Proceedings of the Canadian Society for Mechanical Engineering International Congress
32nd Annual Conference of the Computational Fluid Dynamics Society of Canada
Canadian Society of Rheology Symposium
CSME/CFDSC/CSR 2025
May 25–28, 2025, Montréal, Québec, Canada

NUMBER

PARALLELIZED AND ADAPTIVELY-REFINED PHASE-FIELD MONOLITHIC SCHEME FOR BRITTLE CRACK SIMULATION

Zhao Li[1], Kuiying Chen[2], Tao Jin[1*]

[1]Department of Mechanical Engineering, University of Ottawa, Ottawa, Canada
[2]Aerospace Research Centre, National Research Council Canada, Ottawa, Canada
[*]tao.jin@uottawa.ca

***Abstract*—**The phase-field method has become a widely adopted technique in fracture propagation simulations because it can naturally handle complex crack path. However, it is computationally expensive, which hinders its application in three-dimensional (3D) problems. To overcome this challenge, we adopt the parallel computing paradigm based on the Message Passing Interface (MPI) and the adaptive mesh refinement technique. Specifically, we employ the MPI to parallelize the phase-field monolithic scheme based on the limited-memory BFGS (L-BFGS) method, which reduces the memory requirement by storing a series of vectors instead of a fully dense matrix. Additionally, the MPI enables efficient communications between processors across multiple computing nodes in a distributed system, making it ideal for large-scale simulations. A fully distributed mesh is selected so that only a non-overlapping grid is stored on each processor, which accelerates the computational process by solving multiple smaller tasks simultaneously. Furthermore, the adaptive refinement technique is also applied such that only elements near the cracked region are refined, which could reduce the size of the discretized problem. We present both 2D and 3D crack propagation problems to evaluate the effectiveness of the parallelization and adaptive mesh refinement technique in reducing the computational cost. We also perform comprehensive scaling analyses using the high-performance computing (HPC) resources provided by the Digital Research Alliance of Canada.

***Keywords-component*—phase-field; L-BFGS scheme; adaptive mesh refinement; crack propagation; MPI; HPC**

I. INTRODUCTION

The phase-field method is widely adopted to simulate crack propagation without relying on any heuristic crack tracking strategy or remeshing to accommodate crack growth. This method regularizes the crack using the phase-field in a diffusive manner. A diffusive band, represented by the phase-field, separates the crack from the undamaged parts of the material, avoiding difficulties caused by the discontinuity due to the cracks [1]. However, the expensive computational cost, due to the extremely fine mesh that is used to resolve the typically small length-scale for regularizing the crack surface [14], hinders the application in 3D problems [2]. Three approaches to alleviate the computational cost have been developed: adaptive mesh refinement technique to reduce the number of degrees of freedom (DoFs) by only refining the mesh near the crack region, more efficient solvers for faster convergence, and the parallelization technique to speed up the simulations via a fully distributed grid strategy.

For large-scale phase-field problems, parallelization is a useful tool to ensure that simulations can be finished within a reasonable period of time [3]. Some parallelized phase-field models in various forms have been developed. Liu et al. applied a thread-parallel execution with a shared memory via a commerical finite element software Abaqus to compare the quasi-static and dynamic fracture results computed by the XFEM method and the phase-field method based on both monolithic and staggered schemes and achieved a good agreement [4]. Ishii et al. utilized a thread-parallelized technique using a multi-color ordering method to solve a phase-field model for crack propagation with the conjugate gradient method and the incomplete Cholesky preconditioner [9]. Heister and Wick implemented a fully-parallelized open-source framework called *pfm-cracks* for phase-field fracture problems based on a quasi-monolithic formation solved by the semi-smooth Newton algorithm [5]. Ziaei-Rad and Shen employed graphics processing units (GPUs) to accelerate the phase-field formation to solve for the fracture on the brittle materials [6]. Hao and Shen proposed an explicit-implicit scheme for the phase-field model based on the non-overlapping domain decomposition method to solve dynamic fracture problems in

* corresponding author

a parallel manner [7]. Badri et al. applied a parallelized monolithic phase-field model to compare different preconditioners to improve convergence for large-scale fracture simulations on brittle materials [3]. Wheeler et al. designed an adaptive and parallel phase-field framework, based on the finite element open-source library deal.II [18] with the quasi-monolithic semi-smooth Newton solver for fracture propagation problems in porous media [8]. Rudshaug et al. proposed a staggered and parallel phase-field model on the commercial finite element solver, LS-DYNA, to predict complex and dynamic crack propagation by using a new crack driving force [10]. Wang et al. applied MPI to develop a parallel phase-field model combined with the Mohr-Coulomb failure criterion to simulate the dynamic and quasi-static fracture in 3D rock-like materials under compressive load [11]. Samaniego at al. implemented a phase-field model for shear band formation within Alya, a parallel programming environment, to study the behavior of brittle and ductile fractures [12].

In this paper, we adopt a fully distributed grid strategy through MPI to parallelize our previous algorithm, built up by Threading Building Blocks (TBB) [13], which was based on the limited-memory BFGS (L-BFGS) algorithm [17] to solve for 2D or 3D crack simulations. By the distributed mesh strategy, the domain is fully divided into non-overlapping subdomains and stored independently on different CPUs so that a series of smaller sub-tasks are solved at the same time. This not only saves the computational time, but also reduces the memory requirement. Once each CPU finishes its own task, the results will be synchronized via MPI to obtain the result of the entire task. This program is implemented and parallelized within an open-source finite element library deal.II [18] using C++. The paper is organized as follows. In Section II, we review the phase-field formulation for crack propagation in brittle materials, including the governing equations, the weak form, and the finite element formulation. In Section III, we summarize the L-BFGS method to minimize the non-convex energy functional. In Section IV, we provide scaling analyses for 2D and 3D tests as well as report the speed-up ratio and efficiency for running the tests on a multi-core computer and the Digital Research Alliance of Canada's high-performance cluster.

II. Revisiting of Phase-field crack formulation

We briefly review the phase-field formulation used in this paper for crack propagation problems [14], including the governing equations, the weak form, and the corresponding finite element formation.

A. Governing Equations

The energy functional $\Pi(\boldsymbol{u}, d)$ of the fractured solid system is expressed in terms of a vector-valued displacement $\boldsymbol{u}(\boldsymbol{x})$ and a scalar phase-field $d(\boldsymbol{x}) \in [0, 1]$ for the crack regularization by the states between the cracked region ($d = 1$) and the intact material ($d = 0$), and can be split into three components as

$$\Pi(\boldsymbol{u}, d) = \Pi_{\boldsymbol{\epsilon}}\big(\boldsymbol{\epsilon}(\boldsymbol{u}), d\big) + \Pi_{\mathrm{c}}(d) - \Pi_{\mathrm{ext}}(\boldsymbol{u}), \tag{1}$$

where $\Pi_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}, d)$ is the strain energy stored in the material, $\Pi_{\mathrm{c}}(d)$ is the surface energy for crack surface generation, and $\Pi_{\mathrm{ext}}(\boldsymbol{u})$ is the work done by the external forces. The primary unknowns, $\boldsymbol{u}(\boldsymbol{x})$ and $d(\boldsymbol{x})$ can be computed by minimizing the total energy functional in Eq. (1), under the irreversibility constraints for the damage. Additionally, we assume the linear elastic material response and the relationship between the linear strain tensor and the small displacement field, defined as $\boldsymbol{\epsilon}(\boldsymbol{u}) = \widetilde{\nabla}\boldsymbol{u} := \frac{1}{2}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)$. The strain energy is the integral of the strain energy density function ψ over the volume $\Pi_{\boldsymbol{\epsilon}}\big(\boldsymbol{\epsilon}(\boldsymbol{u}), d\big) = \int_\Omega \psi\big(\boldsymbol{\epsilon}(\boldsymbol{u}, d)\big)\ \mathrm{d}\Omega$. The strain energy density can be decomposed into the positive ψ^+ and negative ψ^- portions and only the positive part will be degraded by the damage,

$$\psi\big(\boldsymbol{\epsilon}, d\big) = \big[\zeta(d) + k\big]\psi^+(\boldsymbol{\epsilon}) + \psi^-(\boldsymbol{\epsilon}), \tag{2}$$

where we use a degradation function as $\zeta(d) = (1-d)^2$, and a small $k > 0$ is introduced to improve numerical convergence. By the spectral decomposition [15], the strain tensor can be decomposed into $\boldsymbol{\epsilon}^\pm = \sum_i \langle\epsilon_i\rangle_\pm\, \mathbf{n}_i \otimes \mathbf{n}_i$, in which $\langle x\rangle_\pm := (x \pm |x|)/2$. The positive part $\boldsymbol{\epsilon}^+$ is associated with tensile strain, while the negative part $\boldsymbol{\epsilon}^-$ represents the compressive strain. Therefore, the strain energy density for the isotropic elastic material becomes

$$\psi^\pm(\boldsymbol{\epsilon}) = \frac{\lambda}{2}\langle \mathrm{tr}\boldsymbol{\epsilon}\rangle_\pm^2 + \mu\boldsymbol{\epsilon}^\pm : \boldsymbol{\epsilon}^\pm, \tag{3}$$

where $\mathrm{tr}(\boldsymbol{\epsilon})$ is the trace of the strain tensor, and λ and μ are the Lamé parameters. Correspondingly, the stress components are defined as

$$\boldsymbol{\sigma}^\pm = \frac{\partial\psi^\pm(\boldsymbol{\epsilon}, d)}{\partial\boldsymbol{\epsilon}} = \lambda\langle \mathrm{tr}\boldsymbol{\epsilon}\rangle_\pm\, \mathbf{I} + 2\mu\sum_{i=1}^{3}\langle\epsilon_i\rangle_\pm \mathbf{n}_i \otimes \mathbf{n}_i. \tag{4}$$

The main characteristic of the phase-field method is to use a regularized diffusive crack band, instead of a sharp one, to circumvent the discontinuity of the fracture. The surface energy, $\Pi_{\mathrm{c}}(d)$ in Eq. (1), is the energy required to form the corresponding crack surface, and its value is proportional to the crack area Γ_c, expressed as $\Pi_{\mathrm{c}}(d) = g_c\Gamma_c$, in which $g_c > 0$ is the critical energy release rate. The crack area is derived [14] using a length-scale parameter ℓ and the phase-field d as

$$\Gamma_c = \int_\Omega \frac{1}{2\ell}\left(d^2 + \ell^2\nabla d \cdot \nabla d\right)\ \mathrm{d}\Omega. \tag{5}$$

Lastly, the work done by the external forces is given by

$$\Pi_{\mathrm{ext}}(\boldsymbol{u}) = \int_\Omega \mathbf{b}\cdot\boldsymbol{u}\ \mathrm{d}\Omega + \int_{\partial\Omega}\mathbf{t}\cdot\boldsymbol{u}\ \mathrm{d}\Gamma, \tag{6}$$

where $\mathbf{b}$ is the body force, and $\mathbf{t}$ is the traction applied on the boundaries.

B. Weak Form

Performing the directional derivative of the energy functional in Eq. (1) using the variations of the displacement field $\delta\boldsymbol{u}$ and phase-field δd yields the following weak form,

$$\big(\widetilde{\nabla}\delta\boldsymbol{u},\ \boldsymbol{\sigma}\big) - \big(\delta\boldsymbol{u},\ \mathbf{b}\big) - \big(\delta\boldsymbol{u},\ \mathbf{t}\big)_{\Gamma_\mathbf{t}} = \mathbf{0} \tag{7}$$

$$\Big(\delta d,\ \frac{g_c}{\ell}d\Big) + \big(\delta d,\ \zeta^{'}\psi^+\big) + \big(\nabla\delta d,\ g_c\ell\nabla d\big) = 0, \tag{8}$$

in which the binary form is defined as $(a,b) := \int_\Omega a\cdot b \,\mathrm{d}\Omega$ and $(a,b)_{\Gamma_\mathbf{t}} := \int_{\Gamma_\mathbf{t}} a\cdot b \,\mathrm{d}\Gamma$. We replace the positive strain energy ψ^+ in Eq. (8) by a historical variable $\mathcal{H}$ of its maximum to prevent the crack from healing,

$$\mathcal{H}(\boldsymbol{x},t) = \max_{s\in[0,t]} \ \psi^+\big(\boldsymbol{\epsilon}(\boldsymbol{x},s)\big). \tag{9}$$

In addition, a viscosity regularization term $(\eta\dot{d},\delta d)$ can be introduced into Eq. (8) to stabilize the numerical treatment approximated by a pseudo-time increment $\Delta t = t_{n+1} - t_n$ as

$$\begin{aligned}\left(\delta d,\ \eta\frac{d_{n+1}-d_n}{\Delta t} + \zeta^{'}\mathcal{H} + \frac{g_c}{\ell}\right).\\ +\big(\nabla\delta d,\ g_c\ell\nabla d\big) = 0\end{aligned} \tag{10}$$

To solve the non-linear equations (7) and (10), the equations are linearized with respect to the increments of vector-valued displacement $\underline{\boldsymbol{u}}$ and the scalar phase-field $\underline{d}$ at the i-th iteration,

$$\begin{cases}\mathcal{D}_{\underline{\boldsymbol{u}}}\mathbf{r}_{\boldsymbol{u}}(\boldsymbol{u}^{[i]},d^{[i]}) + \mathcal{D}_{\underline{d}}\mathbf{r}_{\boldsymbol{u}}(\boldsymbol{u}^{[i]},d^{[i]}) = -\mathbf{r}_{\boldsymbol{u}}(\boldsymbol{u}^{[i]},d^{[i]})\\ \mathcal{D}_{\underline{\boldsymbol{u}}}\mathrm{r}_d(\boldsymbol{u}^{[i]},d^{[i]}) + \mathcal{D}_{\underline{d}}\mathrm{r}_d(\boldsymbol{u}^{[i]},d^{[i]}) = -\mathrm{r}_d(\boldsymbol{u}^{[i]},d^{[i]})\end{cases} \tag{11}$$

where $\mathbf{r}_{\boldsymbol{u}}(\boldsymbol{u},d)$ and $\mathrm{r}_d(\boldsymbol{u},d)$ represent the residuals.

C. Finite Element Formulation

The vector-valued displacement $\boldsymbol{u}(\boldsymbol{x})$ and the scalar phase-field $d(\boldsymbol{x})$ can be approximated by the shape functions $\{\boldsymbol{\varphi}(\boldsymbol{x}),\phi(\boldsymbol{x})\}$,

$$\boldsymbol{u}(\boldsymbol{x}) \approx \sum_{\alpha=1}^{n}\boldsymbol{\varphi}_\alpha(\boldsymbol{x})\boldsymbol{u}_\alpha, \quad d(\boldsymbol{x}) \approx \sum_{\alpha=1}^{n}\phi_\alpha(\boldsymbol{x})d_\alpha,$$

where $\boldsymbol{u}_\alpha$ and d_α denote the displacement and the phase-field values at the α-th node. After taking the Galerkin formulation, Eq. (11) can be approximated and expressed by a blocked linear system for the monolithic increments:

$$\begin{bmatrix}\mathbf{K}^{(i)}_{\boldsymbol{uu}} & \mathbf{K}^{(i)}_{\boldsymbol{u}d}\\ \mathbf{K}^{(i)}_{d\boldsymbol{u}} & \mathrm{K}^{(i)}_{dd}\end{bmatrix}\begin{bmatrix}\underline{\boldsymbol{u}}^{(i)}\\ \underline{d}^{(i)}\end{bmatrix} = \begin{bmatrix}-\mathbf{r}^{(i)}_{\boldsymbol{u}}\\ -r^{(i)}_d\end{bmatrix}. \tag{12}$$

Specifically, the entries of the stiffness matrix and the residual in Eq. (12) are expressed as

$$\begin{aligned}\mathbf{K}^{ab}_{\boldsymbol{uu}} &= \left(\widetilde{\nabla}\boldsymbol{\varphi}_a, \frac{\partial\boldsymbol{\sigma}}{\partial\boldsymbol{\epsilon}} : \widetilde{\nabla}\boldsymbol{\varphi}_b\right),\ \mathbf{K}^{ab}_{\boldsymbol{u}d} = \left(\widetilde{\nabla}\boldsymbol{\varphi}_a, \frac{\partial\boldsymbol{\sigma}}{\partial d}\phi_b\right),\\ \mathbf{K}^{ab}_{d\boldsymbol{u}} &= \left(\phi_a, \zeta^{'}\frac{\partial\mathcal{H}}{\partial\boldsymbol{\epsilon}} : \widetilde{\nabla}\boldsymbol{\varphi}_b\right),\\ \mathrm{K}^{ab}_{dd} &= \left(\phi_a, \left[\frac{g_c}{\ell} + \frac{\eta}{\Delta t} + \zeta^{''}\mathcal{H}\right]\phi_b\right) + \left(\nabla\phi_a, g_c\ell\nabla\phi_b\right),\\ \mathbf{r}^a_{\boldsymbol{u}} &= \left(\widetilde{\nabla}\boldsymbol{\varphi}_a, \boldsymbol{\sigma}\right) - \left(\boldsymbol{\varphi}_a, \mathbf{b}\right) - \left(\boldsymbol{\varphi}_i, \mathbf{t}\right)_{\Gamma_\mathbf{t}},\\ r^a_d &= \left(\phi_a, \frac{g_c}{\ell}d + \eta\frac{d-d_n}{\Delta t} + \zeta^{'}\mathcal{H}\right) + \left(\nabla\phi_a, g_c\ell\nabla d\right).\end{aligned}$$

The material tangent modulus is defined as

$$\frac{\partial\boldsymbol{\sigma}}{\partial\boldsymbol{\epsilon}} = [\zeta + k]\frac{\partial\boldsymbol{\sigma}^+}{\partial\boldsymbol{\epsilon}} + \frac{\partial\boldsymbol{\sigma}^-}{\partial\boldsymbol{\epsilon}}, \tag{13}$$

where

$$\frac{\partial\boldsymbol{\sigma}^\pm}{\partial\boldsymbol{\epsilon}} = \lambda H(\pm\mathrm{tr}\boldsymbol{\epsilon})\mathbf{I}\otimes\mathbf{I} + 2\mu\mathbb{P}^\pm, \tag{14}$$

with $H(x) = 1$ if $x \geq 0$ and $H(x) = 0$ if $x < 0$. The two fourth-order projection tensors are defined as

$$\mathbb{P}^\pm = \sum_i H(\pm\epsilon_i)\mathbf{n}_i\otimes\mathbf{n}_i\otimes\mathbf{n}_i\otimes\mathbf{n}_i + \frac{1}{2}\sum_i\sum_{j\neq i}\vartheta^\pm_{ij}\frac{\mathbb{G}_{ij}+\mathbb{G}_{ji}}{2},$$

where $\vartheta^\pm_{ij} = \frac{\langle\epsilon_i\rangle_\pm - \langle\epsilon_j\rangle_\pm}{\epsilon_i-\epsilon_j}$ if $\epsilon_i \neq \epsilon_j$ and $\vartheta^\pm_{ij} = [H(\pm\epsilon_i) + H(\pm\epsilon_j)]/2$ if $\epsilon_i = \epsilon_j$, and

$$\mathbb{G}_{ij} = \mathbf{n}_i\otimes\mathbf{n}_j\otimes\mathbf{n}_i\otimes\mathbf{n}_j + \mathbf{n}_i\otimes\mathbf{n}_j\otimes\mathbf{n}_j\otimes\mathbf{n}_i.$$

Lastly, the linearized terms can be computed as

$$\frac{\partial\boldsymbol{\sigma}}{\partial d} = \zeta^{'}\boldsymbol{\sigma}^+,\ \frac{\partial\mathcal{H}}{\partial\boldsymbol{\epsilon}} = \begin{cases}\boldsymbol{\sigma}^+(\boldsymbol{u}^{(i)}) & ,\psi^+\big(\boldsymbol{\epsilon}(\boldsymbol{u}^i)\big) > \mathcal{H}_n\\ \mathbf{0} & ,\psi^+\big(\boldsymbol{\epsilon}(\boldsymbol{u}^i)\big) \leq \mathcal{H}_n\end{cases}.$$

III. Limited-memory BFGS scheme

It is well-known that the total energy functional in Eq. (1) is non-convex, for instance, see [22] and [23]. In this work, we use the limited-memory version of the BFGS scheme to overcome the convergence difficulties. As a quasi-Newton method for non-linear optimization problems, the L-BFGS scheme follows an iterative pattern

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\mathbf{B}_k^{-1}\nabla f(\boldsymbol{x}_k) = \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k \tag{15}$$

to find the solution, where α_k is the step length and $\boldsymbol{p}_k$ is the search direction in the k-th iteration defined as

$$\mathbf{B}_k\boldsymbol{p}_k = -\boldsymbol{r}_k = -\nabla f(\boldsymbol{x}_k).$$

A. Conventional BFGS Scheme

In the conventional BFGS algorithm, two vectors $\boldsymbol{s}_k$ and $\boldsymbol{y}_k$ are defined by the solution vector $\boldsymbol{x} = \{\boldsymbol{u},d\}^T$ and the residual vector $\boldsymbol{r} = \{\boldsymbol{r}_{\boldsymbol{u}},r_d\}^T$ as $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, $\boldsymbol{y}_k = \boldsymbol{r}_{k+1} - \boldsymbol{r}_k$. And then, we can define a scalar variable ρ_k and a matrix $\mathbf{V}_k$ as

$$\rho_k = 1/\boldsymbol{y}_k^T\boldsymbol{s}_k,\ \mathbf{V}_k = \mathbf{I} - \rho_k\boldsymbol{y}_k\boldsymbol{s}_k^T.$$

The inverse of the Hessian $\mathbf{H}_k = \mathbf{B}_k^{-1}$ can be approximated by a recursive expression,

$$\mathbf{H}_k = \mathbf{V}_{k-1}^T\mathbf{H}_{k-1}\mathbf{V}_{k-1} + \rho_{k-1}\boldsymbol{s}_{k-1}\boldsymbol{s}_{k-1}^T.$$

By defining an operator for the matrix multiplication, $\mathcal{V}^b_a = \mathbf{V}_a\mathbf{V}_{a+1}\ldots\mathbf{V}_{b-1}\mathbf{V}_b, a < b$, the inverse of the Hessian $\mathbf{H}_k$ can be expanded by m $(m \leq k)$ steps as,

$$\begin{aligned}\mathbf{H}_k &= (\mathcal{V}^{k-1}_{k-m})^T\mathbf{H}_{k-m}(\mathcal{V}^{k-1}_{k-m})\\ &+ \sum_{i=k-m}^{k-2}\rho_i(\mathcal{V}^{k-1}_{i+1})^T\boldsymbol{s}_i\boldsymbol{s}_i^T(\mathcal{V}^{k-1}_{i+1}) + \rho_{k-1}\boldsymbol{s}_{k-1}\boldsymbol{s}_{k-1}^T.\end{aligned} \tag{16}$$

B. Limited-memory BFGS Scheme

Since the conventional BFGS method has to store a fully dense inverse of the Hessian matrix $\mathbf{H}_{k-m}$ for the recursive computation shown in Eq. (16), it is not suitable for large scale finite element problems. In order to avoid the stringent memory requirement, we can employ a sparse matrix $\mathbf{H}_k^{(0)}$ as

the initial guess instead of a fully dense matrix $\mathbf{H}_{k-m}$ needed in Eq. (16). The initial sparse matrix is defined as:

$$\mathbf{H}_k^{(0)} = \hat{\mathbf{K}}^{-1} = \begin{bmatrix} \mathbf{K}_{\boldsymbol{uu}}^{-1} & \mathbf{0} \\ \mathbf{0}^T & \mathrm{K}_{dd}^{-1} \end{bmatrix}.$$

Therefore, only the above sparse matrices $\mathbf{H}_k^{(0)}$ and the most recent m vector pairs, $\{\boldsymbol{s}_i, \boldsymbol{y}_i\}, i = k-m, \ldots, k-1$ need to be stored in the memory to evaluate the inverse of the approximated Hessian $\mathbf{H}_k$.

Additionally, the search direction $\boldsymbol{p}_k$ can be obtained by the two series of vector-vector multiplications in addition to the initial guess $\mathbf{H}_k^{(0)}$ and the vector productions $\boldsymbol{s}_i \boldsymbol{s}_i^T$ in Eq. (16). A two-loop recursive method, owing to the associative law, such as $(\mathbf{a}\mathbf{b}^T)\mathbf{r} = \mathbf{a}(\mathbf{b}^T\mathbf{r})$, is adopted rather than using explicit matrix-matrix multiplications, such as $\mathbf{V}_i\mathbf{V}_j$. This can further save CPU time as the scale of the problem increases. This is because the computational complexity is reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(2n)$. For the detailed L-BFGS algorithm, see [13].

IV. Scaling Analyses

This section presents the scaling analyses for a 2D shear-loading test and a 3D inclusion tension test. We use a fully distributed mesh approach with the MPI to exchange data. The computational domain is divided into multiple smaller non-overlapping subdomains that are assigned to different CPUs. Each CPU only stores data for its assigned subdomain and performs local computing tasks, such as evaluating local stiffness matrix. Ideally, doubling the number of CPUs will halve the problem size for each CPU and reduce the total wall-clock time by half. However, in most parallel computing, there is overhead due to data exchange for global calculations, such as assembling the global stiffness matrix. Each CPU must complete its local task and then exchange results with others, which can cause delays by idling. As more CPUs are used, the data exchange overhead increases, and the workload per CPU decreases. If the problem scale or the number of DoFs per CPU is too small, the data exchange overhead becomes a significant factor, negatively impacting the speedup and even potentially making the computation slower compared with the counterpart using fewer CPUs.

For the adaptive mesh refinement technique, we apply the two-stage refinement strategy that is discussed in our previous paper [13]. Only elements with a phase-field value above a threshold will be refined, and their mesh sizes are controlled by the ratio between the diagonal length of the element h and the length-scale ℓ. Due to the adaptive refinement technique, only the elements in the cracked region will be refined. This process naturally causes an uneven increment in DoFs across the entire computational domain, resulting in an unbalanced workload among multiple CPUs, even if an equal partition is performed during the initial problem setup. To avoid this, the re-partitioning and re-allocation are essential after the adaptive refinement to ensure each CPU has a sub-problem of similar size. This re-partitioning will reduce the speed-up of the parallelization due to the communication cost, but the reduction in problem size through the adaptive mesh refinement improves the overall computational efficiency.

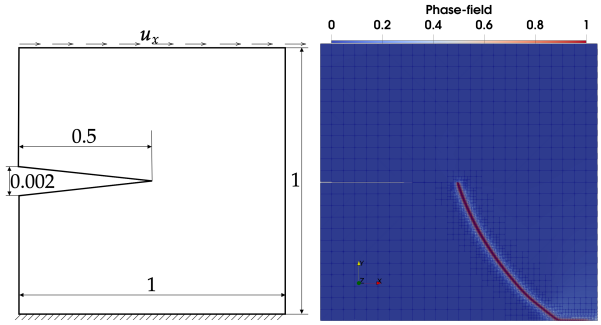

Figure. 1. The schematic (left) of the 2D plate (unit: mm) with a preexisting crack undergoing the displacement-controlled shear load on the top edge. The phase-field distribution (right) is displayed on the adaptively refined mesh when $u_x = 0.0015$ mm.

The speed-up ratio $\mathrm{S}(N_{\mathrm{CPUs}})$, and efficiency $\mathrm{E}(N_{\mathrm{CPUs}})$ by the increasing numbers of CPUs are defined as follows:

$$\mathrm{S}(N_{\mathrm{CPUs}}) = T(N_{\min})/T(N_{\mathrm{CPUs}}), \tag{17}$$

$$\mathrm{E}(N_{\mathrm{CPUs}}) = \mathrm{S}(N_{\mathrm{CPUs}})/(N_{\mathrm{CPUs}}/N_{\min}), \tag{18}$$

in which $T(\cdot)$ is the total time spent on the simulation, N_{CPUs} is the number of CPUs used in the simulation, and $N_{\min}$ is the simulation time needed by using the fewest CPUs. The larger the speedup the program obtains, the more efficiently the parallelization preforms. An ideal parallel problem has $\mathrm{S}(N_{\mathrm{CPUs}}) = N_{\mathrm{CPUs}}/N_{\min}$ and $\mathrm{E}(N_{\mathrm{CPUs}}) = 100\%$. The 2D shear test runs on a MacBook Pro with Apple M2 Max chips (8 performance cores @3.49 GHz, 4 efficiency cores @2.42 GHz) using Trilinos [20], while the 3D test is performed on the cluster Cedar provided by the Digital Research Alliance of Canada using PETSc [21].

A. 2D Shear-Loading Test

We use the proposed method for crack simulation in the 2D shear loading test involving 1, 2, 4, and 8 CPUs, respectively. The computational domain is a 2D unit plate with a preexisting crack, shown in Fig. 1. The top edge is prescribed with a horizontal displacement $u_x = 15.0 \times 10^{-3}$ mm with the increment $\Delta u_x = 10^{-4}$ mm, and the bottom edge is fully fixed. The material parameters include the Lamé parameters $\lambda = 121.15$ kN/mm^2 and $\mu = 80.77$ kN/mm^2, the critical energy release rate $g_c = 2.7 \times 10^{-3}$ kN/mm, and the length-scale parameter $\ell = 0.0075$ mm. The refinement parameters are set as threshold $d_T = 0.4$ and the ratio $h/\ell < 0.5$. The crack grows gradually until the last loading step $u_x = 0.0015$ mm, as shown in Fig. 1.

The computational domain is divided into 8 non-overlapping subdomains allocated to 8 CPUs correspondingly as shown in Fig. 2. As the crack grows, the re-partition is performed after the adaptive mesh refinement to ensure an even DoFs distribution among all the CPUs. The time consumption with different numbers of CPUs and the corresponding speed-up ratios and efficiencies calculated according to Eqs. (17)

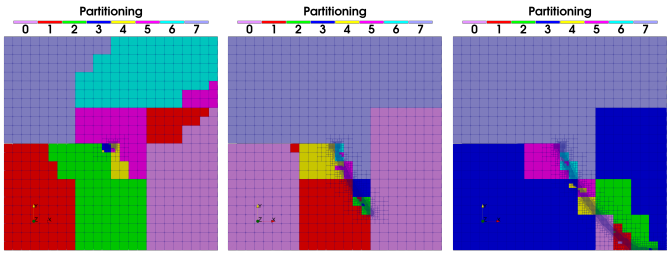

Figure. 2. The domain is divided into 8 non-overlapping subdomains, and each is assigned to a CPU. In the beginning (left), only the region around the preexisting crack tip is refined (2,883 DoFs). As the crack grows, the mesh is refined when $u_x = 0.0012$ mm (middle, 8,997 DoFs) and $u_x = 0.0015$ mm (right, 14,337 DoFs), the subdomains are re-assigned to the CPUs to re-balance the workload.

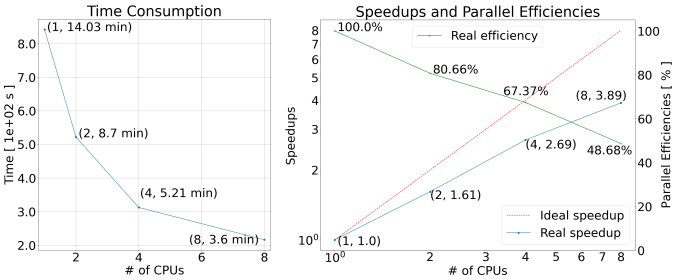

Figure. 3. The wall-clock time (left) using 1, 2, 4, and 8 CPUs to compute the shear-loading test and the corresponding speed-up ratios and efficiencies (right) versus the number of the CPUs.

and (18) are reported in Fig. 3. The computational time consistently decreases, while the speedup drops as more CPUs are used. This drop is caused by a higher overhead due to the data exchange and re-partitioning costs. Since this problem only has 14,337 DoFs (about 1,792 per CPU in the 8-CPU test), the size of the sub-problem assigned to each CPU is small, which also impacts the speed-up.

B. 3D Inclusion-Tension Test

We apply the proposed method for the crack propagation in a 3D inclusion problem using $2^N (N = 3, \ldots, 8)$ CPUs in the cluster Cedar provided by the Digital Research Alliance of Canada. The sample consists of two components: a softer matrix ($\lambda = 12.0$ kN/mm^2 and $\mu = 8.0$ kN/mm^2) and a rigid $1/8$ spherical inclusion ($\lambda = 12.0 \times 10^5$ kN/mm^2 and $\mu = 8.0 \times 10^5$ kN/mm^2), the geometry of which is shown in Fig. 4. The rest of the material parameters include the critical energy release rate $g_c = 5.0 \times 10^{-4}$ kN/mm, the viscosity parameter $\eta = 10^{-6}$ kN $\cdot$ s/mm^2, the length-scale parameter $\ell = 0.01$ mm, and the small positive parameter $k = 0.0$ in Eq. (2). The sample is subjected to a displacement-controlled tension along the z-direction at the surface ($z = 1$ mm) with the incremental displacement $\Delta u_z = 1.0 \times 10^{-3}$ mm for $u_z \leq 1.1 \times 10^{-2}$ mm and $\Delta u_z = 2.0 \times 10^{-4}$ mm up to $u_z = 1.3 \times 10^{-2}$ mm. Constraints are applied at the surface $x = 0$ mm with $u_x = 0$, the surface $y = 0$ mm with $u_y = 0$, and the surface $z = 0$ mm with $u_z = 0$, respectively. The threshold $d_T = 0.8$ and the ratio $h/\ell < 0.8$ are set for the adaptive mesh refinement. The phase-field distribution is reported in Fig. 4 when the crack grows rapidly across the sample when $u_z = 0.0122$ mm.

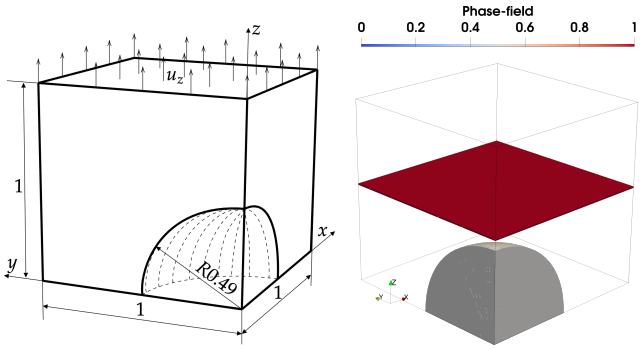

Figure. 4. The schematic (left) of a 3D unit cube (unit: mm) with a $1/8$ spherical inclusion subjected to the displacement-controlled load u_z applied on the top surface. The gray object represents the rigid spherical inclusion. The phase-field value ($d > 0.8$) is distributed inside the cube when $u_z = 0.0122$ mm (right).

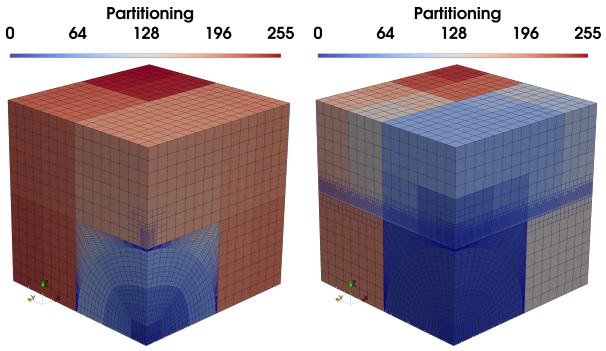

Figure. 5. The domain is divided into 256 non-overlapping subdomains, each of which is assigned to a CPU. In the beginning (51,132 DoFs, left), the region above the inclusion is refined with the partition unchanged until the crack is initiated. As the crack propagates through the sample (right), the mesh is adaptively refined (638,060 DoFs, about 2,492 per CPU), and the domain partitions are re-assigned to CPUs to re-balance the workload.

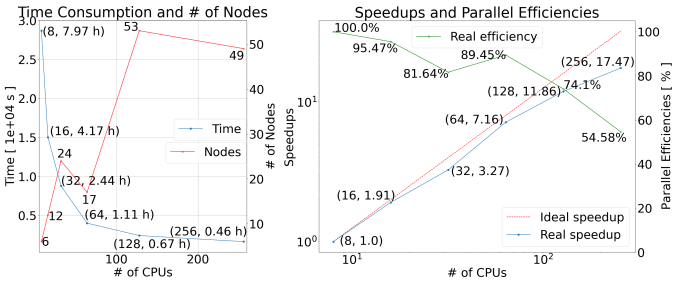

Figure. 6. The number of nodes and the total time (left) used on 8, 16, 32, 64, 128, and 256 CPUs to complete the 3D inclusion composite under tension loading test and the corresponding speed-up ratios defined by Eq. (17) and the efficiencies defined by Eq. (18) versus the number of the CPUs (right).

Figure 5 shows the reallocation of the subdomains to the used 256 CPUs before and after the adaptive mesh refinement, which balances the workload across all the CPUs. The number

of DoFs increases more than 12 times of the initial value, highlighting the importance of the adaptive refinement for 3D problems. The wall-clock time of the simulation, shown in Fig. 6, decreases consistently as more CPUs are involved. The MPI is a higher-layer communication protocol that relies on point-to-point connections, such as sockets and TCP/IP [19]. Unlike simulation on a single multi-processor machine, the communication structures on a cluster are more complex due to many computers located in various nodes separated in several islands and shared by different users via the virtual machine technique. The number of nodes strongly impacts the scaling efficiency. As shown in Fig. 6, the parallelization scales well from 8 to 128 CPUs. The efficiency improves at 128 CPUs because of the fewer nodes are involved. When $N_{\mathrm{CPUs}} \geq 128$, fewer DoFs are assigned to each CPU (about $4,984$ on 128 CPUs and $2,492$ on 256 CPUs), which increases the proportion of the communication costs and therefore reduces the overall efficiency.

CONCLUSION

In this paper, we propose an MPI-based parallel monolithic scheme using the limited-memory BFGS method for both 2D and 3D phase-field crack simulations. During parallelization, we employ a fully distributed mesh via the MPI using different numbers of CPUs to speed up the computation for large-scale problems. The adaptive mesh refinement technique is further introduced to reduce the total computational cost. The detailed scaling analyses are reported for a 2D problem and a 3D problem. We also introduce a re-partition technique to balance the workload among multiple CPUs due to the imbalance caused by the adaptive mesh refinement. Due to these techniques, we are able to model more realistic 3D crack propagation problems within a reasonable wall-clock time.

ACKNOWLEDGMENT

Tao Jin is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under the Discovery Grants Program (funding reference number: RGPIN-2021-02561). Kuiying Chen is supported by the National Research Council Canada DTS-SCS program A1-018177 project. Their supports are greatly appreciated.

REFERENCES

[1] Cervera, M., Barbat, G.B., Chiumenti, M., J.-Y. Wu. A Comparative Review of XFEM, Mixed FEM and Phase-Field Models for Quasi-brittle Cracking. Arch Computat Methods Eng 29, 1009–1083 (2022).

[2] Diehl, P., Lipton, R., Wick, T. Mayank. A comparative review of peridynamics and phase-field models for engineering fracture mechanics. Comput Mech 69, 1259–1293 (2022).

[3] Badri, M.A, G. Rastiello, and E. Foerster. "Preconditioning Strategies for Vectorial Finite Element Linear Systems Arising from Phase-Field Models for Fracture Mechanics." Computer Methods in Applied Mechanics and Engineering 373 (2021): 113472-.

[4] Liu, Guowei, Qingbin Li, Mohammed A Msekh, and Zheng Zuo. "Abaqus Implementation of Monolithic and Staggered Schemes for Quasi-Static and Dynamic Fracture Phase-Field Model." Computational Materials Science 121 (2016): 35–47.

[5] Heister, Timo, and Thomas Wick. "Pfm-Cracks: A Parallel-Adaptive Framework for Phase-Field Fracture Propagation." Software Impacts 6 (2020): 100045-.

[6] Ziaei-Rad, Vahid, and Yongxing Shen. "Massive Parallelization of the Phase Field Formulation for Crack Propagation with Time Adaptivity." Computer Methods in Applied Mechanics and Engineering 312 (2016): 224–53.

[7] Hao, Shourong, and Yongxing Shen. "An Efficient Parallel Solution Scheme for the Phase Field Approach to Dynamic Fracture Based on a Domain Decomposition Method." International Journal for Numerical Methods in Engineering 125, no. 6 (2024).

[8] Wheeler, Mary F, Thomas Wick, and Sanghyun Lee. "IPACS: Integrated Phase-Field Advanced Crack Propagation Simulator. An Adaptive, Parallel, Physics-Based-Discretization Phase-Field Framework for Fracture Propagation in Porous Media." Computer Methods in Applied Mechanics and Engineering 367 (2020): 113124-.

[9] Ishii, Gaku, Yusaku Yamamoto, and Takeshi Takaishi. "Acceleration and Parallelization of a Linear Equation Solver for Crack Growth Simulation Based on the Phase Field Model." Mathematics (Basel) 9, no. 18 (2021): 2248-.

[10] Rudshaug, Jonas, Tore Børvik, and Odd Sture Hopperstad. "Modeling Brittle Crack Propagation for Varying Critical Load Levels: A Dynamic Phase-Field Approach." International Journal of Fracture 245, no. 1–2 (2024): 57–73.

[11] Wang, Tao, Xuan Ye, Zhanli Liu, Dongyang Chu, and Zhuo Zhuang. "Modeling the Dynamic and Quasi-Static Compression-Shear Failure of Brittle Materials by Explicit Phase Field Method." Computational Mechanics 64, no. 6 (2019): 1537–56.

[12] Samaniego, C, J Ulloa, P Rodríguez, G Houzeaux, M Vázquez, and E Samaniego. "A Phase-Field Model for Ductile Fracture with Shear Bands: A Parallel Implementation." International Journal of Mechanical Sciences 200 (2021): 106424-.

[13] Jin T, Li Z, Chen K. A novel phase-field monolithic scheme for brittle crack propagation based on the limited-memory BFGS method with adaptive mesh refinement. Int J Numer Methods Eng. 2024; 125(22):e7572.

[14] Miehe, C, F Welschinger, and M Hofacker. "Thermodynamically Consistent Phase-Field Models of Fracture: Variational Principles and Multi-Field FE Implementations." International Journal for Numerical Methods in Engineering 83, no. 10 (2010): 1273–1311.

[15] Miehe, Christian, and Steffen Mauthe. "Phase Field Modeling of Fracture in Multi-Physics Problems. Part III. Crack Driving Forces in Hydro-Poro-Elasticity and Hydraulic Fracturing of Fluid-Saturated Porous Media." Computer Methods in Applied Mechanics and Engineering 304 (2016): 619–55.

[16] Nocedal, Jorge, Stephen J Wright, ProQuest, and ProQuest. Numerical Optimization. 2nd ed. New York: Springer, 2006.

[17] LIU, D. C, and J NOCEDAL. "On the Limited Memory BFGS Method for Large Scale Optimization." Mathematical Programming 45, no. 3 (1989): 503–28.

[18] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, D. Wells The deal.II finite element library: design, features, and insights Computers & Mathematics with Applications, vol. 81, pages 407-422, 2021.

[19] Lee, Gary. Cloud Networking: Understanding Cloud-Based Data Center Networks. Amsterdam: Elsevier, 2014.

[20] M. Heroux, R. Bartlett, V.H.R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, A. Williams, An Overview of Trilinos, Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[21] Balay S, Abhyankar S, Adams MF, et al. PETSc/TAO Users Manual. Tech. Rep. ANL-21 39–Revision 3.16. Argonne National Laboratory 2021.

[22] Bourdin B, Francfort G, Marigo JJ. Numerical experiments in revisited brittle fracture. J Mech Phys Solids. 2000;48(4):797-826.

[23] Bourdin B, Francfort GA, Marigo JJ. The variational approach to fracture. J Elast. 2008;91(1):5-148.