Finite Elements in Analysis and Design 166 (2019) 103320

Contents lists available at ScienceDirect

Finite Elements in Analysis and Design

journal homepage: www.elsevier.com/locate/finel

A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing

Rui Zhang [a], Sam Naboulsi [b,c], Thomas Eason [d], Dong Qian [a,*]

[a] Department of Mechanical Engineering, The University of Texas at Dallas, Richardson, TX 75080, USA
[b] HPCMP PETTT/SAIC, 2435 5th Street, Bldg. 676, Wright-Patterson Air Force Base, OH 45433, USA
[c] Air Force Research Laboratory, DoD Supercomputing Resource Center, Wright-Patterson Air Force Base, OH 45433, USA
[d] Air Force Research Laboratory, Air Vehicles Directorate, Wright-Patterson Air Force Base, Fairborn, OH 45433, USA

ARTICLE INFO

Keywords:
Space-time FEM
Enrichment
High cycle fatigue
Two-scale damage model
High-performance computing

ABSTRACT

A multiscale space/time computational framework for high cycle fatigue (HCF) life predictions is established by integrating the extended space-time finite element method (XTFEM) with a multiscale progressive damage model. While the robustness of the multiscale space/time method has been previously demonstrated, the associated high computational cost remains a critical barrier for practical applications. In this work, a novel hybrid iterative/direct linear system solver is first proposed with a unique preconditioner. Computational efficiency is further improved by taking advantage of the high-performance computing platform featuring hierarchy of the distributed- and the shared-memory parallelisms using CPUs and GPUs. Robustness of the accelerated framework is demonstrated through benchmark problems. It is shown that the serial version of the hybrid solver is at least 1–2 orders of magnitude faster in computing time and cheaper in memory consumption than the conventional sparse direct or iterative solver, while the parallel version efficiently handles XTFEM stiffness matrix equations with over 100 million unknowns using 64 CPU cores. Optimal speedups are achieved in the parallel implementations of the multiscale progressive damage model using either CPUs or GPUs. HCF simulations on 3D specimens are performed to quantify key effects due to mean stress and multiaxial load conditions.

1. Introduction

Fatigue is a failure mechanism that governs the design of many engineering structures and components. Most of the fatigue design approaches employed by the industry today belong to the category of either *safe-life* or *damage-tolerance* approach. However, those approaches are not without any shortcomings. In particular, both approaches rely on certain empirical relations that are derived from either experiment and/or curve-fitting. As such, extensions to complex fatigue loading conditions, such as extrapolations or corrections for mean stress effects, variable amplitudes, multiaxial loadings, and random loading spectrums remain questionable.

With the rapid advances in high-performance computing (HPC) platform in recent decades, there is an increasing interest in establishing simulation-based tools for fatigue life prediction. In the case of low cycle fatigue (LCF) failure, several simulation-based approaches have been developed by coupling *Finite Element Method* (FEM) with *Continuum Damage Mechanics* (CDM) [1–5]. However, for high cycle fatigue (HCF) problems, direct finite element simulation has not been possible due to the large number of loading cycles (typically ranges from 10^5 to 10^7). To circumvent this limitation, the so-called *jump-in-cycles* approach has been developed and adopted in Refs. [6–11]. It assumes a constant amplitude of the applied cyclic load then extrapolates internal and damage variables from several simulated cycles to large blocks of jumped cycles. Accordingly, the damage variable is cycle-dependent rather than stress/strain-dependent. Although the method greatly extends the predicative capabilities, the assumption of constant load amplitude does not always hold for practical applications.

To address the challenges associated with HCF simulations, Bhamare et al. [12] introduced a computational framework called extended space-time FEM (XTFEM). XTFEM is derived based on the time-discontinuous Galerkin (TDG) method established for elastodynamics [13–15], which discretizes the temporal domain as well as the spatial domain using finite elements. The TDG method has been proved to be A-stable and high-order accurate [13–18]. Based on the key concepts that are introduced in XFEM [19], generalized FEM (GFEM) [20]

* Corresponding author.
E-mail address: dong.qian@utdallas.edu (D. Qian).

https://doi.org/10.1016/j.finel.2019.103320
Received 22 April 2019; Received in revised form 5 August 2019; Accepted 8 August 2019
Available online 28 August 2019

and partition of unity method (PUM) [21,22], XTFEM further improves the approximation by enriching its polynomial-based temporal shape function with enrichment functions that represent the problem physics [12,23–26]. It was shown that enriched approximation effectively captured the dynamic response and enabled the use of very large time step size under various practical fatigue loading histories [12]. To address the multiscale material behavior in HCF, XTFEM was further coupled with a two-scale CDM model proposed by Lemaitre et al. [27,28] and Desmorat et al. [29]. The damage model approximates mesoscale material behavior as elastic, while plasticity and damage are modelled at the microscale that represents the scale of defects such as microcracks and microvoids. Those two scales are bridged through the modified Eshelby-Kröner localization law [30,31]. With this integration, direct numerical simulations of HCF in 304L stainless steel specimen up to 1 *million* cycles have been successfully completed and verified in Ref. [12]. More recently, Wada et al. [32] further extended this framework to predict cyclic failure of rubber by considering both geometric and material nonlinearities at mesoscale. A CDM-based model developed by Cantournet et al. [33] and Lemaitre et al. [34] was incorporated to account for the damage evolution of rubber. HCF simulations of the notched synthetic rubber sheet specimen up to 1 *million* cycles were performed. The simulation results demonstrated good agreement with experimental results.

While the robustness of XTFEM approach has been extensively demonstrated, the high computational cost remains as a critical barrier for practical applications. The computational challenges are twofold. First, due to the additional temporal dimension and the enrichment degrees of freedom (DOFs) that are introduced, XTFEM leads to linear systems of equations that are much larger than those derived from regular FEM [12]. In addition, the corresponding enriched space-time stiffness matrix is non-symmetric, less sparse compared to that of regular FEM [35] and generally not well conditioned. The second significant contribution to computational cost comes from the multiscale fatigue damage model, which needs to be resolved at multiple spatial-temporal quadrature points.

Based on the prior efforts, the main objective of this work is to enhance the efficiency of the XTFEM/CDM computational framework and extend it to large-scale 3D HCF simulations. To accomplish this, a novel hybrid linear system solver is established and the corresponding implementation on HPC platform is developed. The hybrid iterative/direct linear system solver is proposed by utilizing the special block structure and unique properties of the XTFEM stiffness matrix. HPC implementation is aimed at accelerating both the hybrid linear system solver and nonlinear constitutive solver. It features a hierarchy of parallelisms that first partition the space-time computational domain and then redistribute the computationally-intensive tasks associated with each subdomain while minimizing the communication. This framework is implemented using a hybrid parallel programming model, which combines the *Message Passing Interface* (MPI) for distributed-memory parallelisms, the *Open Multi-Processing* (OpenMP) for shared-memory parallelisms, and the *Compute Unified Device Architecture* (CUDA) for the heterogeneous CPU + GPU hardware platforms. The computational performance of the established framework is demonstrated through several benchmark problems. It is shown that the single-thread performance of the proposed hybrid solver is at least 1–2 orders of magnitude better than conventional sparse direct or iterative linear system solver in terms of both computing time and memory consumption. The parallel hybrid solver handles XTFEM stiffness matrix equations with over 100 *million* unknowns using 64 CPU cores and shows excellent parallel efficiency. Parallel implementations of the CDM-based nonlinear constitutive solver show optimal speedup using either CPUs or GPUs. Benchmark problems on HCF applications demonstrate the capabilities of the proposed framework in handling complex fatigue loading conditions.

The rest of this paper is organized as follows. In Section 2, we briefly review the extended space-time finite element method. In Section 3, the hybrid iterative/direct linear system solver for XTFEM is developed. Section 4 presents the two-scale damage model and its parallel implementation. Results and discussions of benchmark problems are provided in Section 5. Finally, conclusions are drawn in Section 6.

2. Extended space-time finite element method

We start by briefly reviewing the formulation of the extended space-time finite element method [12], which is derived based on the time-discontinuous Galerkin method [13–15].

2.1. Governing equations

We consider the initial-boundary-value problem defined over a spatial region Ω and the corresponding temporal domain $I =]0, T[$. The region Ω is bounded by $\Gamma = \Gamma_t \cup \Gamma_u$, where Γ_t and Γ_u are respectively the Neumann and the Dirichlet boundaries and $\Gamma_t \cap \Gamma_u = \varnothing$. The strong form of the governing equations is given by

$$\rho\ddot{\mathbf{u}} = \nabla\cdot\boldsymbol{\sigma}(\nabla\mathbf{u}) + \mathbf{f} \qquad \text{on} \quad Q \equiv \Omega \times]0,T[\tag{1}$$

$$\mathbf{u} = \mathbf{u} \qquad \text{on} \quad \Upsilon_u \equiv \Gamma_u \times]0,T[\tag{2}$$

$$\mathbf{n}\cdot\boldsymbol{\sigma}(\nabla\mathbf{u}) = \mathbf{t} \qquad \text{on} \quad \Upsilon_t \equiv \Gamma_t \times]0,T[\tag{3}$$

$$\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}) \qquad \text{for} \quad \mathbf{x}\in\Omega \tag{4}$$

$$\dot{\mathbf{u}}(\mathbf{x},0) = \mathbf{v}_0(\mathbf{x}) \qquad \text{for} \quad \mathbf{x}\in\Omega \tag{5}$$

where ρ $(\mathbf{x})$ is the volumetric mass density, $\mathbf{u}$ and $\mathbf{f}$ are the displacement and body force vectors, $\mathbf{n}$ is the outward unit vector normal to boundary Γ, $\boldsymbol{\sigma}(\nabla\mathbf{u}) = \mathbf{C} : \boldsymbol{\varepsilon}$ and $\mathbf{C}$ is the constitutive matrix, $\mathbf{u}$ and $\mathbf{t}$ are the prescribed essential boundary condition and traction, $\mathbf{u}_0$ and $\mathbf{v}_0$ denote the initial displacement and velocity. Partial differentiations with respect to time are denoted by superposed dots.

2.2. Time-discontinuous Galerkin method

The time-discontinuous Galerkin method developed by Hughes and Hulbert [13–15] for elastodynamics is adopted in this work. First, the space-time domain $Q = \Omega \times I$ is divided into multiple segments termed space-time slabs. The n-th space-time slab is given by $Q_n = \Omega \times I_n$ where $I_n =]t_{n\text{-}1}, t_n[$. The corresponding Neumann and Dirichlet boundary conditions are defined on $(\Upsilon_u)_n = \Gamma_u \times I_n$ and $(\Upsilon_t)_n = \Gamma_t \times I_n$ respectively. Space-time slab Q_n is further discretized into $(n_{el})_n$ space-time elements. The approximations established will be denoted with a superscript "h". The domain (interior) of the eth element defined as $Q_n^e \subset Q_n$ and its boundary as Υ_n^e. The domain and boundary of the interior of the slab are defined as $Q_n^\Sigma = \cup_{e=1}^{(n_{el})_n} Q_n^e$ and $\Upsilon_n^\Sigma = \cup_{e=1}^{(n_{el})_n} \Upsilon_n^e - \Upsilon_n$ respectively. Fig. 1 shows an example of the TDG space-time discretization described above.

The following notations are introduced for deriving the TDG formulation,

$$\left(\mathbf{w}^h, \mathbf{u}^h\right)_\Omega = \int_\Omega \mathbf{w}^h \cdot \mathbf{u}^h d\Omega \tag{6}$$

$$a\left(\mathbf{w}^h, \mathbf{u}^h\right)_\Omega = \int_\Omega \nabla\mathbf{w}^h \cdot \sigma\left(\nabla\mathbf{u}^h\right) d\Omega \tag{7}$$

$$\left(\mathbf{w}^h, \mathbf{u}^h\right)_{Q_n} = \int_{Q_n} \mathbf{w}^h \cdot \mathbf{u}^h dQ \tag{8}$$

$$a\left(\mathbf{w}^h, \mathbf{u}^h\right)_{Q_n} = \int_{Q_n} \nabla\mathbf{w}^h \cdot \sigma\left(\nabla\mathbf{u}^h\right) dQ \tag{9}$$

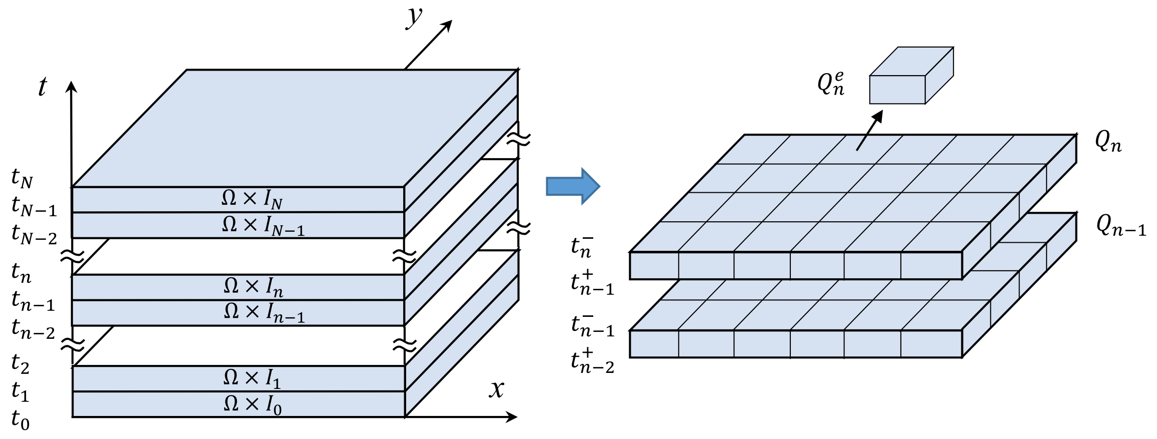

Fig. 1. An illustration of TDG space-time discretization for the case of 2D spatial domain.

$$(\mathbf{w}^h, \mathbf{u}^h)_{Q_n^\Sigma} = \int_{Q_n^\Sigma} \mathbf{w}^h \cdot \mathbf{u}^h dQ \tag{10}$$

$$(\mathbf{w}^h, \mathbf{u}^h)_{\Upsilon_n^\Sigma} = \int_{\Upsilon_n^\Sigma} \mathbf{w}^h \cdot \mathbf{u}^h d\Upsilon \tag{11}$$

$$(\mathbf{w}^h, \mathbf{u}^h)_{(\Upsilon_t)_n} = \int_{(\Upsilon_t)_n} \mathbf{w}^h \cdot \mathbf{u}^h d\Upsilon \tag{12}$$

where $\int_{Q_n}(\bullet)dQ = \int_{I_n}\int_{\Omega}(\bullet)d\Omega dt$ and $\int_{\Upsilon_n}(\bullet)d\Upsilon = \int_{I_n}\int_{\Gamma}(\bullet)d\Gamma dt$. We further introduce the jump operators

$$[\![\mathbf{u}(t_n)]\!] = \mathbf{u}(t_n^+) - \mathbf{u}(t_n^-) \tag{13}$$

$$[\![\mathbf{u}(\mathbf{x})]\!] = \mathbf{u}(\mathbf{x}^+) - \mathbf{u}(\mathbf{x}^-) \tag{14}$$

in which

$$\mathbf{u}(t_n^\pm) = \lim_{\varepsilon \to 0^\pm} \mathbf{u}(t_n + \varepsilon) \tag{15}$$

$$\mathbf{u}(\mathbf{x}^\pm) = \lim_{\varepsilon \to 0^\pm} \mathbf{u}(\mathbf{x} + \varepsilon \mathbf{n}) \tag{16}$$

$$\mathbf{n} = \mathbf{n}^+ = -\mathbf{n}^- \tag{17}$$

The weak form of TDG formulation is derived by introducing displacement trial function $\mathbf{u}^h(\mathbf{x}, t)$ and test function $\delta\mathbf{u}^h(\mathbf{x}, t)$ to be C^0 continuous within each space-time slab. Discontinuities across the adjacent space-time slabs are allowed for the trial and test functions. The spaces of the trial and test functions are given by

$$\mathbf{u}^h(\mathbf{x}, t) \in U \qquad U = \left\{\mathbf{u}^h(\mathbf{x}, t) \middle| \mathbf{u}^h \in C^0\left(\cup_{n=1}^N Q_n\right),\ \mathbf{u}^h = \mathbf{u} \text{ on } \Gamma_u\right\} \tag{18}$$

$$\delta\mathbf{u}^h(\mathbf{x}, t) \in U_0 \qquad U_0 = \left\{\delta\mathbf{u}^h(\mathbf{x}, t) \middle| \delta\mathbf{u}^h \in C^0\left(\cup_{n=1}^N Q_n\right),\ \delta\mathbf{u}^h = 0 \text{ on } \Gamma_u\right\} \tag{19}$$

With these definitions, the weak form of TDG formulation is derived. For the n-th space-time slab, it is given as

$$B_{DG}\left(\delta\mathbf{u}^h, \mathbf{u}^h\right)_n = L_{DG}\left(\delta\mathbf{u}^h\right)_n \tag{20}$$

for $n = 1, 2, \ldots$, where

$$\begin{aligned} B_{DG}\left(\delta\mathbf{u}^h, \mathbf{u}^h\right)_n = {} & \left(\delta\dot{\mathbf{u}}^h, \rho\ddot{\mathbf{u}}^h\right)_{Q_n} + a\left(\delta\dot{\mathbf{u}}^h, \mathbf{u}^h\right)_{Q_n} + \left(\delta\dot{\mathbf{u}}^h\left(t_{n-1}^+\right), \rho\dot{\mathbf{u}}^h\left(t_{n-1}^+\right)\right)_\Omega \\ & + a\left(\delta\mathbf{u}^h\left(t_{n-1}^+\right), \mathbf{u}^h\left(t_{n-1}^+\right)\right)_\Omega \end{aligned} \tag{21}$$

$$\begin{aligned} L_{DG}\left(\delta\mathbf{u}^h\right)_n = {} & \left(\delta\dot{\mathbf{u}}^h, \mathbf{f}\right)_{Q_n} + \left(\delta\dot{\mathbf{u}}^h, \mathbf{t}\right)_{(\Upsilon_t)_n} + \left(\delta\dot{\mathbf{u}}^h\left(t_{n-1}^+\right), \rho\dot{\mathbf{u}}^h\left(t_{n-1}^-\right)\right)_\Omega \\ & + a\left(\delta\mathbf{u}^h\left(t_{n-1}^+\right), \mathbf{u}^h\left(t_{n-1}^-\right)\right)_\Omega \end{aligned} \tag{22}$$

2.3. Discretization

In XTFEM, the unknown displacement field is approximated by

$$\mathbf{u}(\mathbf{x}, t) = \sum_I \mathbf{N}_I(\mathbf{x}, t)\mathbf{d}_I + \sum_J \tilde{\mathbf{N}}_J(\mathbf{x}, t)\tilde{\mathbf{d}}_J \tag{23}$$

in which $\mathbf{N}$ and $\tilde{\mathbf{N}}$ are standard and enriched space-time shape functions, $\mathbf{d}$ and $\tilde{\mathbf{d}}$ represent the standard and enriched DOFs respectively.

The standard space-time shape function is constructed in a multiplicative form, given by

$$\mathbf{N}(\mathbf{x}, t) = \mathbf{N}_t \otimes \mathbf{N}_\mathbf{x} = \begin{bmatrix} N_1\mathbf{N}_\mathbf{x} & \cdots & N_i\mathbf{N}_\mathbf{x} & \cdots & N_{n_t}\mathbf{N}_\mathbf{x} \end{bmatrix} \tag{24}$$

where $\mathbf{N}_\mathbf{x}$ and $\mathbf{N}_t$ are the spatial and standard temporal shape functions respectively, symbol $\otimes$ denotes the Kronecker product, subscript i represents the i-th temporal node. The spatial shape functions are the same as these in regular FEM. A quadratic interpolation scheme is employed in this work for the standard temporal shape function. Three temporal nodes are equally spaced along the time axis for each space-time slab, i.e., $n_t = 3$ in Eq. (24). The polynomials-based temporal shape function is given by

$$\mathbf{N}_t = \begin{bmatrix} \frac{(t_2 - t)(t_3 - t)}{(t_2 - t_1)(t_3 - t_1)} & \frac{(t_3 - t)(t_1 - t)}{(t_3 - t_2)(t_1 - t_2)} & \frac{(t_1 - t)(t_2 - t)}{(t_1 - t_3)(t_2 - t_3)} \end{bmatrix} \tag{25}$$

For certain class of problems, the polynomials-based shape functions do not provide the ideal basis for interpolation. Therefore, an enrichment function $\Phi(\mathbf{x}, t)$ that represents the problem physics is introduced to extend the predictive capability. For the J-th node, the enriched space-time shape function is given by

$$\tilde{\mathbf{N}}_J(\mathbf{x}, t) = \mathbf{N}_J(\mathbf{x}, t)\Phi_J(\mathbf{x}, t) \tag{26}$$

where the enrichment function is defined as

$$\Phi_J(\mathbf{x}, t) = \Phi(\mathbf{x}, t) - \Phi(\mathbf{x}_J, t_J) \tag{27}$$

Proper enrichment function can be selected by considering prior knowledge of problem physics. For HCF problems considered in this paper, a harmonic enrichment function is employed to capture the oscillating components in structural response, i.e.

$$\Phi_J(t) = \Phi(t) - \Phi(t_J) = \sin(\omega t) - \sin(\omega t_J) \tag{28}$$

where ω is the circular frequency of the imposed cyclic loading.

For convenience, we define $\mathbf{N} = \begin{bmatrix} \mathbf{N} & \tilde{\mathbf{N}} \end{bmatrix}$ and $\mathbf{d} = \begin{bmatrix} \mathbf{d} & \tilde{\mathbf{d}} \end{bmatrix}$ so that the approximation of unknown displacement field can be simply expressed as

$$\mathbf{u}(\mathbf{x}, t) = \sum_I \mathbf{N}_I(\mathbf{x}, t)\mathbf{d}_I \tag{29}$$

By substituting Eq. (29) into the weak form of TDG formulation, the discretized linear system of equations for the n-th space-time slab is obtained as

$$\mathcal{K}_n \mathbf{d}_n = \mathcal{F}_n \tag{30}$$

where the XTFEM stiffness matrix is given by

$$\begin{aligned}\mathcal{K}_n = &\int_{Q_n} \dot{\mathbf{N}}^T \rho \ddot{\mathbf{N}} dQ + \int_{Q_n} \dot{\mathbf{N}}_{,\mathbf{x}}^T \mathbf{C} \mathbf{N}_{,\mathbf{x}} dQ + \int_\Omega \dot{\mathbf{N}}^T (t_{n-1}^+) \rho \dot{\mathbf{N}}(t_{n-1}^+) d\Omega \\ &+ \int_\Omega \mathbf{N}_{,\mathbf{x}}^T (t_{n-1}^+) \mathbf{C} \mathbf{N}_{,\mathbf{x}}(t_{n-1}^+) d\Omega\end{aligned} \tag{31}$$

and the force vector is

$$\begin{aligned}\mathcal{F}_n = &\int_{(\gamma_t)_n} \dot{\mathbf{N}}^T \mathbf{t} d\gamma + \int_{Q_n} \dot{\mathbf{N}}^T \mathbf{f} dQ \\ &+ \left[\int_\Omega \dot{\mathbf{N}}^T (t_{n-1}^+) \rho \dot{\mathbf{N}}(t_{n-1}^-) d\Omega + \int_\Omega \mathbf{N}_{,\mathbf{x}}^T (t_{n-1}^+) \mathbf{C} \mathbf{N}_{,\mathbf{x}}(t_{n-1}^-) d\Omega \right] \mathbf{d}_{n-1}\end{aligned} \tag{32}$$

Since the space-time shape function is defined in a multiplicative form using Kronecker product, integrations over spatial domain can be done independently from the ones over the temporal domain. Thus, the XTFEM stiffness matrix can be expressed in a similar form, which is given by

$$\mathcal{K}_n = \boldsymbol{\Phi}_n \otimes \mathbf{K} + \boldsymbol{\Psi}_n \otimes \mathbf{M} \tag{33}$$

where $\mathbf{K}$ and $\mathbf{M}$ are respectively the spatial stiffness and mass matrices from the regular FEM, and the corresponding temporal matrices are

$$\boldsymbol{\Phi}_n = \int_{I_n} \dot{\mathbf{N}}_t^T \mathbf{N}_t dt + \mathbf{N}_t^T (t_{n-1}^+) \mathbf{N}_t (t_{n-1}^+) \tag{34}$$

$$\boldsymbol{\Psi}_n = \int_{I_n} \dot{\mathbf{N}}_t^T \ddot{\mathbf{N}}_t dt + \dot{\mathbf{N}}_t^T (t_{n-1}^+) \dot{\mathbf{N}}_t (t_{n-1}^+) \tag{35}$$

In the current implementation, Eqs. (34) and (35) are evaluated analytically. Based on the choices of the temporal shape function and enrichment function, the size of the resulting temporal matrices is $2n_t \times 2n_t$, which is significantly smaller than the spatial matrices.

2.4. Numerical implementation

The numerical implementation of XTFEM is summarized in Table 1. Computational cost analysis [35] showed that solving the stiffness matrix equations (line 8 in Table 1) is the most expensive part. It requires O(N^2) in memory for full storage of the stiffness matrix and O(N^3) in CPU time for solving the equations using direct solver, where N is the total number of DOFs.

Fig. 2 (a) and (b) illustrate sparsity patterns of the stiffness matrices that are obtained from regular FEM and XTFEM for the same number of spatial nodes. These patterns are generated based on a thin plate problem that will be described in Section 5.1. For spatial discretization, a 3D structured mesh (2880 8-node brick elements, 3965 nodes) is generated and leads to the banded pattern of stiffness matrix for regular FEM as shown in Fig. 2 (a). The number of spatial DOFs in regular FEM is $n_s = 11,895$. Due to the additional temporal dimension and enrichment that are introduced, the number of space-time DOFs in XTFEM is increased to $N = 2n_t \times n_s = 71{,}370$. Direct solution of the XTFEM stiffness equation leads to computational cost that is at least two orders of magnitude higher comparing to solving the regular stiffness equation in FEM, which becomes one of the critical barriers for its extensive and practical application.

Table 1
Implementation of XTFEM.

```
1     Initialization
2     Formulate spatial matrices $\mathbf{K}$ and $\mathbf{M}$
3     **DO** (loop over space-time slabs)
4         Integrate temporal matrices $\boldsymbol{\Phi}_n$ and $\boldsymbol{\Psi}_n$
5         Assemble space-time matrix $\mathcal{K}_n$
6         Calculate force vector $\mathcal{F}_n$
7         Apply boundary conditions
8         Solve $\mathcal{K}_n \mathbf{d}_n = \mathcal{F}_n$
9         Update and output solutions
10    **END DO**
```

3. Hybrid iterative/direct linear system solver

For each space-time slab, the stiffness matrix equation of XTFEM is given by

$$\mathcal{K}\mathbf{d} = \mathcal{F} \tag{36}$$

where $\mathcal{K} \in \mathbb{R}^{N\times N}$ is a non-symmetric sparse matrix, $\mathbf{d} \in \mathbb{R}^N$ and $\mathcal{F} \in \mathbb{R}^N$ are the unknowns and force vectors respectively.

In regular FEM, two types of linear systems solvers are generally employed to solve the stiffness matrix equations. The first type is the frontal direct sparse solver [36], which is robust and efficient particularly for equations with multiple right-hand-side vectors. The second type is the iterative solver [37,38], such as the Preconditioned Conjugate Gradient (PCG) method. Iterative solver is less robust for systems that are not well conditioned but more flexible. It can achieve higher efficiency than the direct solver for particular linear systems. In space-time FEM, iterative solvers such as Gauss-Jacobi and Gauss-Seidel methods have been adopted to reduce the computational cost [35,39–41], see Ref. [42] for a brief review.

In this work, by exploiting the unique block structure of XTFEM stiffness matrix, we propose a novel hybrid iterative/direct linear system solver that takes advantage of both the efficiency of iterative method and the robustness of direct method. In the proposed hybrid approach, the main solution algorithm is implemented by a nonstationary iterative solver, while the direct sparse solver is employed for preconditioning.

3.1. Preconditioning approach

3.1.1. Construction of the preconditioner

The basic idea of preconditioning is to modify the linear system to make it easier to solve while the saved computational effort outweighs the extra cost of preconditioning itself. With preconditioning, the original XTFEM stiffness matrix equation (36) is modified as

$$\mathcal{P}^{-1}\mathcal{K}\mathbf{d} = \mathcal{P}^{-1}\mathcal{F} \tag{37}$$

in which matrix $\mathcal{P}$ is the preconditioner.

In general, the choice and quality of a preconditioner greatly depend on the specific linear system. Therefore, the preconditioner $\mathcal{P}$ is built by approximating the XTFEM stiffness matrix $\mathcal{K}$. Zhang et al. [42] recently studied various space-time stiffness matrices obtained from different TDG formulations, including the single-field, the two-field and the currently employed enriched formulations. They showed that in space-time stiffness matrix $\mathcal{K}$, the component matrix $\boldsymbol{\Psi} \otimes \mathbf{M}$ is always singular due to the singularity of the temporal matrix $\boldsymbol{\Psi}$. On the other hand, the component matrix $\boldsymbol{\Phi} \otimes \mathbf{K}$ is nonsingular and dominates $\mathcal{K}$ for many applications. Based on those observations, a preconditioner was proposed by approximating the dominant component matrix $\boldsymbol{\Phi} \otimes \mathbf{K}$ as

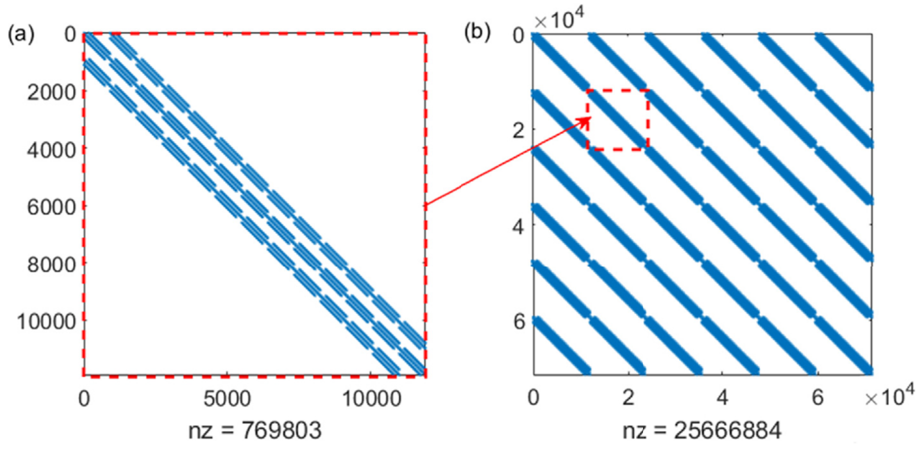

Fig. 2. Comparison among the sparse pattern of the stiffness matrices formed by (a) regular FEM and (b) XTFEM, dashed boxes indicate the size of the regular FEM stiffness matrix and *nz* represents the number of non-zero elements in the sparse matrix.

$$\mathcal{P} = \mathbf{\Phi} \otimes \mathbf{P} \tag{38}$$

in which matrix **P** was derived by an incomplete factorization of matrix **K** using incomplete LU factorization (ILU) or incomplete Cholesky factorization (ICHOL). It was shown that this type of preconditioner worked well for space-time FEM in the case of serial implementation [42].

To further improve the numerical efficiency of this type of preconditioning approach and develop parallel implementation, the dominant component matrix $\mathbf{\Phi} \otimes \mathbf{K}$ is employed in this paper as the preconditioner instead of formulating $\mathbf{\Phi} \otimes \mathbf{P}$ since the former already provides a good approximation of the XTFEM stiffness matrix. Therefore,

$$\mathcal{P} = \mathbf{\Phi} \otimes \mathbf{K} \tag{39}$$

With the use of Eq. (39), the process of evaluating **P** is avoided and computational effort is thus saved. Another important feature of the proposed preconditioner is that the direct sparse solver can now be introduced into the preconditioning operation, which improves its robustness and efficiency.

3.1.2. *Optimization of the preconditioning operation*

The preconditioning operation is generally expressed as

$$\mathbf{y} = \mathcal{P}^{-1}\mathbf{x} \tag{40}$$

in which **x** and **y** are vectors of size $N = 2n_t \times n_s$ with n_s being the dimension of spatial matrix.

Substituting the proposed preconditioner $\mathcal{P}$ to Eq. (40) yields

$$\mathbf{y} = (\mathbf{\Phi} \otimes \mathbf{K})^{-1}\mathbf{x} \tag{41}$$

The inverse of a Kronecker product is given by $(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$ if and only if both **A** and **B** are invertible. By using this property, Eq. (41) is further simplified as

$$\mathbf{y} = \left(\mathbf{\Phi}^{-1} \otimes \mathbf{K}^{-1}\right)\mathbf{x} \tag{42}$$

The inverse of the larger matrix $\mathcal{P}$ is now replaced by inverses of two smaller matrices **Φ** and **K**. The cost of inverting temporal matrix **Φ** is negligible since its size is very small ($2n_t \times 2n_t$). The spatial matrix **K** is the same as that in the regular FEM stiffness matrix, which is sparse, symmetric, and better conditioned. However, inverting it explicitly is still very expensive. To avoid that, we can further decompose the evaluation of Eq. (42) into the following two steps:

First, we introduce an intermediate vector **z** that can be solved from

$$\mathbf{z}^{(i)} = \mathbf{K}^{-1}\mathbf{x}^{(i)}, \quad i = 1, 2, \ldots, 2n_t \tag{43}$$

in which the size of $\mathbf{z}^{(i)}$ or $\mathbf{x}^{(i)}$ is n_s.

Then, the desired vector **y** is obtained by

$$\mathbf{y}^{(i)} = \sum_{j=1}^{2n_t} \phi_{ij}\mathbf{z}^{(j)}, \quad i = 1, 2, \ldots, 2n_t \tag{44}$$

in which ϕ_{ij} is the corresponding element of matrix $\mathbf{\Phi}^{-1}$.

In those two steps, solving Eq. (43) is the most computationally intensive step. In fact, it is equivalent to solving the corresponding regular FE stiffness matrix equations. Hence, by using the proposed method, computational cost of the preconditioning is reduced to the same order as for solving the corresponding stiffness matrix equations in linear static FEM analysis. Furthermore, Eq. (43) can be handled efficiently by many well-established direct sparse solvers for regular FEM. It is also possible to use commercial FE packages to solve Eq. (43).

In the current work, a parallel multi-frontal direct sparse solver named MUMPS [43,44] is employed to solve Eq. (43). In general, the direct sparse method solves a linear system by three main steps: (1) symbolic analysis; (2) numerical factorization, and (3) the final solution steps. Computational cost of the direct method is dominated by the first two steps. Once those two steps are completed, the final solution step can be performed separately and repeatedly for the case of multiple RHS vectors. Therefore, for linear elastodynamics the analysis and factorization steps of direct solver is performed only once during the first time increment. For nonlinear HCF simulations, the same solution strategy is employed unless the preconditioner $\mathbf{\Phi} \otimes \mathbf{K}$ no longer provides a good approximation of the XTFEM stiffness matrix $\mathcal{K}$ due to the progressive evolution of spatial matrix **K**. In that case, the convergence rate of the main iterative solver slows down and triggers the re-evaluation of the preconditioner. Thus, computational cost can be saved since the first two steps of the direct solver are minimized.

3.2. *Iterative solution approach*

Due to the asymmetry of XTFEM stiffness matrix, we employ the Generalized Minimum RESidual (GMRES) method [45] as the main iterative solver in the proposed hybrid approach. To accelerate matrix-vector multiplication, which is the most computationally expensive operation in GMRES algorithm, we propose an optimized implementation by utilizing the unique block structure of space-time matrix.

The matrix-vector multiplication is generally given by

$$\mathbf{y} = \mathcal{K}\mathbf{x} \tag{45}$$

Using the block structure of XTFEM stiffness matrix, Eq. (45) is rewritten as

$$\mathbf{y} = (\mathbf{\Phi} \otimes \mathbf{K} + \mathbf{\Psi} \otimes \mathbf{M})\mathbf{x} \tag{46}$$

Based on definitions given in Eq. (40), vectors **x** and **y** can be divided into smaller segments

$$\mathbf{x} = \left\{ \begin{array}{c} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(2n_t)} \end{array} \right\} \quad \text{and} \quad \mathbf{y} = \left\{ \begin{array}{c} \mathbf{y}^{(1)} \\ \mathbf{y}^{(2)} \\ \vdots \\ \mathbf{y}^{(2n_t)} \end{array} \right\} \tag{47}$$

in which the size of $\mathbf{x}^{(i)}$ or $\mathbf{y}^{(i)}$ is n_s.

By defining two intermediate vectors as

$$\mathbf{u}^{(i)} = \mathbf{K}\mathbf{x}^{(i)} \quad \text{and} \quad \mathbf{v}^{(i)} = \mathbf{M}\mathbf{x}^{(i)}, \quad i = 1, 2, ..., 2n_t \tag{48}$$

we can compute the desired vector **y** as

$$\mathbf{y}^{(i)} = \sum_{j=1}^{2n_t} \Phi_{ij}\mathbf{u}^{(j)} + \sum_{j=1}^{2n_t} \Psi_{ij}\mathbf{v}^{(j)}, \quad i = 1, 2, ..., 2n_t \tag{49}$$

where Φ_{ij} and Ψ_{ij} are components of the temporal $\mathbf{\Phi}$ and $\mathbf{\Psi}$ matrices.

It can be shown that direct evaluation of the matrix-vector product in Eq. (45) requires approximately $2N^2$ arithmetic operations, including $N(N\text{-}1)$ additions and N^2 multiplications, and $O(N^2)$ memory assuming a dense matrix storage scheme, where $N = 2n_t n_s$ is the size of space-time matrix. In the optimized implementation, the matrix-vector products in Eq. (48) require $8n_t n_s^2$ arithmetic operations and Eq. (49) requires only $O(n_s)$ operations. The space and time matrices in Eqs. (48) and (49) require a storage cost of $O(2n_s^2)$. Therefore, both operations and storage cost of the optimized implementation are reduced to only $1/n_t$ and $1/(2n_t^2)$ of the direct evaluation method, respectively.

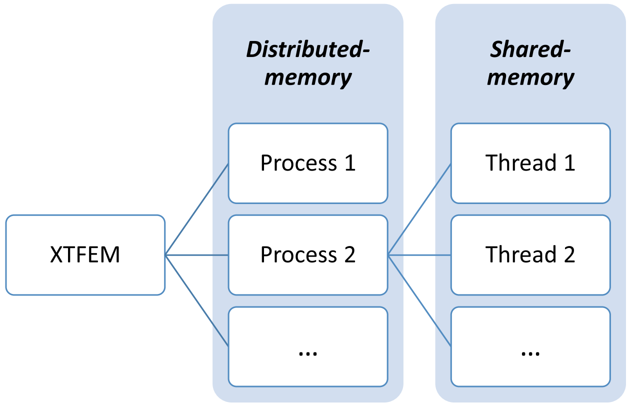

Fig. 3. A two-level hierarchy of two types of parallelism.

3.3. HPC implementation

To further extend the capability of XTFEM on handling large-scale problems using HPC platforms, we incorporate high-performance parallel computing techniques to accelerate the proposed hybrid iterative/direct linear system solver. A hybrid of parallelisms, i.e. the distributed-memory and the shared-memory parallelisms, is exploited in this work. It forms a hierarchy of those two types of parallelism as shown in Fig. 3. In this way, the parallel computing hardware can be used more efficiently.

3.3.1. Distributed-memory parallelism

The distributed-memory parallelism arises from partitioning the computational domain of XTFEM. The space-time finite element mesh is first partitioned into smaller subdomains along the element boundaries based on its spatial discretization. Computing tasks on each subdomain are then handled by a specific process as shown in Fig. 4. Data on the interface nodes between the subdomains are transferred through communications among processes using the *Message Passing Interface* (MPI) protocol [46].

To achieve optimal parallel efficiency, partitioning of the computational domain should satisfy the following three objectives: First, computing load should be balanced by assigning different number of elements to each subdomain based on the amount of computing resources available to the corresponding process. Second, the number of interface nodes among the subdomains should be minimized to reduce the amount of data communication between processes. Third, the algorithm for domain partitioning itself should be fast and efficient. Among those objectives, the second is particularly important for the proposed hybrid iterative/direct linear system solver. A good partitioning not only significantly reduces the amount of data communications of parallel matrix-vector multiplication, which leads to higher efficiency of the GMRES iteration, but also produces a high-quality fill-reducing ordering that leads to a higher degree of concurrency for the factorization phase of

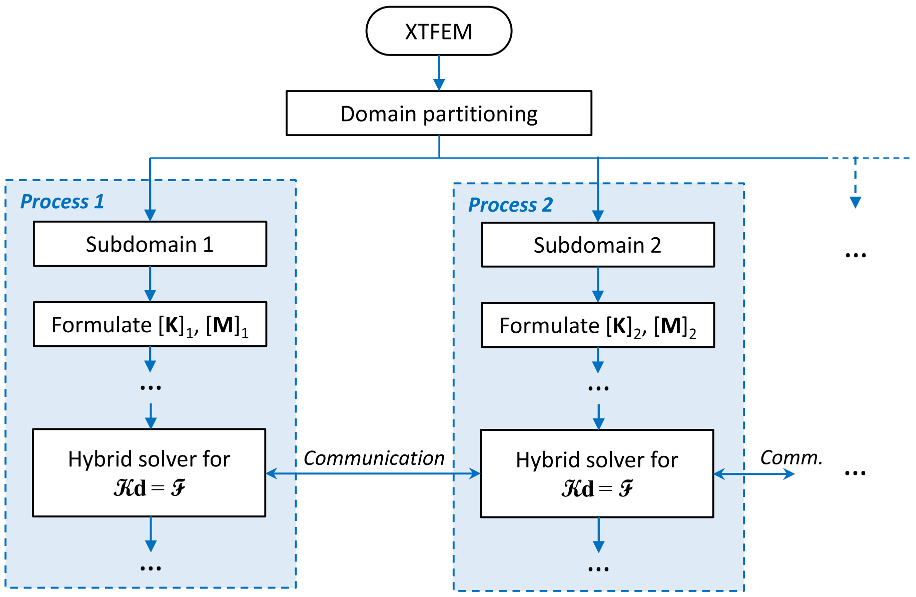
XTFEM
Domain partitioning
Process 1
Subdomain 1
Formulate [K]$_1$, [M]$_1$
...
Hybrid solver for $\mathcal{K}\mathbf{d} = \mathcal{F}$
...
Communication
Process 2
Subdomain 2
Formulate [K]$_2$, [M]$_2$
...
Hybrid solver for $\mathcal{K}\mathbf{d} = \mathcal{F}$
...
Comm.
...
...

Fig. 4. Distributed-memory parallelism of XTFEM.

the direct solver [47]. As such, domain partitioning is critical to the parallel implementation of the proposed hybrid solver. In this work, we employ a multilevel partitioning scheme developed by Karypis and Kumar [48,49], also known as the METIS package. Fig. 5 shows an example of domain partitioning by using the METIS package. The finite element mesh of a thin plate is divided into 48 subdomains, which are represented by different colors.

3.3.2. Shared-memory parallelism

As illustrated in Fig. 6, the shared-memory parallelism of XTFEM is exploited from three computationally-intensive tasks that are assigned to individual MPI process: (1) the formulation of local spatial stiffness and mass matrices, (2) the sparse matrix-vector multiplication (SpMV) subroutine in the iterative part of the hybrid solver, and (3) the double precision general matrix multiplication (DGEMM) subroutine in the direct part of the hybrid solver.

The spatial matrices are formed in two steps. First, the element matrices are calculated and assembled into a coordinate (COO) format sparse matrix. Then, the COO format matrix is converted to the compressed sparse row (CSR) format matrix, which is the input format used for the hybrid solver. Details of this algorithm are described in Ref. [35]. Among the above two steps, formulating element matrices is the most time-consuming and well-suited for the single instruction, multiple data (SIMD) multithreading parallelization. Thus, OpenMP [50] multithreading technique is employed to accelerate this step based on its intrinsic element-wise parallelism.

In terms of computational cost, the iterative and direct parts of the hybrid solver are respectively dominated by the SpMV and the DGEMM subroutines [51]. Those subroutines are well-suited for SIMD parallelization based on either row-wise or column-wise parallelism. However, parallel efficiency of those two subroutines also depends on hardware architecture. To analyze the efficiency of SpMV and DGEMM subroutines, we introduce the arithmetic to memory ratio (AMR), which is defined by the number of arithmetic operations to the amount of memory accesses required by an algorithm. The theoretical upper bound of the AMR of SpMV is 2:1 (flops/bytes) [51]. SpMV is a memory-intensive subroutine in the sense that its computing time is bounded by memory bandwidth of hardware. In contrast, DGEMM is a computing-intensive subroutine with an AMR of n:1, where n is the size of the dense frontal matrix [51]. Currently, HPC platforms typically have large AMR values and the trend of architecture evolves towards higher AMR. For example, the peak performance of top supercomputer in the past decade (see Table 2) increased 86 times while the memory and network bandwidth improved only 5.3 and 5.8 times, respectively. Similarly, the flops to bytes ratio, i.e. AMR, of NVIDIA's TESLA general-purpose GPUs (GPGPU) increased 11 times since 2008 (see Table 3). Therefore, the SpMV subroutine is less efficient on most HPC platforms compared to the DGEMM subroutine. From this perspective, the proposed hybrid iterative/direct solver utilizes HPC platforms more efficiently than the conventional iterative solvers.

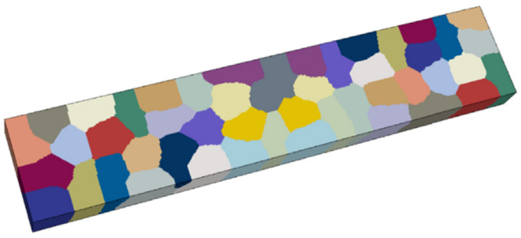

Fig. 5. Domain partitioning by METIS for finite element mesh of a thin plate, subdomains are indicated by different colors.

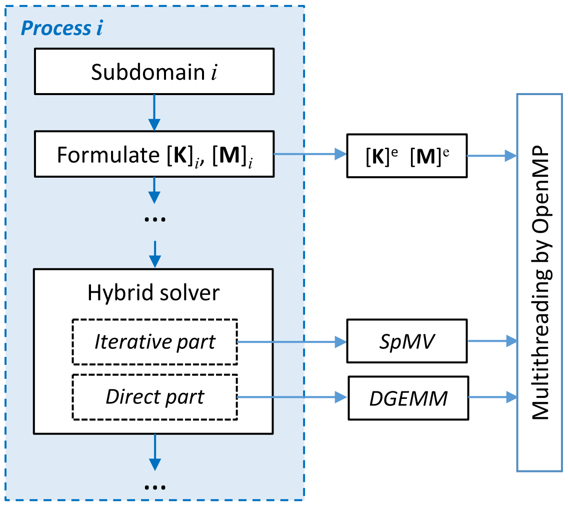

Fig. 6. Shared-memory parallelism of XTFEM.

4. Two-scale fatigue damage model

To predict nonlinear material behavior under HCF loading conditions, XTFEM is further coupled with a CDM-based constitutive model. An important feature of the CDM model is that its kinetic law of damage evolution is constructed as a function of the incremental constitutive variables, such as stress, strain or effective plastic strain, rather than number of loading cycles, which is critical for multiaxial and other complex loading conditions. A CDM-based two-scale fatigue damage model is developed by Lemaitre and Doghri [27,28] and Desmorat et al. [29]. It has been successfully applied for HCF problems under uniaxial, biaxial, random and thermal cyclic loading histories [35,57]. The two-scale model is adopted in current work. However, it should be noted that many other microstructure-based material damage models [58–61] can also be integrated.

4.1. Damage model and coupling with XTFEM

A sketch of the two-scale damage model is illustrated in Fig. 7. In this model, mesoscale material behavior is assumed to be elastic since the amplitude of HCF loading is moderate. Nonlinear behaviors such as plasticity and damage are modelled at the microscale that represent the scale of defects, e.g. microcracks and microvoids. These two scales are bridged by the modified Eshelby-Kröner localization law [30,31].

Table 2
Performance evolution of supercomputers (2009–2018).

Name	Year	Theoretical peak performance (Tflops)	CPU memory bandwidth (GB/s)	Network bandwidth (GB/s)
Summit [52]	2018	200,795	135.0	23
Titan [51, 53]	2012	27,113	52.0	8
K computer [54]	2011	11,280	64.0	10
Jaguar [55]	2009	2331	25.6	4

Table 3
Performance evolution of NVIDIA TESLA GPGPU [56] (2008–2018).

Model	Year	Double precision performance (Gflops)	Memory bandwidth (GB/s)	flops/bytes
V100	2018	7834	897	8.7
P100	2016	4763	732	6.5
K40	2013	1430	288	5.0
C2090	2011	656	177	3.7
S1070	2008	78	102	0.8

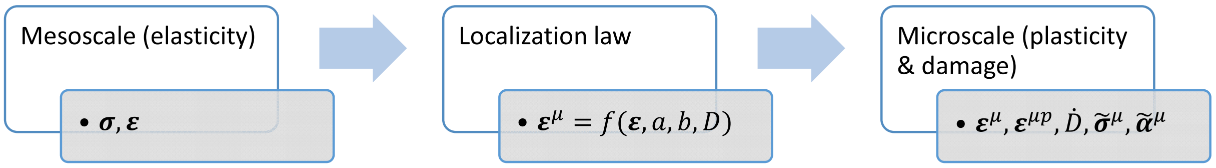
Mesoscale (elasticity)
• $\boldsymbol{\sigma}, \boldsymbol{\varepsilon}$
Localization law
• $\boldsymbol{\varepsilon}^{\mu} = f(\boldsymbol{\varepsilon}, a, b, D)$
Microscale (plasticity & damage)
• $\boldsymbol{\varepsilon}^{\mu}, \boldsymbol{\varepsilon}^{\mu p}, \dot{D}, \tilde{\boldsymbol{\sigma}}^{\mu}, \tilde{\boldsymbol{\alpha}}^{\mu}$

Fig. 7. Sketch of the two-scale damage model.

The implicit single-step Backward Euler integration scheme proposed by Desmorat et al. [29] is employed to solve the coupled plasticity-damage equations at the microscale of the two-scale model. At the mesoscale, the corresponding element will be deleted once the damage value reaches the critical damage D_c at crack initiation. Since HCF life is dominated by crack initiation, the damage increment is small at each loading cycle. To avoid the cumbersome iterations in the Newton's method, the damage is assumed a constant per time increment [12]. Detailed formulation of this model can be found in Refs. [12,27–29].

Fig. 8 shows the flowchart for the implementation of the two-scale damage model and its coupling with XTFEM. In terms of computational cost, solving the nonlinear constitutive model is a major component in regular FEM implementation due to the complex constitutive update algorithms and repeated evaluations on all the material or quadrature points. In the current computational framework, this implementation is more expensive than in regular FEM since both spatial and temporal resolutions are needed for capturing HCF events. In terms of the spatial resolution, refined mesh and more material points are placed at high stress/strain gradient regions. On the other hand, a good temporal resolution typically requires more than 200 increments per loading cycle to ensure convergence of solution to the two-scale damage model [35]. To satisfy those requirements, the two-scale damage model is embedded in the nested temporal and spatial loops as illustrated in Fig. 8. Consequently, solving the two-scale damage model adds to significant computational cost, which motivates the development of high-performance computing algorithm.

4.2. Parallel implementation of the constitutive model

The constitutive updates at different quadrature points are independent of each other. As such, evaluations of the two-scale damage model on all quadrature points can be carried out simultaneously. This computing task is well-suited for the SIMD type of parallelism and can be accelerated efficiently by using multithreading parallel computing technique. In accordance with the previously developed parallel computing framework for XTFEM, a hierarchy of parallelisms is also established for the two-scale damage model. Similarly, the first-level of parallelism arises from the domain partitioning. However, communication among processes is not required since it is a SIMD-type computing task. At the second-level of parallelism, an element-wise multithreading is employed. In the current work, we have developed both OpenMP (CPU) and CUDA (GPU) versions of the parallel implementation of the two-scale damage model.

The OpenMP version of the two-scale damage model is shown in Table 4. The first and last lines are the OpenMP directives. Without those directives, it reduces to a serial code. Note that only the outer loop over the material points on subdomain is expanded by OpenMP directive. Within each time increment, the elastic prediction, plastic correction and damage update stages of the two-scale model are evaluated. In Ref. [35], an optimized GPU algorithm based on CUDA is developed to accelerate the two-scale damage model. This GPU algorithm is adopted in the current framework, which is shown in Table 5. More details of this GPU algorithm can be found in Ref. [35].

Theoretically, OpenMP and CUDA versions of the two-scale damage

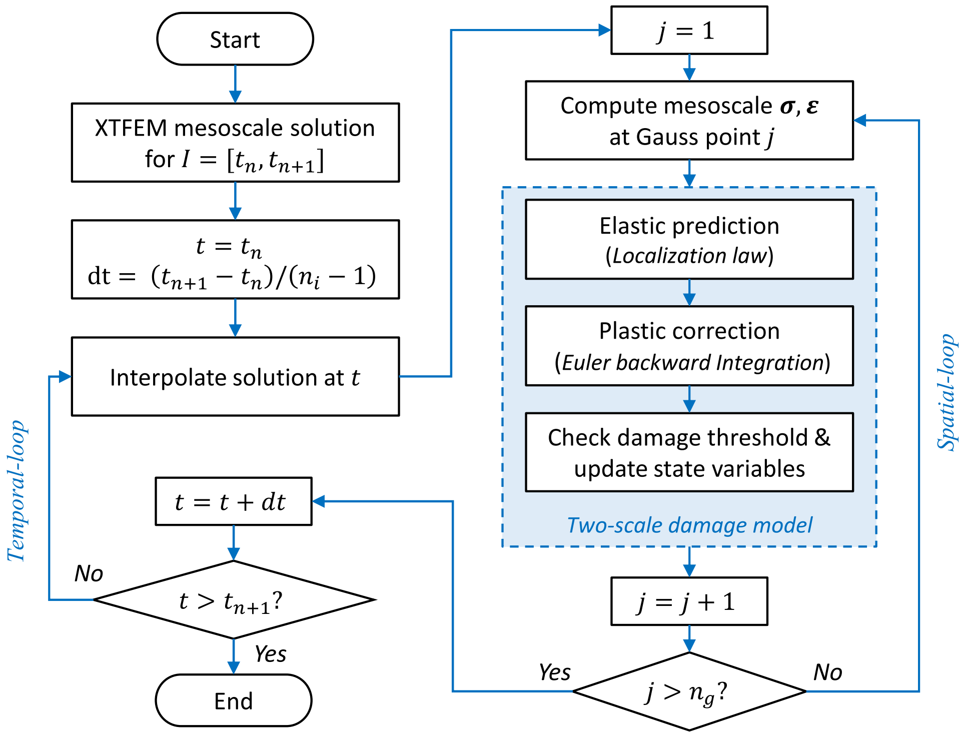
Start
XTFEM mesoscale solution for $I = [t_n, t_{n+1}]$
$t = t_n$
$\mathrm{dt} = (t_{n+1} - t_n)/(n_i - 1)$
Interpolate solution at t
Temporal-loop
$t = t + dt$
No
$t > t_{n+1}$?
Yes
End
$j = 1$
Compute mesoscale $\boldsymbol{\sigma}, \boldsymbol{\varepsilon}$ at Gauss point j
Elastic prediction (Localization law)
Plastic correction (Euler backward Integration)
Check damage threshold & update state variables
Two-scale damage model
Spatial-loop
$j = j + 1$
Yes
$j > n_g$?
No

Fig. 8. Flowchart for the implementation of the two-scale damage model and its coupling with XTFEM (n_i is the number of temporal interpolation points and n_g is the number of spatial Gauss integration points).

Table 4
OpenMP version of the two-scale damage model.

```
1   !$OMP PARALLEL DO
2   DO (loop over local material points)
3       DO (loop over time increments)
4           Elastic prediction
5           Plastic correction
6           Damage update
7       END DO
8   END DO
9   !$OMP END PARALLEL DO
```

model can be used simultaneously. In practice, however, it makes the load balancing between CPUs and GPUs too complicated. In addition, it is shown that a single GPU is typically orders of magnitude faster than a single core of CPU [35]. Therefore, only the CUDA version is employed when GPUs are available, otherwise the OpenMP version is used.

5. Results and discussion

5.1. Plate subjected to cyclic load

5.1.1. Problem statement

The geometric dimensions and boundary conditions of the plate problem are illustrated in Fig. 9. The plate is fixed at the left end and a uniformly distributed cyclic load $p(t) = 100\sin(40\pi t)$ MPa is applied on the right end. The material of the plate is assumed to be isotropic elastic with Young's modulus $E = 200$ GPa, Poisson's ratio $\nu = 0.3$ and mass density $\rho = 7860$ kg/m^3.

The spatial domain of the plate is discretized by the same structured mesh with 8-node linear hexahedral elements with full integration (C3D8). For temporal discretization, a space-time slab size of 0.25s was employed along with an enrichment function of $\Phi(t) = \sin(40\pi t)$. The characteristic length of the element is 0.5 mm, which leads to 2880 elements and 3965 nodes.

5.1.2. Performance of the hybrid solver

Extensive demonstrations of the accuracy of XTFEM can be found in Ref. [12]. To demonstrate the performance of the proposed hybrid solver, we evaluate the CPU time and memory usages for solving a single space-time slab. The performance is assessed in three steps. First, we check the single-thread performance without using the parallel computing capability. Second, the computational complexity of the accelerated XTFEM is compared with regular FEM. Lastly, the parallel performance is demonstrated.

To study the single-thread performance of the hybrid solver, we gradually refine the spatial discretization of the plate until it reaches about half million space-time DOFs. During the refinement process, the aspect ratio of the element remains the same to ensure mesh quality. For

Table 5
CUDA version of the two-scale damage model.

Host program

```
1   Allocate memory on device
2   Copy data from host to device
3   Launch device program
4   Copy data from device to host
5   Clear device memory
```

Device program

```
1   Obtain thread index (0-based)
2   IF (thread index < number of local material points)
3       Copy data from global memory to registers
4       DO (loop over time increments)
5           … (same with the OpenMP version)
6       END DO
7       Copy data from registers to global memory
8   END IF
```

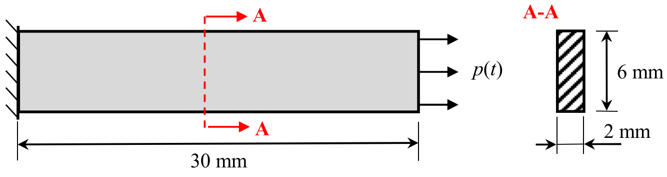

Fig. 9. The geometric dimensions and boundary conditions of the thin plate problem.

comparison purpose, we also employ the sparse direct and iterative solvers to solve the XTFEM stiffness matrix equations. The direct solver is the same as the direct part of the hybrid solver, i.e. the serial-version of MUMPS solver. The iterative solver [35] employs the commonly used preconditioner based on incomplete LU factorization and the GMRES algorithm. A desktop workstation equipped with the Intel Xeon E5-2623v3 CPU (3 GHz) and 32 GB RAM is used for the single-thread performance testing. Fig. 10 provides a comparison on the performances of those linear system solvers. According to Fig. 10(a), the hybrid solver achieves a time complexity of O($N^{1.4}$) while the time complexities of the direct and iterative solvers are respectively O($N^{2.6}$) and O($N^{1.8}$). For the cases of $N > 10^4$, the hybrid solver is at least 1–2 orders of magnitude faster than the others. The memory costs are shown in Fig. 10(b). In terms of storage complexity, the iterative solver achieves the best performance of O($N^{1.4}$). The direct solver shows the worst performance of O($N^{2.2}$). The hybrid solver demonstrates a complexity of O($N^{1.6}$), which is slightly higher than that of the iterative solver. However, in terms of the actual memory usage, the hybrid solver is 1–2 orders of magnitude lower than the other solvers since explicit formulation of the space-time matrices is avoided.

Next, we compare the performance of the accelerated XTFEM with the regular FEM that employs the implicit Newmark-β time integration method. The Newmark-β method forms linear systems of equations with an effective stiffness matrix, which is given by

$$\mathbf{K}_{\text{eff}} = \mathbf{K} + \frac{1}{\beta \Delta t^2}\mathbf{M} \tag{50}$$

where the parameter $\beta = 0.25$. To solve the Newmark-β linear systems of equations, the direct sparse solver, i.e. the serial-version of MUMPS solver, is employed. Performance of the Newmark-β method is quantified by both the CPU time and memory usage at the initial time increment. The testing hardware remains the same as in the previous single-thread tests.

Performance comparison between the XTFEM and Newmark-β method is shown in Fig. 11. As a reminder, the system matrix of XTFEM is 6 times larger than that of the Newmark-β method for the same spatial discretization. Thus, the performance metrics are plotted with respect to the number of spatial nodes instead of the number of DOFs. It shows that the time usages are on the same level while the memory costs are almost identical. Although Newmark-β method is slightly faster, its time complexity of O($N^{1.64}$) is higher than XTFEM, which is only O($N^{1.37}$). In other words, XTFEM will eventually outperform the Newmark-β method for a single time increment when the number of nodes is large enough. In addition, XTFEM typically employs a time increment size that is several orders of magnitude larger than that of standard implicit/explicit FEM [12,42,62].

Lastly, we check the parallel performance of XTFEM. Like the single-thread test, a set of spatial mesh grids is created by refining the element size, which is summarized in Table 6. The largest case leads to over 100 *million* space-time DOFs. It is noted that all the parallel performance tests presented in this paper are conducted on the Lonestar-5, a supercomputer from the Texas Advanced Computing Center (https://www.tacc.utexas.edu/systems/lonestar). Instead of using CPU time, the elapsed time or the wall-clock time usage is employed to measure the parallel performance of XTFEM.

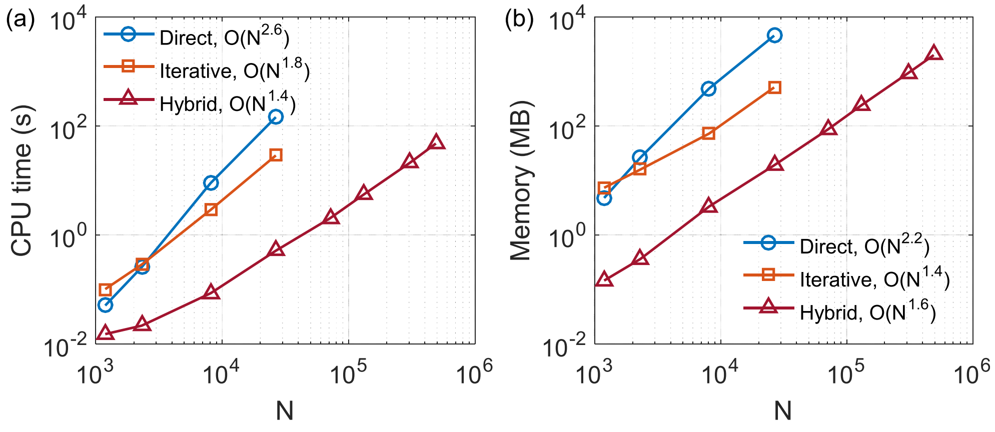

Fig. 10. Comparison of computational performance among the direct, iterative and hybrid linear solvers, (a) CPU time usage and (b) memory cost.

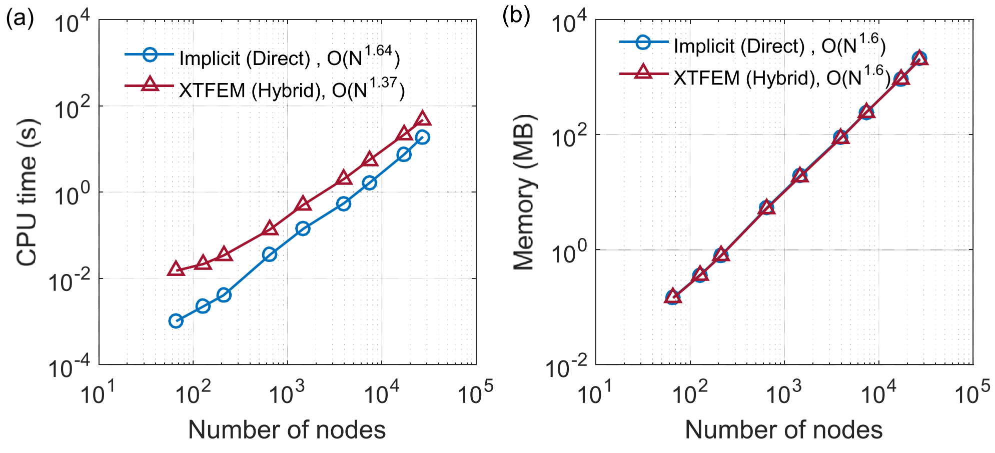

Fig. 11. Comparison of performance between the implicit Newmark-β method and the XTFEM, (a) CPU time usage and (b) memory cost.

Table 6
Mesh grids for plate problem.

No.	Element size (mm)	# Elements	# Nodes	# DOFs
1	0.5	2880	3965	71,370
2	0.25	23,040	27,225	490,050
3	0.125	184,320	200,753	3,613,554
4	0.0625	1,474,560	1,539,681	27,714,258
5	0.05	2,880,000	2,981,561	53,668,098
6	0.04	5,625,000	5,783,451	104,102,118

As discussed in Section 3.1.2, for linear elastodynamics, analysis and factorization phases of the direct solver are performed only once to save computational cost. Therefore, we further decompose the wall-clock time usage into two parts. The first part is the time usage by the preconditioner, which is mainly contributed by the analysis and factorization phases of the MUMPS solver. The second part is the time usage by the GMRES solver for each space-time slab, which is mainly contributed by the matrix-vector multiplication and the solution phase of the MUMPS solver.

Performance of the preconditioner evaluation is shown in Fig. 12. It

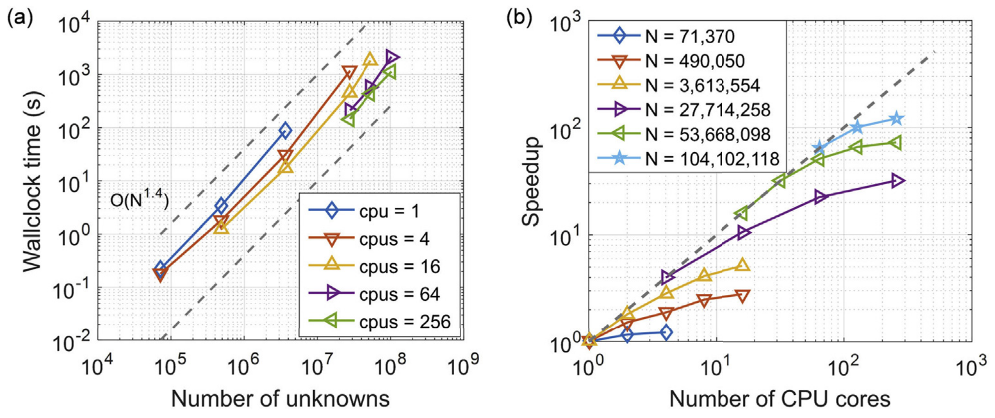

Fig. 12. Performance of preconditioner evaluation by direct solver: (a) Wall-clock time usage vs. number of unknowns and (b) Speedup vs. number of CPU cores.

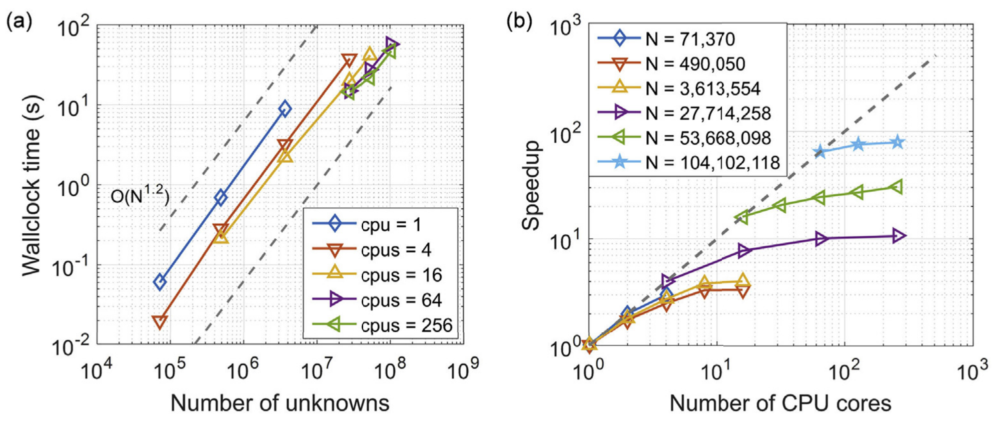

Fig. 13. Performance of the GMRES solver: (a) Wall-clock time usage vs. number of unknowns and (b) Speedup vs. number of CPUs.

can be seen from Fig. 12(a) that the overall time complexity for different number of CPU cores is around O($N^{1.4}$). The speedup ratios are shown in Fig. 12(b) with dashed line representing the ideal speedup. For all the cases, the speedup ratio increases efficiently with the increasing number of CPU cores. The speedup curves deviate from the ideal case when the number of CPUs get large since the cost of communication becomes more significant and overwhelms that of computation.

Fig. 13 shows the performance of the second part – the GMRES solver in terms of computational cost of each time step. It can be seen from Fig. 13(a) that the time complexity of the GMRES solver is only O($N^{1.2}$). The corresponding speedup ratio is shown in Fig. 13(b), where a similar trend to that of the preconditioning part can be observed. However, parallel efficiency of the GMRES part is lower than that of the preconditioning part. A comparison of the time usages between those two parts for different number of unknowns is shown in Fig. 14. We find that the GMRES part contributes to less than 20% of the total time for $N = 71,370$, which corresponds to the coarsest mesh. For the finest mesh, the preconditioner contributes to more than 97% of the total time of the hybrid solver. Therefore, a significant amount of computational cost can be saved by using the solution strategy proposed in Section 3.1.2, which minimizes the preconditioner evaluation.

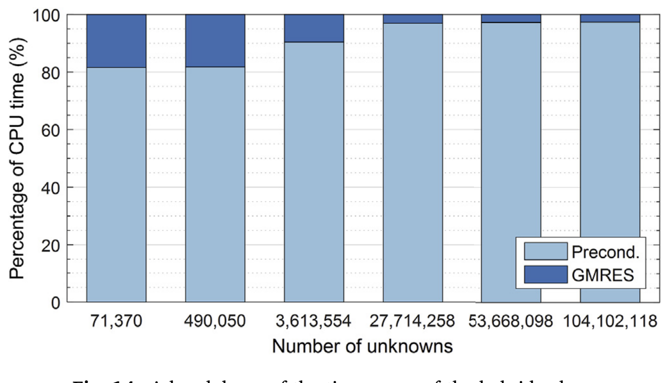

Fig. 14. A breakdown of the time usage of the hybrid solver.

The above performances are obtained by using the first-level parallelism (MPI) only. To demonstrate the full capability of the hybrid parallel computing using both MPI and OpenMP, we test the case with 54 *million* unknowns with 16 MPI processes that are distributed to 16 compute nodes. Note that on Lonestar-5, each compute node has 24 CPU cores. In other words, the maximum number of OpenMP threads for each MPI process is 24. In addition, 16 compute nodes on Lonestar-5 are configured with 1 Tesla K40 GPU from NVIDIA. The GPUs are not utilized in this test. The performance of such a test configuration is summarized in Table 7. It shows that with the second-level parallelism additional speedup can be achieved for both parts (preconditioning and GMRES) of the hybrid solver. For this specific case, a total of 4–8 OpenMP threads per MPI process yields the optimal efficiency.

As a brief summary, the serial version of the proposed hybrid solver is at least 1–2 orders of magnitude better than conventional solvers in terms of both CPU time and storage. The parallel version performs well and efficiently handles problems with over 100 *million* DOFs using 64 CPU cores. Therefore, cost of solving XTFEM stiffness matrix equations is significantly reduced.

5.2. HCF of a notched plate

5.2.1. Problem statement

The geometric dimensions and boundary conditions of the notched plate are illustrated in Fig. 15. The geometry is designed according to the widely employed single edge notched tension (SENT) experiment for fatigue studies of metals and alloys [63]. The specimen is fixed at its left end. A cyclic load $P(t)$ is applied on the right end. The material properties are given as Young's modulus $E = 197$ GPa, Poisson's ratio $\nu = 0.3$ and mass density $\rho = 7860\ \mathrm{kg/m^3}$. Other parameters associated with the two-scale damage model are given in Table 8, which are based on [12,29] and verified with experiment results in Ref. [64].

Table 7
Hybrid MPI/OpenMP parallel performance of the hybrid solver.

# DOFs	# MPI processes[a]	# OpenMP threads per process	Preconditioner		GMRES	
			Time (s)	Speedup	Time (s)	Speedup
53,668,098	16	1	1467.7	1.0	42.0	1.0
		2	860.2	1.7	31.6	1.3
		4	559.5	2.6	25.6	1.6
		8	443.1	3.3	25.7	1.6
		16	390.5	3.8	25.4	1.7
		24	407.5	3.6	24.0	1.8

[a] One MPI process per compute node.

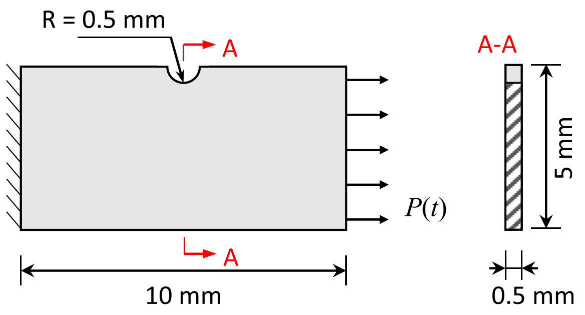

Fig. 15. Geometric dimensions and boundary conditions of the single edge notched plate.

Table 8
Parameters of the two-scale damage model for type 304L stainless steel.

C_y (MPa)	σ_u (MPa)	ε_{pd}	σ_f^{∞} (MPa)	h	D_c	S	s
1740	577	0.08	180	0.2	0.3	0.5	0.5

5.2.2. *Results of HCF simulations*

A mesh convergence study is first conducted to determine the spatial discretization. The notched plate is discretized by standard 8-node brick elements. To reduce computational cost, an unstructured, gradient spatial mesh is created. A sample spatial mesh with element size 0.1 mm at notch root is illustrated in Fig. 16(a). Six mesh densities of element size 0.2, 0.1, 0.05, 0.025, 0.0125 and 0.01 mm are employed near the notch root and along the estimated path of crack propagation. A fully-reversed cyclic load $P(t) = 100\sin(40\pi t)$ MPa is applied. Fig. 16(b) shows that the maximum von-Mises stress converges to 330 MPa in the case of element size = 0.0125 mm. The corresponding mesh is employed for the subsequent HCF simulations, which leads to a discretization of 108,080 elements, 113,160 nodes, and 2,036,880 DOFs. The time step size or equivalently the temporal size of space-time slab is initially set to 100T, in which T is the period of the loading cycle. After crack initiation, the time step size is reduced to 10T to capture crack propagation.

Results of HCF simulation under cyclic load $P(t) = 62.5\sin(40\pi t)$ MPa are shown in Fig. 17. Fatigue crack initiates at the notch root and propagates to the half width of the specimen. Fig. 17(b) shows the microscale nonlinear damage accumulation at the notch root. An exponential crack growth is captured and shown in Fig. 17(c), which is consistent with the trends that are observed in the HCF experiments. The number of cycles for crack initiation and fatigue failure are 146,219 and 155,320, respectively. Most of the fatigue life is consumed by crack initiation, which is a typical HCF behavior.

Series of such HCF simulations are performed by varying the loading amplitude. Results of those simulations are presented in Fig. 18 in the

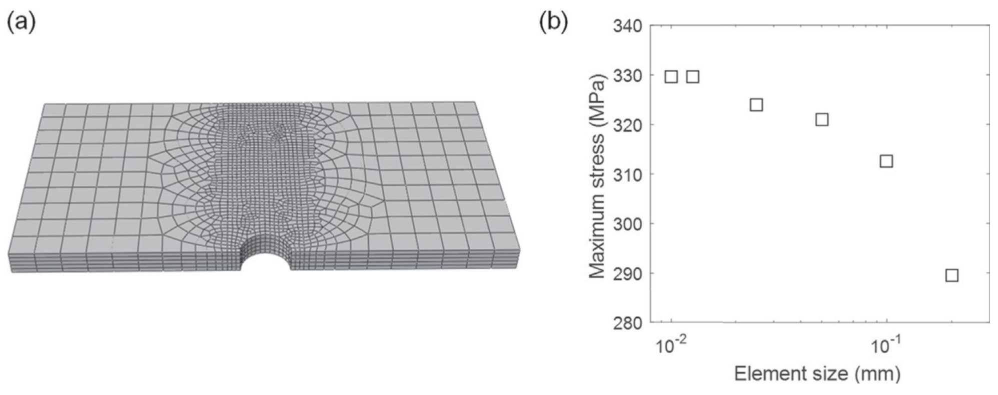

Fig. 16. Mesh convergence study: (a) a sample spatial mesh with element size = 1 mm and (b) maximum stress versus element size at the notch root.

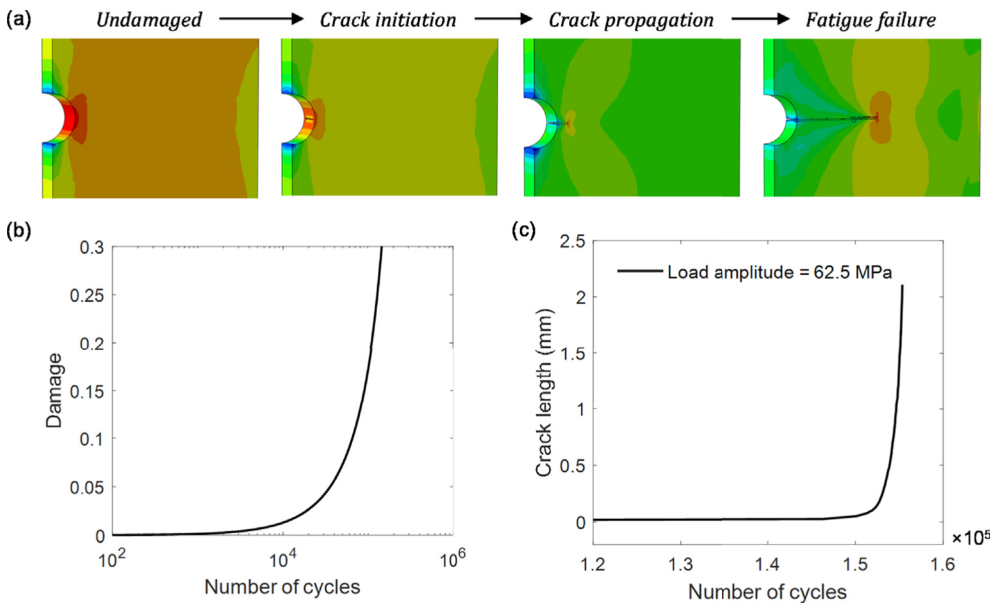

Fig. 17. Results of HCF simulation: (a) the processes of crack initiation and propagation (colored by von-Mises stress in logarithmic scale), (b) damage accumulation at the notch root, and (c) crack length vs. number of cycles.

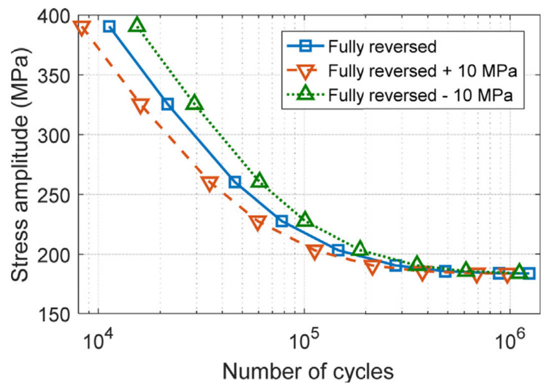

Fig. 18. S-N curves obtained from series of HCF simulations on the notched specimen.

form of *S-N* curve. It is shown that more than 1 million cycles are simulated by the proposed framework. To the best of our knowledge, such direct 3D HCF simulations have not been reported elsewhere. Furthermore, both tensile and compressive mean stress effects are simulated to demonstrate the capability of the proposed framework in terms of handling complex fatigue loading conditions. For the simulations on mean stress effects, a constant load is also imposed on the traction surface of the notched specimen. The total loading history is then expressed as $P(t) = P_0 \sin(40\pi t) + P_1 H(t)$ MPa, in which $H(t)$ is the Heaviside function, P_0 and P_1 are the amplitudes of cyclic and constant loads, respectively. Here we consider two scenarios with constant loading amplitudes $P_1 = \pm 10$ MPa, which represent tensile and compressive mean stresses respectively. The *S-N* curves obtained from both scenarios are plotted in Fig. 18 and compared with the fully-reversed case. The mean stress effects on fatigue life is effectively captured by the two-scale damage model with a microdefects closure parameter $h = 0.2$. It can be clearly observed that tensile mean stresses reduce fatigue life while compressive mean stresses extend fatigue life. Hence, a potential application of the proposed framework along this line is to study the effects of mean stress on fatigue life, which can be induced by surface treatments [65,66].

5.2.3. Parallel performance of the two-scale damage model

Fig. 19(a) shows the wall-clock time usage by the damage model for 100 loading cycles versus number of Gauss points for different number of CPU cores. The CPUs used are Intel Xeon E5-2690 v3 (2.6 GHz). The computational complexity of the damage algorithm is O(*N*), where *N* is the number of Gauss quadrature points. Parallel efficiency of the OpenMP version damage code is illustrated in Fig. 19(b) and shows an optimal speedup. Similarly, performance of the CUDA version damage code is illustrated in Fig. 20. The GPUs employed is NVIDIA TESLA K40. The maximum number of GPUs employed for testing is 4. In Fig. 20(b), the speedup of the CUDA version damage code shows a low performance for the coarse mesh ($N = 14{,}592$) when number of GPUs > 2, which is caused by insufficient GPU occupancy. Performance of the CUDA version improves with the increasing number of Gauss points and reaches an optimal efficiency for the fine mesh.

Based on the performance demonstrated so far, we conclude that the developed framework is efficient and has a good parallel scalability,

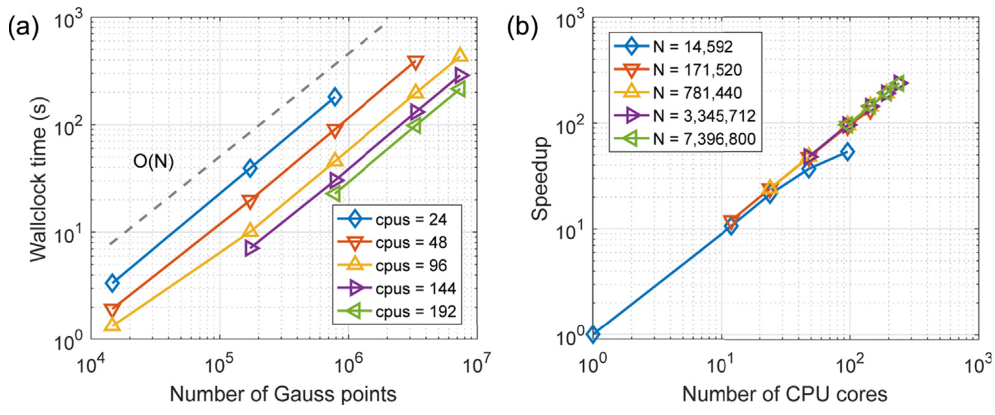

Fig. 19. Parallel performance of the two-scale damage model on CPUs: (a) wall-clock time vs. number of Gauss points for different number of CPU cores (shown in the inset box), (b) speedup vs. number of CPU cores for different number of Gauss points (shown in the inset box).

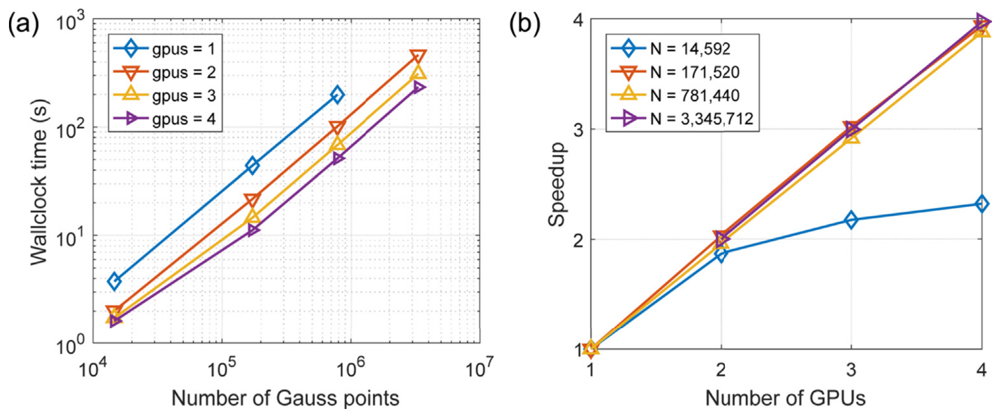

Fig. 20. Parallel performance of the two-scale damage model on GPUs: (a) wall-clock time vs. number of Gauss points for different number of CPUs (shown in the inset box), (b) speedup vs. number of GPUs for different number of Gauss points (shown in the inset box).

thereby enabling the large-scale applications.

5.3. Biaxial HCF of a cruciform specimen

5.3.1. Problem statement

Based on the series of biaxial HCF experiments conducted by Poncelet et al. [67] and Cláudio et al. [68], biaxial HCF simulations are performed to further demonstrate the capability of the developed framework on handling complex multiaxial loading conditions. The biaxial specimen considered here is adopted from Ref. [67]. Geometry and dimensions of the specimen are provided in Fig. 21. The cruciform specimen has a thinned circular region located at its center, which serves as a stress concentration zone for fatigue damage accumulation and crack initiation. Fatigue loadings are imposed along both x and y directions. The edges opposite to the traction surfaces are fixed in both the out-of-plane and the corresponding loading directions. The material parameters are chosen to be the same as in the previous example.

A mesh convergence study is conducted under an equibiaxial cyclic load with an amplitude of 45 MPa and a frequency of 10 Hz. In this benchmark example, the second-order, 20-node hexahedral element with reduced integration (C3D20R) is employed for spatial discretization. To further improve accuracy and efficiency, a gradient, structured mesh pattern is created, which is shown in Fig. 22. The finest discretization is located at the center of the thinned circular region, which is called the gauge zone and has a uniform element size. The size of this squared gauge zone is 10 mm by 10 mm. Fig. 23 shows that the maximum von-Mises stress converges when the element size is less than 0.5 mm in the gauge zone. In fact, the relative error of stresses between the coarsest and the finest discretizations is only 0.3% due to the higher-order element formulation. Therefore, the element size of 0.5 mm at the gauge zone is

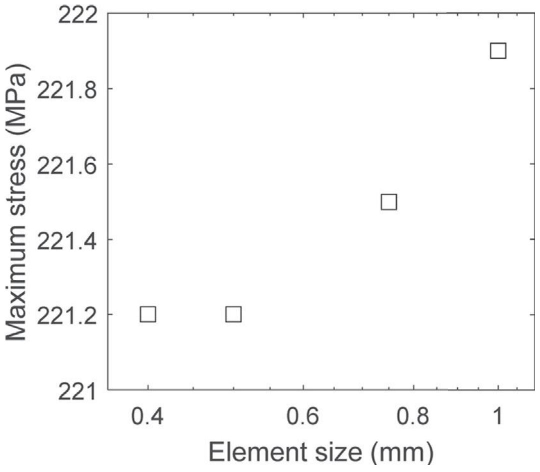

Fig. 23. Mesh convergence for the biaxial HCF specimen.

employed for the subsequent HCF simulations. It leads to a mesh size of 19,200 elements, 89,133 nodes, and 1,604,394 DOFs. The same time stepping strategy as in the previous example is used for tracking the initiation and propagation of HCF failure.

5.3.2. Results of biaxial HCF simulations

The first biaxial HCF simulation is performed under the same loading condition as in the equibiaxial case for the convergence study. The cracks initiate at 82,910 cycles and are located at the center of the gauge zone, the thinnest section of the specimen. After this, two orthogonal cracks

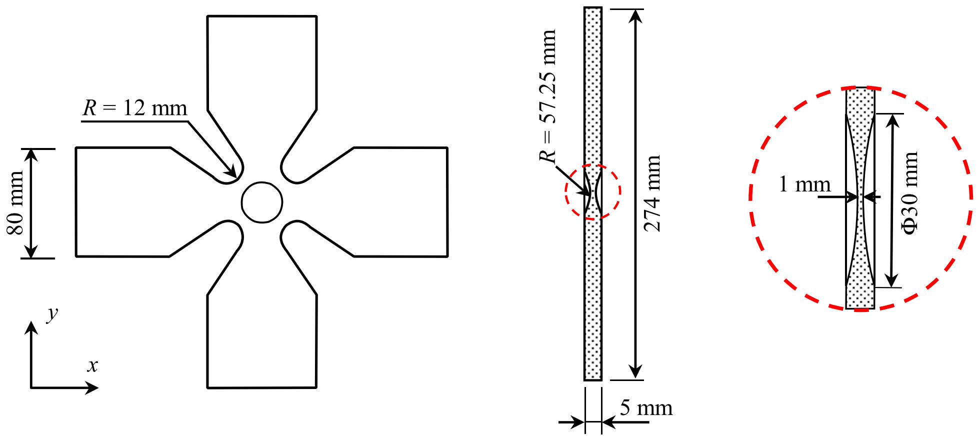

Fig. 21. Geometry and dimensions of the cross-shaped biaxial HCF specimen.

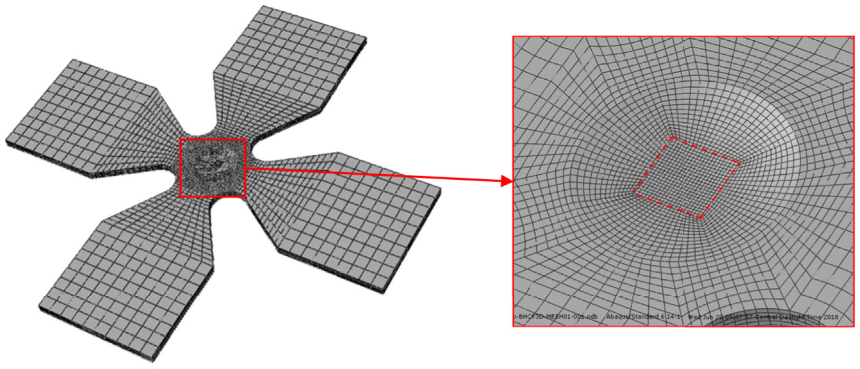

Fig. 22. Spatial discretization of the biaxial specimen, dashed box indicates the gauge zone.

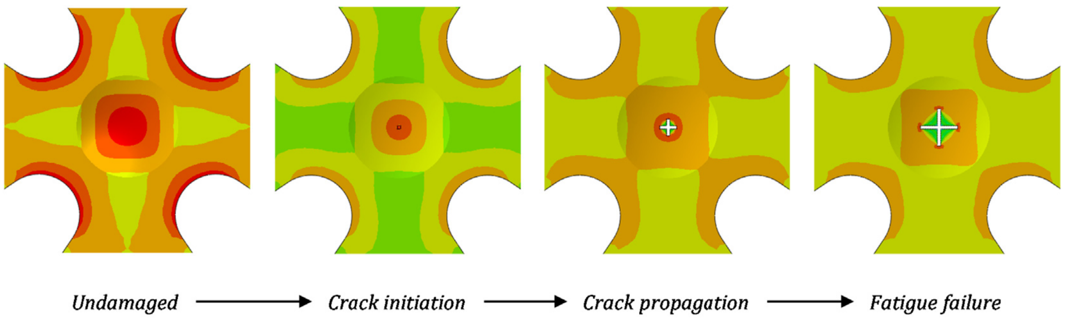

Fig. 24. Crack initiation and propagation of the biaxial HCF specimen (colored by von-Mises stress in logarithmic scale).

propagate along both x and y directions to the critical crack length (the size of the gauge zone) at 98,420 cycles. The process of crack initiation and propagation are illustrated in Fig. 24. Crack growth data obtained from the simulation is plotted in Fig. 25 against the number of cycles. Like the previous benchmark example, most fatigue life of the biaxial specimen is consumed by crack initiation. Due to the symmetry of geometry and loading, the crack growth along both directions are identical. The symmetry of simulation results can be clearly observed from Fig. 24.

To further study the interactions between loadings that are applied in different directions, two groups of biaxial HCF simulations are carried out. In the first group, fully-reversed biaxial cyclic loadings with constant amplitude are applied in both x and y directions. Load amplitudes in each direction, i.e. P_x and P_y, are varied from 35 to 55 MPa with an interval of 5 MPa. Thus, a total 25 combinations of load amplitudes are generated. The number of simulations is reduced to 15 combinations due to symmetry in x-y plane. Results of this group of biaxial HCF simulations are presented in Fig. 26, in which the discrete dots denote the simulation data and the trend surface is obtained from a curve fitting. Note that the case of $P_x = P_y = 35$ MPa is a runout, i.e. no damage initiation occurred during the entire simulation. From Fig. 26(a) it can be clearly observed that the fatigue life is monotonically increasing along the diagonal direction, where the biaxiality ratio is 1, i.e. $P_x = P_y$. However, as shown in Fig. 26(b), for a fixed value of P_y the fatigue life is not always monotonically increasing with the decrease of P_x. A 2D contour plot of the fatigue life shown in Fig. 26(c) clearly demonstrates such complex interactions between P_x and P_y.

In the second group of biaxial HCF simulations, the constant amplitude, fully-reversed cyclic loading is imposed only along the x direction. Amplitude of the cyclic loading is fixed at 30 MPa. A constant tensile load

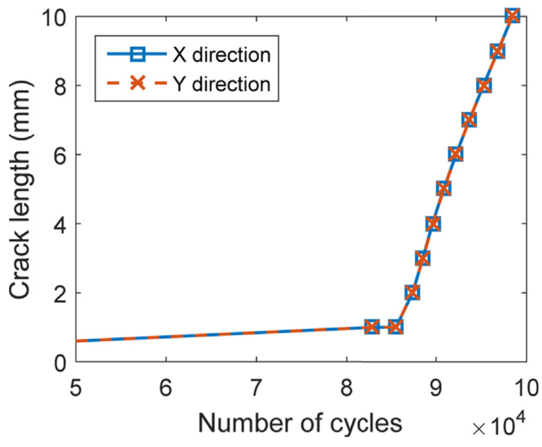

Fig. 25. Cracks growth of the biaxial specimen.

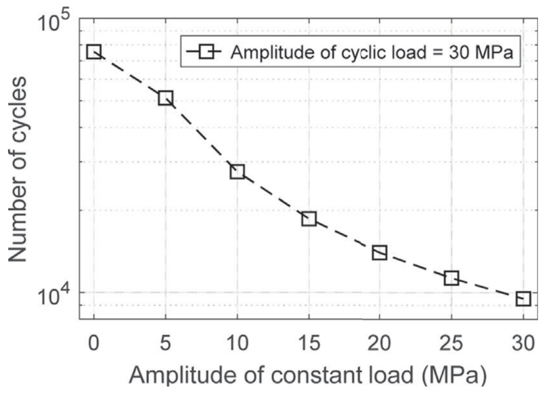

Fig. 27. Results of biaxial HCF simulations conducted under cyclic loading along x direction and constant loading along y direction.

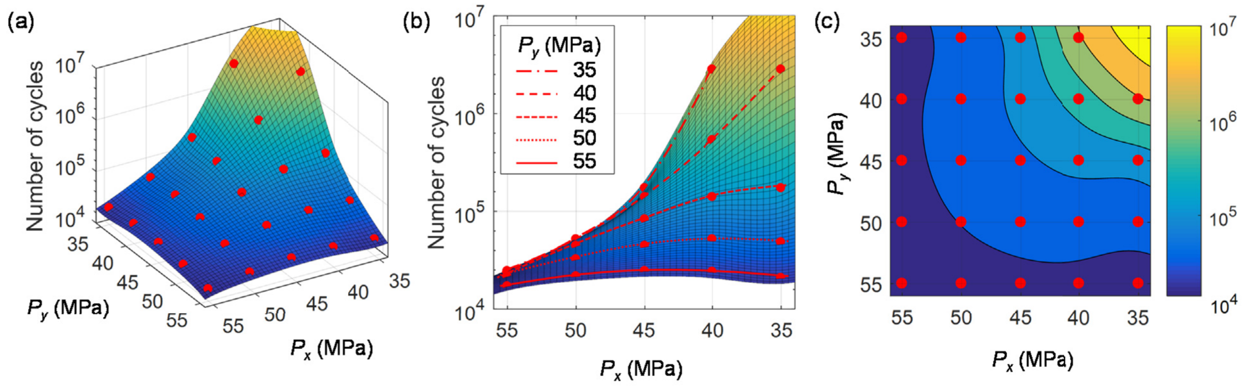

Fig. 26. Biaxial fatigue life as function of stress amplitudes in both x and y directions: (a) the 3D plot, (b) the projected 2D view of the 3D plot, and (c) the 2D contour plot (dots are simulation results; trend surface is obtained from a curve fitting).

Table 9
Computational performance of the proposed framework.

Number of DOFs	Number of compute nodes[a]	Wall-clock time usage (s)		
		Preconditioner	Solver (each step)	Damage (100 cycles)
1,604,394	1	10.0	2.1	70.2
3,034,890	2	18.4	2.7	69.6
10,025,820	6	54.3	8.2	77.3
15,557,778	10	93.9	10.0	73.3

[a] With 4 MPI processes per node and 6 OpenMP threads per process.

is applied in the y direction with amplitude varying from 0 MPa to 30 MPa with an interval of 5 MPa. Fig. 27 presents the results of the second group of biaxial HCF simulations. It shows that fatigue life decreases with the increasing amplitude of the constant load.

5.3.3. *Computational performance*

For different number of unknowns, the wall-clock time usages for three of the most computationally intensive parts are summarized in Table 9. Note that the time usage of damage model is based on the OpenMP version with 24 CPU cores per compute node. Table 9 shows that in HCF simulations the computing time is dictated by the solution of the nonlinear fatigue damage model. For each space-time slab, time usage of the damage model is about 7–35 times of the hybrid solver depending on problem size.

6. Conclusions

In summary, a high-performance multiscale computational framework is presented for direct numerical simulations of 3D HCF problems. The proposed framework is established by integrating the *Extended Space-time Finite Element Method* with the *Continuum Damage Mechanics*. The current work has been mainly focused on improving the numerical efficiency of the framework for practical HCF applications. This objective is achieved by developing a novel hybrid iterative/direct linear system solver and a high-performance hybrid parallel computing framework. Benchmark examples show that the serial version of the hybrid solver is at least 1–2 orders of magnitude faster in computing time and cheaper in memory consumption than the conventional solvers. The parallel hybrid solver efficiently handles XTFEM stiffness matrix equations with over 100 million unknowns using 64 CPU cores. Parallel implementations of the CDM-based two-scale damage model using CPUs and GPUs both achieve optimal speedup. Series of HCF simulations on the single edge notched specimen and the cruciform biaxial specimen demonstrate the capabilities of the proposed framework on handling large 3D problems and complex fatigue loading conditions.

With significant improvement in the numerical efficiency that is enabled by the proposed algorithms, the framework of XTFEM/CDM is ideal and efficient for predicting HCF responses in many engineering structures and components. The proposed approach can also serve as a robust tool for facilitating the experimental studies of HCF. Future efforts are directed towards integrating multiphysics methods such as thermo-mechanical coupling, analysis the effects of surface treatments induced residual stress on fatigue life, and fatigue problems in broad engineering applications.

Acknowledgements

The work of R. Zhang is supported by the Eugene McDermott Graduate Fellowship at The University of Texas at Dallas, which is gratefully acknowledged. The authors also would like to thank the National Science Foundation (grant# CMMI-1727960) for financial support of this research and thank the Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP) for materials based upon work supported by, or in part by, the User Productivity Enhancement, Technology Transfer, and Training (PETTT) contract #GS04T09DBC0017. This work was also supported in part by an allocation of computing time from the Texas Advanced Computing Center. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the funding agencies.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.finel.2019.103320.

References

[1] J. Cedergren, S. Melin, P. Lidström, Numerical modelling of P/M steel bars subjected to fatigue loading using an extended Gurson model, Eur. J. Mech. A Solid. 23 (6) (2004) 899–908, 2004/11/01.
[2] S. Oller, O. Salomón, E. Oñate, A continuum mechanics model for mechanical fatigue analysis, Comput. Mater. Sci. 32 (2) (2005) 175–195, 2005/02/01.
[3] A. Pirondi, N. Bonora, D. Steglich, W. Brocks, D. Hellmann, Simulation of failure under cyclic plastic loading by damage models, Int. J. Plast. 22 (11) (2006) 2146–2170, 2006/11/01.
[4] M. Takagaki, T. Nakamura, Fatigue crack modeling and simulation based on continuum damage mechanics, J. Press. Vessel Technol. 129 (1) (2007) 96–102.
[5] T. Bednarek, W. Sosnowski, Practical fatigue analysis of hydraulic cylinders – Part II, damage mechanics approach, Int. J. Fatigue 32 (10) (2010) 1591–1599, 2010/10/01.
[6] K.L. Roe, T. Siegmund, An irreversible cohesive zone model for interface fatigue crack growth simulation, Eng. Fract. Mech. 70 (2) (2003) 209–232, 2003/01/01.
[7] T. Siegmund, A numerical study of transient fatigue crack growth by use of an irreversible cohesive zone model, Int. J. Fatigue 26 (9) (2004) 929–939.
[8] H. Jiang, X. Gao, T.S. Srivatsan, Predicting the influence of overload and loading mode on fatigue crack growth: a numerical approach using irreversible cohesive elements, Finite Elem. Anal. Des. 45 (10) (2009) 675–685.
[9] N. Raje, T. Slack, F. Sadeghi, A discrete damage mechanics model for high cycle fatigue in polycrystalline materials subject to rolling contact, Int. J. Fatigue 31 (2) (2009) 346–360.
[10] P. Lestriez, F. Bogard, J.L. Shan, Y.Q. Guo, Damage evolution on mechanical parts under cyclic loading, AIP Conf. Proc. 908 (1) (2007) 1389–1394.
[11] L.G. Barbu, S. Oller, X. Martinez, A. Barbat, High cycle fatigue simulation: a new stepwise load-advancing strategy, Eng. Struct. 97 (2015) 118–129, 2015/08/15.
[12] S. Bhamare, T. Eason, S. Spottswood, S.R. Mannava, V.K. Vasudevan, D. Qian, A multi-temporal scale approach to high cycle fatigue simulation, Comput. Mech. 53 (2) (2014) 387–400.
[13] T.J.R. Hughes, G.M. Hulbert, Space-time finite element methods for elastodynamics: formulations and error estimates, Comput. Methods Appl. Mech. Eng. 66 (3) (1988) 339–363.
[14] G.M. Hulbert, Time finite element methods for structural dynamics, Int. J. Numer. Methods Eng. 33 (2) (1992) 307–331.
[15] G.M. Hulbert, T.J.R. Hughes, Space-time finite element methods for second-order hyperbolic equations, Comput. Methods Appl. Mech. Eng. 84 (3) (1990) 327–348.
[16] P. Lesaint, P.A. Raviart, On a finite element method for solving the neutron transport equation, in: C. de Boor (Ed.), Mathematical Aspects of Finite Elements in Partial Differential Equations, Academic press, New York, 1974, pp. 89–123.
[17] C. Johnson, Error estimates and adaptive time-step control for a class of one-step methods for stiff ordinary differential equations, SIAM J. Numer. Anal. 25 (4) (1988) 908–926.
[18] M. Delfour, W. Hager, F. Trochu, Discontinuous Galerkin methods for ordinary differential equations, Math. Comput. 36 (154) (1981) 455–473.
[19] N. Moes, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, Int. J. Numer. Methods Eng. 46 (1) (1999) 131–150.
[20] T. Strouboulis, I. Babuška, K. Copps, The design and analysis of the generalized finite element method, Comput. Methods Appl. Mech. Eng. 181 (1–3) (2000) 43–69.
[21] I. Babuška, J.M. Melenk, The partition of unity method, Int. J. Numer. Methods Eng. 40 (4) (1997) 727–758.
[22] J.M. Melenk, I. Babuska, The partition of unity finite element method: basic theory and applications, Comput. Methods Appl. Mech. Eng. 139 (1–4) (1996) 289–314.
[23] S. Chirputkar, D. Qian, Coupled atomistic/continuum simulation based on extended space-time finite element method, Cmes-Comput. Model. Eng. Sci. 24 (2–3) (2008) 185–202.
[24] J. Chessa, T. Belytschko, Arbitrary discontinuities in space–time finite elements by level sets and X-FEM, Int. J. Numer. Methods Eng. 61 (15) (2004) 2595–2614.
[25] Y. Yang, S. Chirputkar, D.N. Alpert, T. Eason, S. Spottswood, D. Qian, Enriched space-time finite element method: a new paradigm for multiscaling from elastodynamics to molecular dynamics, Int. J. Numer. Methods Eng. 92 (2) (2012) 115–140.
[26] D. Qian, S. Chirputkar, Bridging scale simulation of lattice fracture using enriched space-time Finite Element Method, Int. J. Numer. Methods Eng. 97 (11) (2014) 819–850.
[27] J. Lemaitre, J.P. Sermage, R. Desmorat, A two scale damage concept applied to fatigue, Int. J. Fract. 97 (1–4) (1999) 67–81.

[28] J. Lemaitre, I. Doghri, Damage 90: a post processor for crack initiation, Comput. Methods Appl. Mech. Eng. 115 (3–4) (1994) 197–232.

[29] R. Desmorat, A. Kane, M. Seyedi, J.P. Sermage, Two scale damage model and related numerical issues for thermo-mechanical High Cycle Fatigue, Eur. J. Mech. A Solid. 26 (6) (2007) 909–935.

[30] J.D. Eshelby, The determination of the elastic field of an ellipsoidal inclusion, and related problems, Proc. R. Soc. Lond. Ser A Math. Phys. Sci. 241 (1226) (1957) 376–396.

[31] E. Kröner, On the plastic deformation of polycrystals, Acta Metall. 9 (2) (1961) 155–161.

[32] S. Wada, R. Zhang, S.R. Mannava, V.K. Vasudevan, D. Qian, Simulation-based prediction of cyclic failure in rubbery materials using nonlinear space-time finite element method coupled with continuum damage mechanics, Finite Elem. Anal. Des. 138 (2018) 21–30.

[33] S. Cantournet, R. Desmorat, J. Besson, Mullins effect and cyclic stress softening of filled elastomers by internal sliding and friction thermodynamics model, Int. J. Solids Struct. 46 (11) (2009) 2255–2264, 2009/06/01.

[34] J. Lemaitre, R. Desmorat, Engineering Damage Mechanics: Ductile, Creep, Fatigue and Brittle Failures, Springer-Verlag Berlin Heidelberg, Berlin, 2005.

[35] R. Zhang, L. Wen, S. Naboulsi, T. Eason, V.K. Vasudevan, D. Qian, Accelerated multiscale space–time finite element simulation and application to high cycle fatigue life prediction, Comput. Mech. 58 (2) (2016) 329–349.

[36] T. Davis, Direct Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, 2006.

[37] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., Society for Industrial and Applied Mathematics, 2003, p. 537.

[38] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H.V. der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, second ed., SIAM, Philadelphia, PA, 1994.

[39] X.D. Li, N.E. Wiberg, Structural dynamic analysis by a time-discontinuous Galerkin finite element method, Int. J. Numer. Methods Eng. 39 (12) (1996) 2131–2152.

[40] C.C. Chien, T.Y. Wu, An improved predictor/multi-corrector algorithm for a time-discontinuous Galerkin finite element method in structural dynamics, Comput. Mech. 25 (5) (2000) 430–437.

[41] P. Kunthong, L.L. Thompson, An efficient solver for the high-order accurate time-discontinuous Galerkin (TDG) method for second-order hyperbolic systems, Finite Elem. Anal. Des. 41 (7–8) (2005) 729–762.

[42] R. Zhang, L. Wen, J. Xiao, D. Qian, An efficient solution algorithm for space–time finite element method, Comput. Mech. 63 (3) (2019) 455–470.

[43] P.R. Amestoy, A. Guermouche, J.-Y. L'Excellent, S. Pralet, Hybrid scheduling for the parallel solution of linear systems, Parallel Comput. 32 (2) (2006) 136–156.

[44] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, J. Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM J. Matrix Anal. Appl. 23 (1) (2001) 15–41.

[45] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.

[46] W. Gropp, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, second ed., The MIT Press, Cambridge, Massachusetts, 1999.

[47] V. Kumar, A. Grama, A. Gupta, G. Karypis, Introduction to Parallel Computing: Design and Analysis of Algorithms, Benjamin-Cummings Publishing Co., Inc., 1994.

[48] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1) (1998) 359–392.

[49] G. Karypis, V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, J. Parallel Distrib. Comput. 48 (1) (1998) 96–129, 1998/01/10.

[50] L. Dagum, R. Menon, OpenMP: an industry standard API for shared-memory programming, IEEE Comput. Sci. Eng. 5 (1) (1998) 46–55.

[51] R. Tian, Simulation at extreme-scale: Co-design thinking and practices, Arch. Comput. Methods Eng. 21 (1) (2014) 39–58. March 01.

[52] ORNL, Summit User Guide: System Overview, 2018, 12/21, https://www.olcf.ornl.gov/for-users/system-user-guides/summit/system-overview/.

[53] ORNL, Ttian User Guide: System Overview, 2018, 12/21, https://www.olcf.ornl.gov/for-users/system-user-guides/titan/system-overview/.

[54] R-CCS, System Configuration of the K Computer, 2018, 12/21, https://www.r-ccs.riken.jp/en/wp-content/uploads/system_handout.pdf.

[55] A.S. Bland, W. Joubert, R.A. Kendall, D.B. Kothe, J.H. Rogers, G.M. Shipman, Jaguar: the World's Most Powerful Computer System - an Update, 2010.

[56] TechPowerUp, GPU Specs Database: NVIDIA TESLA, 2018, 12/21, https://www.techpowerup.com/gpu-specs/?mfgr=NVIDIA&generation=Tesla&sort=generation.

[57] N. Lautrou, D. Thevenet, J.Y. Cognard, Fatigue crack initiation life estimation in a steel welded joint by the use of a two-scale damage model, Fatigue Fract. Eng. Mater. Struct. 32 (5) (2009) 403–417.

[58] W.K. Liu, D. Qian, S. Gonella, S. Li, W. Chen, S. Chirputkar, Multiscale methods for mechanical science of complex materials: bridging from quantum to stochastic multiresolution continuum, Int. J. Numer. Methods Eng. 83 (8–9) (2010) 1039–1080.

[59] R. Tian, S. Chan, S. Tang, A.M. Kopacz, J.-S. Wang, H.-J. Jou, L. Siad, L.-E. Lindgren, G.B. Olson, W.K. Liu, A multiresolution continuum simulation of the ductile fracture process, J. Mech. Phys. Solids 58 (10) (2010) 1681–1700, 2010/10/01.

[60] M. Anahid, M.K. Samal, S. Ghosh, Dwell fatigue crack nucleation model based on crystal plasticity finite element simulations of polycrystalline titanium alloys, J. Mech. Phys. Solids 59 (10) (2011) 2157–2176, 2011/10/01.

[61] S. Ghosh, P. Chakraborty, Microstructure and load sensitive fatigue crack nucleation in Ti-6242 using accelerated crystal plasticity FEM simulations, Int. J. Fatigue 48 (2013) 231–246, 2013/03/01.

[62] Z. Ma, L. Kong, X. Jin, An explicit-implicit mixed staggered asynchronous step integration algorithm in structural dynamics, CMES: Comput. Model. Eng. Sci. 116 (1) (2018) 51–67.

[63] T.L. Anderson, Fracture Mechanics: Fundamentals and Applications, 4 ed., CRC Press, 2017.

[64] L. Vincent, J.C. Le Roux, S. Taheri, On the high cycle fatigue behavior of a type 304L stainless steel at room temperature, Int. J. Fatigue 38 (2012) 84–91.

[65] S. Bhamare, G. Ramakrishnan, S.R. Mannava, K. Langer, V.K. Vasudevan, D. Qian, Simulation-based optimization of laser shock peening process for improved bending fatigue life of Ti–6Al–2Sn–4Zr–2Mo alloy, Surf. Coat. Technol. 232 (2013) 464–474, 2013/10/15.

[66] M.R. Karim, M. Kattoura, S.R. Mannava, V.K. Vasudevan, A.S. Malik, D. Qian, A computational study on the microstructural evolution in near-surface copper grain boundary structures due to femtosecond laser processing, Comput. Mech. 61 (1) (2018) 105–117, 2018/02/01.

[67] M. Poncelet, G. Barbier, B. Raka, S. Courtin, R. Desmorat, J.C. Le-Roux, L. Vincent, Biaxial High Cycle Fatigue of a type 304L stainless steel: cyclic strains and crack initiation detection by digital image correlation, Eur. J. Mech. A Solid. 29 (5) (2010) 810–825.

[68] R.A. Cláudio, L. Reis, M. Freitas, Biaxial high-cycle fatigue life assessment of ductile aluminium cruciform specimens, Theor. Appl. Fract. Mech. 73 (2014) 82–90, 2014/10/01/.