically analyzed in Bittencourt et al. [45]. The geometry and the boundary conditions for the test is shown in Figure 17(a). Material parameters are taken as $E = 4.75 \times 10^5$ psi, $\nu = 0.35$, $G_c = 1.8$ lbf/in, and $l_s = 0.01$ inch. To verify whether the proposed Julia implementation of PFM can predict experimentally observed complex crack patterns, three different configurations of the specimen characterized by the values $e_1$ and $e_2$ are considered. For each configuration, one can generate a finite-element mesh file by writing a Julia code (see for example Julia code for meshing of notched beam with holes) and using the generated mesh file with the corresponding modifications in the Julia implementation outlined in Section 3, one can simulate brittle fracture in the notched beam with holes under asymmetric bending (see for example, Julia code for simulation of fracture in a notched beam with holes under asymmetric three-point bending). Non-uniform finite-element mesh with finer mesh where crack may propagate, as shown in Figure 17(b), is used for simulations. Prediction of crack path for the beam with three holes and a pre-notch defined by (a) $e_1 = 6$ inch and $e_2 = 1$ inch, (b) $e_1 = 5$ inch and $e_2 = 1.5$ inch and (c) $e_1 = 4.75$ inch and $e_2 = 1.5$ inch are considered. Displacement control loading (monotonic displacement $\bar{u}_2$ is applied in small increments $\Delta \bar{u}_2$) is considered and the damage profiles for the beam with three holes and a pre-notch defined by (a) $e_1 = 6$ inch and $e_2 = 1$ inch, (b) $e_1 = 5$ inch and $e_2 = 1.5$ inch and (c) $e_1 = 4.75$ inch and $e_2 = 1.5$ inch at different stages of applied displacement are presented in Figures 18–20, respectively. As can be seen from Figures 21–23, the proposed Julia implementation shows a very good prediction of the experimentally observed crack paths which are very sensitive to the height and relative location of the pre-notch. Remarkably, the proposed implementation reproduces the intricate deviation of crack path due to the local stress concentration around the bottom hole which is experimentally observed (see Figure 22).

5. Concluding remarks

This study has provided a novel numerical implementation of a thermodynamically consistent PFM for brittle fracture using an open-source finite-element toolbox, Gridap in Julia. The proposed implementation is validated against a few numerical and experimental results available in the literature. As the
Figure 18. Damage profiles for a beam with three holes and a pre-notch defined by $e_1 = 6$ inch and $e_2 = 1$ inch under asymmetric three-point bending at different deformation stages shown in sub-figures at applied displacement (a) 0.055 inch, (b) 0.0585 inch, (c) 0.061 inch, and (d) 0.0625 inch.

Figure 19. Damage profiles for a beam with three holes and a pre-notch defined by $e_1 = 5$ inch and $e_2 = 1.5$ inch under asymmetric three-point bending at different deformation stages shown in sub-figures at applied displacement (a) 0.034 inch, (b) 0.035 inch, (c) 0.036 inch, and (d) 0.0369 inch.
The proposed implementation is available with an open-source license, it may eliminate the technical barrier for practitioners and researchers who are interested to explore the PFM for solving a wide range of brittle fracture problems. Moreover, the proposed implementation will expose the users to many built-in packages of Julia that may be useful for researchers who want to extend the proposed implementation for the case of ductile fracture or other applications. Most importantly, the availability of an open-source license allows for broader adoption and collaboration in the scientific community.

**Figure 20.** Damage profiles for a beam with three holes and a pre-notch defined by $e_1 = 4.75$ inch and $e_2 = 1.5$ inch under asymmetric three-point bending at different deformation stages shown in sub-figures at applied displacement (a) 0.035 inch, (b) 0.0375 inch, (c) 0.039 inch, and (d) 0.04 inch.

**Figure 21.** Comparison of the crack paths for the beam with three holes and a pre-notch defined by $e_1 = 6$ inch and $e_2 = 1$ inch under asymmetric three-point bending. Sub-figures (a) and (b) show crack paths for the experimentally observed [44] and the numerically predicted, respectively.
source code that is compact, user-friendly, highly efficient, and accessible to everyone will allow a third-party verification and essentially establish a high standard for efficient open-source code development.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The author of this article gratefully acknowledge support from the Indian Institute of Technology Bhubaneswar under the grant SP107.

ORCID iD
Mohammad Masiur Rahaman https://orcid.org/0000-0001-5568-8963

Figure 22. Comparison of the crack paths for the beam with three holes and a pre-notch defined by $e_1 = 5$ inch and $e_2 = 1.5$ inch under asymmetric three-point bending. Sub-figures (a) and (b) show crack paths for the experimentally observed [44] and the numerically predicted, respectively.

Figure 23. Comparison of the crack paths for the beam with three holes and a pre-notch defined by $e_1 = 4.75$ inch and $e_2 = 1.5$ inch under asymmetric three-point bending. Sub-figures (a) and (b) show crack paths for the experimentally observed [44] and the numerically predicted, respectively.
Data accessibility
The present work does not generate any experimental data. Source codes for the proposed Julia implementation of a phase-field model can be downloaded as jupyter notebook files from Julia codes for modeling brittle fracture using phase-field model.

References
[1] Griffitli, A. The phenomena of rupture and flow in solids. Philos Trans R Soc London A 1920; 221: 163–198.
[2] Irwin, GR. Onset of fast crack propagation in high strength steel and aluminum alloys. Technical report, Naval Research Lab, Washington, DC, 24 May 1956.
[3] Williams, J. Introduction to linear elastic fracture mechanics. Eur Struct Integr Soc 2001; 28: 3–10.
[4] Luo, Y, Xie, H, Ren, L, et al. Linear elastic fracture mechanics characterization of an anisotropic shale. Sci Rep 2018; 8: 8505.
[5] Barenblatt, GI. The mathematical theory of equilibrium cracks in brittle fracture. Adv Appl Mech 1962; 7: 55–129.
[6] Dugdale, DS. Yielding of steel sheets containing slits. J Mech Phys Solids 1960; 8: 100–104.
[7] Ambati, M, Gerasimov, T, and De Lorenzis, L. A review on phase-field models of brittle fracture and a new fast hybrid formulation. Comput Mech 2015; 55: 383–405.
[8] Francfort, GA, and Marigo, J-J. Revisiting brittle fracture as an energy minimization problem. J Mech Phys Solids 1998; 46: 1319–1342.
[9] Bournet, B, Francfort, GA, and Marigo, J-J. Numerical experiments in revisited brittle fracture. J Mech Phys Solids 2000; 48: 797–826.
[10] Bournet, B, Francfort, GA, and Marigo, J-J. The variational approach to fracture. J Elasticity 2008; 91: 5–148.
[11] Kuhn, C, and Müller, R. A phase field model for fracture. Proc Appl Math Mech 2008; 8: 10223–10224.
[12] Amor, H, Marigo, J-J, and Maurini, C. Regularized formulation of the variational brittle fracture with unilateral contact: numerical experiments. J Mech Phys Solids 2009; 57: 1209–1229.
[13] Kuhn, C, and Müller, R. A continuum phase field model for fracture. Eng Fract Mech 2010; 77: 3625–3634.
[14] Borden, MJ, Hughes, TJ, Landis, CM, et al. A higher-order phase-field model for brittle fracture: formulation and analysis within the isogeometric analysis framework. Comput Methods Appl Mech Eng 2014; 273: 100–118.
[15] Dhas, B, Rahaman, M, Akella, K, et al. A phase-field damage model for orthotropic materials and delamination in composites. J Appl Mech 2018; 85: 011010.
[16] Aranson, I, Kalatsky, V, and Vinokur, V. Continuum field description of crack propagation. Phys Rev Lett 2000; 85: 118.
[17] Elgibali, A, Kessler, DA, and Levine, H. Phase-field model of mode III dynamic fracture. Phys Rev Lett 2001; 87: 045501.
[18] Hakim, V, and Elgibali, A. Laws of crack motion and phase-field models of fracture. J Mech Phys Solids 2009; 57: 342–368.
[19] Borden, MJ, Hughes, TJ, Landis, CM, et al. Abaqus implementation of phase-field model for brittle fracture. Finite Elem Anal Des 2017; 130: 27–38.
[20] Wu, J-Y, and Huang, Y. Comprehensive implementations of phase-field damage models in Abaqus. Theor Appl Fract Mec 2020; 106: 102440.
[31] Goswami, S, Anitescu, C, Chakraborty, S, et al. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theor Appl Fract Mec* 2020; 106: 102447.

[32] Bezanson, J, Karpinski, S, Shah, VB, et al. Julia: a fast dynamic language for technical computing. *arXiv preprint arXiv* 2012; 1209.5145, https://arxiv.org/pdf/1209.5145.pdf

[33] Bezanson, J, Edelman, A, Karpinski, S, et al. Julia: a fresh approach to numerical computing. *SIAM Rev* 2017; 59: 65–98.

[34] Badia, S, and Verdugo, F. Gridap: an extensible finite element toolbox in Julia. *J Open Source Softw* 2020; 5: 2520.

[35] Verdugo, F, and Badia, S. A user-guide to Gridap — grid-based approximation of partial differential equations in Julia. *arXiv preprint arXiv* 2019: 1910.01412, https://arxiv.org/pdf/1910.01412.pdf

[36] Frondelius, T, and Aho, J. JuliaFEM-open source solver for both industrial and academia usage. *Rakenteiden Mekaniikka* 2017; 50: 229–233.

[37] Coleman, BD, and Noll, W. The thermodynamics of elastic materials with heat conduction and viscosity. In: Truesdell, C, and Noll, W (eds) *The foundations of mechanics and thermodynamics*. New York: Springer, 1974, pp. 145–156.

[38] Miehe, C, and Gürses, E. A robust algorithm for configurational-force-driven brittle crack propagation with R-adaptive mesh alignment. *Int J Numer Meth Eng* 2007; 72: 127–155.

[39] Wu, J-Y, and Nguyen, VP. A length scale insensitive phase-field damage model for brittle fracture. *J Mech Phys Solids* 2018; 119: 20–42.

[40] Li, H, Wang, W, Cao, Y, et al. Phase-field modeling fracture in anisotropic materials. *Adv Civil Eng* 2021; 2021: 4313755.

[41] Areias, P, Msekh, M, and Rabczuk, T. Damage and fracture algorithm using the screened Poisson equation and local remeshing. *Eng Fract Mech* 2016; 158: 116–143.

[42] Areias, P, Reinoso, J, Camanho, P, et al. Effective 2d and 3d crack propagation with local mesh refinement and the screened Poisson equation. *Eng Fract Mech* 2018; 189: 339–360.

[43] Barchiesi, E, Yang, H, Tran, C, et al. Computation of brittle fracture propagation in strain gradient materials by the FEniCS library. *Math Mech Solids* 2021; 26: 325–340.

[44] Ingraffea, AR, and Grigoriu, M. Probabilistic fracture mechanics: a validation of predictive capability. *Technical report, Department of Structural Engineering, Cornell University, Ithaca, NY*, August 1990.

[45] Bittencourt, T, Wawrzynek, P, Ingraffea, A, et al. Quasi-automatic simulation of crack propagation for 2d LEFM problems. *Eng Fract Mech* 1996; 55: 321–334.