MULTISCALE METHODS FOR FATIGUE AND DYNAMIC FRACTURE FAILURE AND HIGH-PERFORMANCE COMPUTING IMPLEMENTATION

by

Rui Zhang

APPROVED BY SUPERVISORY COMMITTEE:

Dong Qian, Chair

Hongbing Lu

Ill Ryu

Rodrigo Bernal

Thomas Eason

This dissertation

is dedicated to my beloved wife Meng

and our daughter Norah.

MULTISCALE METHODS FOR FATIGUE AND DYNAMIC FRACTURE FAILURE AND HIGH-PERFORMANCE COMPUTING IMPLEMENTATION

by

RUI ZHANG, BS, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

MECHANICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT DALLAS

December 2020

ACKNOWLEDGMENTS

This dissertation would not be possible without the help and support from many people to whom I am grateful. First and foremost, I wish to express my deepest gratitude to my advisor, Professor Dong Qian, who provided me invaluable guidance on many aspects over the past years. It is a great honor to be his student and a privilege to work with him. I would also like to acknowledge my committee members: Professors Hongbing Lu, Ill Ryu, Rodrigo Bernal, and Dr. Thomas Eason. Their advice and encouragement are highly appreciated.

I am grateful to my colleagues and friends at The University of Texas at Dallas (UTD): Wenwei Jiang, Zhong Zhou, Shogo Wada, Clint Nicely, Mohammad Rezaul Karim, Tao Zhang, Yang Wang, Ryan Schlinkman, Yingjian Liu, Tingge Xu, Xuemin Wang, Lianjun Wu, Sadeq Malakooti, Mohammad Issa Hatamleh, Peiyuan Kang, Xiaowei Zhu, Lu Zhan, Jamasp Azarnoosh, Sumair Sunny, and many other individuals.

I wish to extend my sincere gratitude to my collaborators outside UTD: Professor Wing Kam Liu and his group from Northwestern University, Professor Zunfeng Liu and his group from Nankai University, and Alexey Falin and Luhua Li from Deakin University.

Finally, I would like to express an exceptional appreciation for my wife Meng, our daughter Norah, my parents and my extended family for their support and love.

This work was supported by the Eugene McDermott Graduate Fellowship at UTD, the Texas Advanced Computing Center at The University of Texas at Austin, and the National Science Foundation, which are gratefully acknowledged.

October 2020

MULTISCALE METHODS FOR FATIGUE AND DYNAMIC FRACTURE FAILURE AND HIGH-PERFORMANCE COMPUTING IMPLEMENTATION

Rui Zhang, PhD
The University of Texas at Dallas, 2020

Supervising Professor: Dong Qian, Chair

This dissertation presents several multiscale methods for material failure and implementations on high-performance computing (HPC) platforms. The work is motivated by the challenges in fully capturing the mechanics of failure using a single scale method. As such, multiscale approaches that incorporate multiple temporal and spatial scales have been established. To address the high computational costs, efficient algorithms and their implementations on the HPC platform featuring many-core architectures have been developed.

Based on the topics being addressed, the dissertation is divided into two parts. First, a multiscale computational framework for high cycle fatigue (HCF) life prediction is established by integrating the Extended Space-Time Finite Element Method (XTFEM) with multiscale fatigue damage models. XTFEM is derived based on the time-discontinuous Galerkin approach, which is shown to be A-stable and high-order accurate. While the robustness of XTFEM has been extensively demonstrated, the associated high computational cost remains a critical barrier for its practical applications. A novel hybrid iterative/direct solver is proposed with a unique preconditioner based on Kronecker product decomposition of the space-time stiffness matrix. XTFEM is further accelerated by utilizing HPC platforms featuring a hierarchy of distributed- and shared-memory parallelisms. A two-scale damage model is coupled with XTFEM to capture nonlinear material behaviors under HCF loading

and accelerated by parallel computing using both CPUs and GPUs. Furthermore, an efficient data-driven microstructure-based multiscale fatigue damage model is established by employing the Self-consistent Clustering Analysis, which is a reduced-order method derived from Machine Learning. Robustness and efficiency of the framework are demonstrated through benchmark problems. HCF simulations are conducted to quantify key effects due to mean stress, multiaxial load conditions, and material microstructures.

In the second part, a concurrent multiscale method to dynamic fracture is established by coupling Peridynamics (PD) with the classical Continuum Mechanics (CCM). PD is a novel nonlocal generalization of CCM. It is governed by an integro-differential equation of motion, which is free of spatial derivatives. This salient feature makes it attractive for problems with spatial discontinuities such as cracks. However, it generally leads to a much higher computational cost due to its nonlocality. There is a continuing interest to couple PD with CCM to improve efficiency while preserving accuracy in critical regions. In this work, Finite Element (FE) simulation is performed over the entire domain and coexists with a local PD region where crack pre-exists or is expected to initiate. The coupling scheme is accomplished by a bridging-scale projection between the two scales and a class of two-way nonlocal matching boundary conditions that eliminates spurious wave reflections at the numerical interface and transmits waves from the FE domain to the PD region. An adaptive scheme is established so that the PD region is dynamically relocated to track propagating crack. Accuracy and efficiency of the proposed method are illustrated by wave propagation examples. Its effectiveness and robustness in material failure simulation are demonstrated by benchmark problems featuring brittle fracture.

Finally, conclusions are drawn from the research work presented and prospective future developments of the established multiscale methods are provided.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

Material failures such as fatigue or fracture usually involve multiple scales in both space and time (Fish et al., 2012). The development of computational approaches for accurate prediction of mechanical responses with a wide range of spatial and temporal scales is of critical interest to many industrial applications, especially for those employing advanced heterogeneous materials such as composites (Matouš et al., 2017). It should be noted that even for materials that are conventionally considered homogeneous in the macroscopic mechanical analysis, accounting for detailed micro-structural features such as defects can substantially improve the accuracy of material failure prediction. However, direct numerical simulation of most engineering parts with existing computational approaches over a wide range of scales in both time and space is still prohibitively expensive. This grand challenge drives the development of multiscale approaches from various aspects over the past few decades (Hashin, 1960; Hill, 1965; Mori and Tanaka, 1973; Babuška, 1976; Papanicolau et al., 1978; Sánchez-Palencia, 1980; Dvorak, 1992; Hughes, 1995; Hughes and Stewart, 1996; Hughes et al., 1998; Feyel, 1999; Feyel and Chaboche, 2000; Garikipati and Hughes, 2000; Kouznetsova et al., 2001, 2002; Feyel, 2003; Kouznetsova et al., 2004; Oden et al., 2006; Oskay and Fish, 2007;

The following articles were reused in this chapter with permissions from the publishers:

1. Zhang, R., S. Naboulsi, T. Eason, and D. Qian (2019). A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing. *Finite Elements in Analysis and Design 166*, 103320. Reuse with permission from *Elsevier*.
2. Zhang, R., L. Wen, J. Xiao, and D. Qian (2019). An efficient solution algorithm for space-time finite element method. *Computational Mechanics 63*(3), 455–470. Reuse with permission from *Springer Nature*.
3. Zhang, R., L.Wen, S. Naboulsi, T. Eason, V. K. Vasudevan, and D. Qian (2016). Accelerated multiscale space-time finite element simulation and application to high cycle fatigue life prediction. *Computational Mechanics 58*(2), 329–349. Reuse with permission from *Springer Nature*.

Committee on Integrated Computational Materials Engineering, 2008; Kalamkarov et al., 2009; Kanouté et al., 2009; Yuan and Fish, 2009; National Science Technology Council, 2011; Fish, 2011; Coenen et al., 2012; Fish et al., 2012; Liu et al., 2016; Zhang and Oskay, 2016; Oliver et al., 2017; Yu et al., 2019), which grows into an active and fruitful interdisciplinary research field with collaborative efforts by researchers coming from different backgrounds in science and engineering.

1.2 High cycle fatigue life prediction

1.2.1 Empirical models

Fatigue is a failure mechanism that dominates the design of many engineering structures and components (Schütz, 1996; Schijve, 2003). Most of the fatigue design approaches employed by the industry today belong to the category of either *safe-life* or *damage-tolerance* approach (Basquin, 1910; Irwin, 1957; Coffin, 1960; Manson, 1965; Paris and Erdogan, 1963; Cui, 2002; Oller et al., 2005). 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 of the empirical models to complex fatigue loading conditions, such as extrapolations or corrections for mean stress effect, variable amplitude, multiaxial loading, and random loading spectrum remain questionable (Wheeler, 1972; Takagaki and Nakamura, 2007; Molent et al., 2008).

1.2.2 Numerical simulation approaches

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) (Cedergren et al., 2004; Oller et al., 2005; Pirondi et al., 2006; Takagaki and Nakamura, 2007; Bednarek and Sosnowski, 2010). In these methods, finite element analysis provides the stress/strain histories under complex fatigue loading, while the progressive damage evolution is characterized by internal variables in the CDM framework (Lemaitre, 1985; Mear and Hutchinson, 1985; Becker and Needleman, 1986; Bonora and Newaz, 1998). Numerical tools for LCF are readily available in commercial finite element software (Simulia, 2014b). However, for high cycle fatigue (HCF) problems, direct finite element simulation has not been possible due to the large number of load 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 (Roe and Siegmund, 2003; Siegmund, 2004; Lestriez et al., 2007; Jiang et al., 2009; Raje et al., 2009; Barbu et al., 2015). 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 this method greatly extends the predictive capabilities, the assumption of constant load amplitude does not always hold for practical applications. In addition, mathematical consistency of the *jump-in-cycles* approaches is still questionable and needs further study.

1.2.3 Space-Time Finite Element Method

Material/structural responses under fatigue loading are inherently dynamic. In the field of computational mechanics, it is a common practice to employ the so-called semi-discrete scheme to solve the time-dependent dynamic problems. In this implementation, the spatial domain of the partial differential equations (PDEs) is first discretized by finite element mesh or meshfree particles, producing a system of ordinary differential equations (ODEs) with time as the independent variable. Subsequently, the temporal domain is resolved using the finite difference approach. For problems featuring strong temporal nonlinearities such as

sharp gradient or jump, however, traditional FEM based on semi-discrete schemes is not well suited for these types of analysis as it lacks the flexibility in establishing multiscale approximations in the temporal domains. Specific time integration schemes such as the popular *central difference* or *Newmark-β* methods are known to suffer from either the time-step constraints or poor convergence. As such, there is an immense difficulty associated with the semi-discrete scheme to handle the above issues that are frequently encountered in practical engineering problems.

In contrast to the traditional semi-discrete schemes, *Space-Time Finite Element Method* (STFEM) is emerging as an interesting method as it provides an entirely different way of treating temporal scales. The idea of discretizing both the spatial and temporal domains by finite elements was first proposed in (Argyris and Scharpf, 1969; Fried, 1969; Oden, 1969) and derived based on Hamilton's principle for dynamics during the late 1960s — shortly after the establishment of regular FEM. In the initial developments of STFEM, the entire temporal domain was discretized and continuous approximations for the unknowns were introduced. This led to the *Time-Continuous Galerkin* (TCG) formulation (Hughes et al., 1987). TCG generally yields very large system of equations due to the simultaneous discretizations of the spatial and temporal domains, which severely limits its application. As an alternative, a divide-and-conquer approach was proposed in which the entire spatial-temporal domain is first partitioned into smaller space-time slabs. A Galerkin formulation is then established within each space-time slab. The resulting formulation is called a *Time-Discontinuous Galerkin* (TDG) formulation (Reed and Hill, 1973; Lesaint and Raviart, 1974) as the neighboring space-time slabs are coupled through the jump conditions in the weak form. TDG method was originally developed for solving first-order hyperbolic equations (Reed and Hill, 1973; Lesaint and Raviart, 1974). It was further extended to second-order hyperbolic systems such as elastodynamics in (Hughes and Hulbert, 1988; Hulbert and Hughes, 1990; Hulbert, 1992; Hughes and Stewart, 1996; Li and Wiberg, 1996, 1998). Based on the

choice of unknown fields, two different formulations, i.e., single-field and two-field formulations have been developed (Hughes and Hulbert, 1988; Hulbert, 1992). It was shown that TDG method significantly reduces artificial oscillations that are commonly associated with semi-discrete time integration schemes in capturing sharp gradients or discontinuities (Hulbert and Hughes, 1990). Also, the TDG formulation has been proved to be A-stable and high-order accurate (Lesaint and Raviart, 1974; Delfour et al., 1981; Hughes and Hulbert, 1988; Johnson, 1988; Hulbert and Hughes, 1990; Hulbert, 1992; Hughes and Stewart, 1996; Li and Wiberg, 1996, 1998; Wiberg and Li, 1999; Chien and Wu, 2000; Chien et al., 2003).

1.2.4 Extended Space-Time FEM

Based on the key concepts introduced in *Generalized FEM* (GFEM) (Strouboulis et al., 2000), *Extended FEM* (XFEM) (Moës et al., 1999) and *Partition of Unity Method* (PUM) (Melenk and Babuška, 1996; Babuška and Melenk, 1997), the widely adopted polynomial-based shape function in STFEM can be further enhanced using enrichment functions that represent the problem physics (Chessa and Belytschko, 2004; Réthoré et al., 2005; Chirputkar and Qian, 2008; Yang et al., 2012; Qian and Chirputkar, 2014). This enriched formulation is referred to as the *Extended Space-Time FEM* (XTFEM). For example, an enriched space-time formulation was proposed by Chessa and Belytschko (Chessa and Belytschko, 2004) to model arbitrary discontinuities in the temporal domain. An enriched formulation for coupled atomistic-continuum simulation of lattice fractures was developed by Chirputkar and Qian (Chirputkar and Qian, 2008; Qian and Chirputkar, 2014). Furthermore, Yang et al. (Yang et al., 2012) demonstrated the advantages of XTFEM for effectively handling the dynamic problems at both the continuum and atomistic levels. XTFEM formulation was shown to have better convergence properties over the regular space-time method for a proper choice of enrichment function.

To address the challenges associated with HCF simulations, Bhamare et al. (Bhamare et al., 2014) developed a computational framework based on XTFEM and CDM. In most

HCF applications, the load conditions typically consist of static (mean) and dynamic (alternating) loads, or dynamic loads with both low and high frequencies. The magnitude of the load is moderate so that the mesoscopic material response can be considered elastic. Correspondingly, the structural response also features a mixture of two components: one slow varying in time (coarse scale) and the other fast oscillating (fine scale). The basic idea of the enrichment is to capture the fine scale using enrichment functions, as opposed to employing very fine grids in the temporal dimension. It was shown that the enriched approximation effectively captured the dynamic response and enabled the use of very large time step size under various practical fatigue loading histories (Bhamare et al., 2014). To capture the multiscale material behavior in HCF, XTFEM was coupled with a two-scale CDM model proposed by Lemaitre et al. (Lemaitre and Doghri, 1994; Lemaitre et al., 1999) and Desmorat et al. (Desmorat et al., 2007). The damage model approximates mesoscale material behavior as elastic, while plasticity and damage are modeled 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 (Eshelby, 1957; Kröner, 1961). 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 (Bhamare et al., 2014).

More recently, Wada et al. (Wada et al., 2018) further extended this framework to predict the cyclic failure of rubber by considering both geometric and material nonlinearities at mesoscale. A CDM-based model developed by Lemaitre et al. (Lemaitre and Desmorat, 2005) and Cantournet et al. (Cantournet et al., 2009) 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.

1.2.5 Motivation and objective of this study

While many advantages of the TDG-based space-time methods have been extensively demonstrated in previous studies, the associated high computational expense remains a critical barrier for practical applications due to the additional time dimension that is introduced (Hulbert, 1992). As such, the extended predictive capability of the method is paid at the price of converting an n-dimensional spatial problem to an $(n+1)$-dimensional space-time problem. This issue is further convoluted with the enrichment of the approximations. Compared with the standard FEM, TDG-based FEM typically leads to larger system of coupled equations that are expressed in the matrix form of $\boldsymbol{\mathcal{K}d} = \boldsymbol{\mathcal{F}}$ in which $\boldsymbol{\mathcal{K}}$ is space-time stiffness matrix of size $N \times N$. Assuming a quadratic interpolation in time and the number of spatial degrees of freedom (DOFs), then $N = 3n_s$, $4n_s$, and $6n_s$ respectively for single-field, two-field STFEM and XTFEM. Obtaining solutions to the space-time stiffness equation requires $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage if the direct solver is employed, which are several orders of magnitude higher comparing to solving the corresponding stiffness equation in standard FEM. In addition, the space-time stiffness matrix is non-symmetric, less sparse compared to that of regular FEM and generally not well conditioned (Zhang et al., 2016). Another significant contribution to the computational cost comes from the constitutive updates, especially for nonlinear materials. Although a similar issue also exists in the standard FEM, implementation of the CDM-based multiscale fatigue damage models in XTFEM is more expensive due to the need to resolve stress-strain histories at spatial-temporal interpolation points.

One approach to accelerate the solution of space-time stiffness equation is introducing a multiplicative form of the space-time shape function. With this decomposition, it can be shown that the space-time stiffness matrix $\boldsymbol{\mathcal{K}}$ is generally expressed as

$$\boldsymbol{\mathcal{K}} = \boldsymbol{\Phi} \otimes \boldsymbol{K} + \boldsymbol{\Psi} \otimes \boldsymbol{M} \tag{1.1}$$

in which $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are temporal submatrices, $\boldsymbol{K}$ and $\boldsymbol{M}$ are spatial stiffness and mass submatrices respectively, and symbol $\otimes$ denotes the Kronecker product. It should be noted that $\boldsymbol{K}$ and $\boldsymbol{M}$ matrices have exactly the same form as their counterparts in the standard FEM.

In the case of two-field TDG formulation, $\boldsymbol{\mathcal{K}}$ matrix is weakly coupled, i.e., there is no single block in $\boldsymbol{\mathcal{K}}$ matrix coupled to both $\boldsymbol{K}$ and $\boldsymbol{M}$ matrices. A family of iterative predictor/multi-corrector algorithms have been developed based on this distinctive feature (Li and Wiberg, 1996; Chien and Wu, 2000; Kunthong and Thompson, 2005). In these algorithms, the original equations are first recast into a partially decoupled form. Subsequently stationary iterative method, such as Gauss-Jacobi method (Li and Wiberg, 1996; Kunthong and Thompson, 2005) or Gauss-Seidel method (Chien and Wu, 2000), is applied to the multi-corrector phase. However, these algorithms are not directly applicable to the single-field TDG formulation or the enriched formulation as the corresponding space-time stiffness matrix is fully coupled.

Alternatively, a general preconditioned iterative solution algorithm was developed in (Zhang et al., 2016). It first constructs the preconditioner by incomplete factorization of space-time stiffness matrix $\boldsymbol{\mathcal{K}}$, in which the computational cost is minimized using matrix reordering algorithms based on Graph theory. The preconditioned space-time system of linear equations is then solved by nonstationary iterative method based on Krylov-subspace approach. It was shown that this algorithm worked efficiently for both the single-field TDG formulation and XTFEM on two-spatial-dimensional (2D) problems. However, the approach does not take into account of the unique block structure of $\boldsymbol{\mathcal{K}}$ matrix as indicated by Eq. (1.1). Further benchmark tests yielded degraded performance for three-spatial-dimensional (3D) problems with this algorithm, which is due to the fact that the condition number of $\boldsymbol{\mathcal{K}}$ matrix in 3D has significantly deteriorated when compared with the 2D cases.

Motivated by the challenges associated with the computational cost as outlined above, the main objective of this study is to establish an efficient solution algorithm to significantly

scale down the computational cost of both the regular and extended STFEM based on TDG formulation and enhance the efficiency of the XTFEM/CDM computational framework and extend it to large-scale 3D HCF simulations. This goal is realized by further exploiting the unique block structure of coupled space-time matrix equations (Zhang et al., 2019). The proposed algorithm has two key components. First, a novel and efficient Kronecker preconditioner is proposed by utilizing the special block structure of space-time matrix and properties of temporal and spatial submatrices. Second, matrix-vector multiplications and preconditioning operations, which are the most computing-intensive operations associated with iterative solvers, are optimized and accelerated employing the inverse property of Kronecker product. Computational cost of the resulting algorithm is first analyzed theoretically and then demonstrated in both 2D and 3D numerical examples using various TDG formulations. It is shown that performance of the proposed algorithm is at least $1 \sim 2$ orders of magnitude better than that of either direct sparse solver or the previously developed iterative approach (Zhang et al., 2016) for problems with relatively large number of unknowns (e.g., $N > 10^4$). Through this novel implementation, the computational cost of solving space-time stiffness equations is reduced to the same order as solving the corresponding stiffness equations in standard FEM, thereby enabling practical applications of STFEM.

To further accelerate the XTFEM/CDM computational framework, a novel hybrid linear system solver is established and the corresponding implementation on HPC platform is developed (Zhang et al., 2019). The hybrid iterative/direct linear system solver is proposed based on the previous Kronecker preconditioning algorithm. HPC implementation is aimed at accelerating both the hybrid linear system solver and nonlinear CDM constitutive solver. It features a hierarchy of parallelisms that first partitioning the space-time computational domain and then redistributing the computing-intensive tasks associated with each subdomain while minimizing communication. The 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 CPUs and Graphic Processing Units (GPUs) hardware platforms. Computational performance of the established framework is demonstrated through several benchmark examples. It is shown that the serial computational performance of the proposed hybrid solver is at least $1 \sim 2$ orders of magnitude better than conventional sparse direct and iterative linear system solvers 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 model show optimal speedup using either CPUs or GPUs. Capabilities of the proposed framework in handling complex fatigue load conditions are demonstrated by benchmark problems on HCF application.

Finally, as a significant extension to the two-scale fatigue damage model that was developed earlier, a data-driven microstructure-based concurrent multiscale fatigue damage model has been established by integrating the *Self-consistent Clustering Analysis* (SCA) developed by (Liu et al., 2016). It is a novel reduced-order multiscale material model derived from *Machine Learning* techniques. The method greatly reduces the computational cost of direct modeling of complex material microstructures. For HCF applications, an efficient solution algorithm is developed to accelerate its numerical implementation. Examples are presented to demonstrate the unique capability of the proposed method. Although this study is still at the preliminary stage, it shows great promises in high-fidelity HCF simulations and microstructure-based high performance material design in the future.

1.3 Dynamic fracture simulation

Fracture is one of the most common and catastrophic material failure mechanisms (Anderson, 2017). Fracture problems are intrinsically multiscale and dynamic (Rafii-Tabar, 1998).

Cracks usually initiate and propagate at the nanoscale, then carry over various length scales, and eventually lead to macroscale fracture failure observed in engineering structures. In the event of brittle fracture, the entire above-mentioned multiscale crack growth could happen rapidly in a very short time period. Developing accurate, robust and efficient computational approaches to dynamic fracture problems over a wide spectrum of length scales poses a great challenge to the computational solid mechanics community (Aduloju and Truster, 2019; Patil and Heider, 2019).

1.3.1 Atomistic scale approaches and AtC coupling

As an atomistic simulation approach, *Molecular Dynamics* (MD) has been extensively employed to simulate fracture problems in various applications, such as carbon nanotubes (CNT), graphene, silicon carbide and other nanostructured solids (Ashurst and Hoover, 1976; Swadener et al., 2002; Kikuchi et al., 2005; Karimi et al., 2006; Kang and Cai, 2007; Cheng and Sun, 2014; Dewapriya et al., 2014; Zhang et al., 2014; Bitzek et al., 2015). Cracks in MD simulation are simply represented by atoms being pulled apart from each other. The unique advantage of MD fracture simulation over the continuum scale approaches is that it considers atomistic scale features that cannot be directly modeled by the latter, e.g., direct interactions between atoms/molecules, crystal structure, grain size, lattice spacing, etc. Hence, it provides comprehensive understanding of the physics in dynamic fracture at the nanoscale. It is interesting to note that many nanoscale defect interactions such as dislocation patterning are often driven by long-range fields that do not require an atomistic description (Curtin and Miller, 2003). However, MD simulation of such phenomenon at larger scales is still out of reach even with the fastest supercomputers. Therefore, the spatial and temporal scales that can be handled by MD are very limited. This restriction can be partially relaxed to a certain extent by the so-called *coarse-grained* MD (CGMD) simulation (Rudd and Broughton, 1998; Aoyagi et al., 2002; Rudd and Broughton, 2005; Bond et al., 2007; Wallace and Sansom, 2007; Qian et al., 2015).

To further circumvent the limitation of MD simulation in larger scales, one alternative approach is to employ multiscale descriptions by coupling MD with continuum scale approaches (Curtin and Miller, 2003). In multiscale approaches, MD simulation is typically limited to a small local region while the continuum model is employed elsewhere. A large amount of work has been devoted to this so-called *atomistic-to-continuum* (AtC) coupling approach during the past two decades. Cai et al. (Cai et al., 2000) introduced the Langevin approach to minimize wave reflections at boundaries for linear systems. Wagner and Liu (Wagner and Liu, 2003) proposed a bridging-scale method based on a projection operator and a time-history kernel approach. Qian et al. (Qian et al., 2004) further extended this approach and proposed a *virtual atom cluster* (VAC) model in coarse-scale meshfree approximation for post-buckling analysis of CNT structures. E and Huang (E and Huang, 2001, 2002) developed nonreflecting interfaces by eliminating the high frequency components. Belytschko and Xiao (Belytschko and Xiao, 2003) proposed a bridging domain method for coupling molecular and continuum mechanics and further extended the method to dynamics in (Xiao and Belytschko, 2004; Xu and Belytschko, 2008). To and Li (To and Li, 2005) and Li et al. (Li et al., 2006) developed a multiscale method by combining the bridging scale method and the *perfectly matched layer* (PML). Spurious reflections are eliminated by matching the impedance at the interface between MD and PML. Tang and co-workers (Tang, 2008; Wang and Tang, 2010, 2013) proposed *matching boundary conditions* (MBCs) to suppress wave reflections. In MBCs, velocities of boundary atoms are directly correlated to displacements/velocities at a local neighborhood of atoms. The coefficients in MBCs are solved by minimizing a residual and its derivatives that are proportional to wave dispersion relation mismatch. The *quasi-continuum* (QC) method is another major class of coupling method between MD and FEM (Tadmor et al., 1996; Shenoy et al., 1999), which is mainly applied in static or quasi-static problems. Space-time FEM introduced in the previous section is coupled with MD for dynamic fracture simulation in (Chirputkar and Qian, 2008;

Yang et al., 2012; Qian and Chirputkar, 2014). More recently, Tong and Li (Tong and Li, 2016) and Gur et al. (Gur et al., 2019) proposed multiscale coupling methods between MD and Peridynamics, which is a nonlocal continuum theory to be introduced later.

1.3.2 Classical continuum scale approaches

Crack surfaces in the *classical continuum mechanics* (CCM) theory can be represented by boundaries of the model geometry. However, in the simulation of dynamic fracture problems using FEM, this means continuous remeshing has to be performed to match the moving discontinuities and mapping of solution between meshes, which is not only cumbersome but also problematic. In the context of FEM, there are three major methods to model dynamic crack propagation, which are discussed as follows.

The simplest and probably the most widely employed crack modeling approach is the so-called *element deletion* or *erosion* method (Belytschko and Lin, 1987; Johnson and Stryk, 1987). In this approach, an element is removed from the model by setting its stress to zero once a certain failure criterion is satisfied, e.g., the stress at an associated quadrature point exceeds the material strength. Although the element deletion method is simple and straightforward in terms of implementation, it is known to suffer from mesh dependency and it does not satisfy the conservation laws of mass and energy in the system since both of them are lost with the deleted elements.

The second major class of crack modeling method is the *cohesive zone model* (CZM) (Dugdale, 1960; Barenblatt, 1962; Needleman, 1987; Xu and Needleman, 1994; Yang et al., 2001; Elices et al., 2002; Roe and Siegmund, 2003; Song et al., 2006; Park and Paulino, 2013; Needleman, 2014; Nordmann et al., 2020), which is popular for problems with cracks along known paths. CZM is an interelement crack method in the sense that crack is modeled by either the separation of regular elements at their interfaces or the failure of special surface elements that are placed between regular elements. CZM constitutive behavior is governed

by phenomenological traction separation laws. As one can expect, this method also leads to mesh dependence since cracks can only grow along element boundaries and remeshing is necessary in many situations.

To resolve the mesh dependency and avoid the cumbersome remeshing in the above methods, Belytschko and Black (Belytschko and Black, 1999) and Moës et al. (Moës et al., 1999) develpoed the XFEM to model crack and its growth by introducing discontinuous and near-tip asymptotic enrichment functions to the standard FE shape function through the PUM concept (Melenk and Babuška, 1996; Babuška and Melenk, 1997). In XFEM, cracks can grow within elements. Therefore, it is more suitable for dynamic fracture simulation since arbitrary crack path can be modeled without remeshing. Song et al. (Song et al., 2008) conducted a comparative study among the above FE-based methods for dynamic fracture. They concluded that none of these methods can accurately predict the speed of crack propagation or its path even for relatively simple problems in 2D, not to mention more complex scenarios such as fragmentation.

1.3.3 Peridynamics and its coupling with CCM

The main difficulty of fracture modeling in CCM lies in its mathematical foundation, which is based on PDEs that are not well defined over spatial discontinuities such as crack. CCM is also a local theory in the sense that material points only interact with immediate neighbors. However, accounting for nonlocality is critical in material damage modeling (Bažant et al., 1984; Pijaudier-Cabot and Bažant, 1987; Bažant and Lin, 1988; Bažant and Pijaudier-Cabot, 1988; Bažant, 1994; Pijaudier-Cabot et al., 2004).

Motivated by these limitations in CCM for fracture modeling, Silling (Silling, 2000) and co-workers (Silling et al., 2007) proposed the *Peridynamics* (PD) in the early 2000s, which is a novel nonlocal generalization of CCM to deal with discontinuities and long-range forces without any special treatment. PD is governed by a spatial-integral temporal-differential

equation of motion, which is well defined mathematically at discontinuities. In PD, material points interact with each other directly through bonds over a finite distance termed as horizon radius, which is an internal length scale. Crack initiates and propagates spontaneously in PD as a result of bond breaking between material points. Discretization of the governing equations of PD typically leads to a meshfree formulation (Silling and Askari, 2005) that shows some similarities to other meshfree methods (Bessa et al., 2014). Compared to CCM and other nonlocal methods, PD is still a relatively new theory with an active, increasing research community. In terms of applications to dynamic fracture, PD has been employed by Bobaru and co-workers to study crack branching in brittle materials (Ha and Bobaru, 2010; Bobaru and Zhang, 2015) as well as impact and fragmentation of multi-layer glass systems (Bobaru et al., 2016). PD applications on damage modeling of fiber-reinforced composites are reported in (Kilic et al., 2009; Hu et al., 2012; Oterkus and Madenci, 2012; Diyaroglu et al., 2019). Many other applications and theoretical aspects of PD are extensively studied, please refer to (Askari et al., 2008; Silling and Lehoucq, 2010; Madenci and Oterkus, 2014; Bobaru et al., 2016; Javili et al., 2019) for reviews on this method.

A major drawback of PD also comes from its nonlocality. Its computational cost is usually much higher compared to the classical FEM due to the nonlocal interactions between material points, which makes modeling of large-scale applications very expensive. Similar to AtC coupling approaches, the coupling between PD and CCM has been actively studied in the past decade. To transmit the forces between PD and FEM subdomains in the coupled simulation, a class of coupling approach termed as the *splice* method is developed and employed in both static and dynamic fracture simulations (Silling et al., 2015; Galvanetto et al., 2016; Zaccariotto et al., 2018; Bie et al., 2018; Kulkarni and Tabarraei, 2018; Ni et al., 2019). Kilic and Madenci (Kilic and Madenci, 2010) introduced an overlap region to couple PD and FEM. Oterkus et al. (Oterkus et al., 2012) presented a sub-modeling approach to couple PD and FEM for the failure prediction of curved composite panel with stiffeners. In (Liu and Hong,

2012), the authors developed a coupling approach based on interface elements for quasi-static problems, which is also employed by Lee et al. (Lee et al., 2017) and Yaghoobi and Chorzepa (Yaghoobi and Chorzepa, 2018). Energy-based blending method such as the *Arlequin* method (Han and Lubineau, 2012) is introduced to couple local and nonlocal continuum models. Seleson et al. (Seleson et al., 2013) proposed a force-based blending scheme. In these blending methods, the computational domain is decomposed into subdomains modeled by PD or FEM and an overlap region to blend these models. More recently, Yu et al. (Yu et al., 2018) proposed an optimization-based coupling scheme by finding optimal coefficient for Robin boundary conditions. For static and quasi-static problems, Sun and Fish (Sun and Fish, 2019) proposed a superposition-based coupling scheme between PD and FEM. For elastodynamic problems, Wang et al. (Wang et al., 2019) proposed a concurrent coupling scheme based on the Arlequin method and showed that the spurious wave reflections can be effectively suppressed. Giannakeas et al. (Giannakeas et al., 2019) compared different coupling schemes between PD and FEM on wave reflections at the numerical interfaces. They also developed a coupling scheme between PD and XFEM for brittle fracture simulation (Giannakeas et al., 2020a,b). Brief reviews on these coupling approaches can be found in (Yu et al., 2018; Sun and Fish, 2019).

1.3.4 Motivation and objective of this study

This study is motivated by the AtC coupling approaches as well as the coupling schemes between PD and FEM for the purpose of multiscale dynamic fracture simulation. Since this work mainly focuses on modeling dynamic fracture for larger-scale problems, we employ PD and FEM to develop a coupled simulation approach to address dynamic fracture at the continuum scale. The numerical interfaces between PD and FEM has not been systematically investigated. Improper treatments can lead to spurious wave reflections at the PD/FEM interface. As such, a critical development is to establish a class of effective boundary condition

called *matching boundary condition* (MBC) to realize non-reflective condition. The existing literature also lacks studies on developing concurrent multiscale scheme for dynamic fracture applications. Therefore, main objective of this study is to integrate MBC with a concurrent multiscale framework based on PD and FEM for dynamic fracture problems to achieve both high accuracy and efficiency.

In the proposed concurrent multiscale framework, we first establish a coarse-fine scale decomposition of the system based on which multiscale Lagrangian of the system and equations of motion are derived. The coarse-scale equations are further solved with FEM for the entire domain of interest. The coarse-scale solution coexists with a localized fine-scale PD subdomain where material fracture pre-exists or is expected to initiate. A bridging-scale projection of the fine-scale PD solution onto the overlap coarse-scale FE basis functions is introduced based on the work by Wagner and Liu (Wagner and Liu, 2003). High frequency waves are originated from fine-scale PD simulation due to bond breaking during crack initiation and propagation. Artificial reflections of these short waves at the PD/FE interfaces are detrimental to accuracy of the fine-scale PD simulation. To eliminate artificial wave reflections at the numerical interface as well as transmit long waves from coarse-scale FE model to fine-scale PD model, a class of two-way *nonlocal matching boundary conditions* (NMBC) is developed based on the previous work described in (Wang and Tang, 2010, 2013; Wang et al., 2017; Nicely et al., 2018). To further accommodate the evolving nature of dynamic fracture, an adaptive scheme is established so that PD region is dynamically prescribed to track propagating crack (Wada, 2017). The accuracy and efficiency of the proposed concurrent multiscale framework are first illustrated by wave propagation examples. The effectiveness and robustness in material failure simulation are further demonstrated by benchmark examples that involve brittle fracture.

1.4 Outline of the dissertation

The rest of the dissertation is organized in two parts as follows.

In Part I, we start by introducing the theory of space-time FEM in Chapter 2. Then several efficient solution algorithms for implementing the space-time FEM are proposed and the corresponding HPC implementations are developed in Chapter 3. In Chapter 4, we establish the multiscale damage model and develop highly efficient numerical algorithms based on hybrid CPUs/GPUs platform. In Chapter 5, a data-driven microstructure-based concurrent multiscale material modeling method is proposed using SCA for HCF applications. Finally, applications of the proposed computational framework for HCF applications are presented in Chapter 6.

In Part II, we first introduce various PD formulations accompanied by several numerical examples in Chapter 7. Chapter 8 presents a closed-form wave dispersion analysis on discretized PD in both 1D and 2D cases, which is critical to the coupling scheme. In Chapter 9, the concurrent multiscale coupling scheme between PD and CCM is established. Numerical examples on elastodynamics wave propagation and dynamic fracture problems are presented in Chapter 10 to demonstrate the performances of the proposed concurrent multiscale simulation approach.

Finally, conclusion and future work are presented in Chapter 11.

PART I

A HIGH PERFORMANCE MULTISCALE SPACE-TIME METHOD TO HIGH CYCLE FATIGUE LIFE PREDICTION

CHAPTER 2

SPACE-TIME FINITE ELEMENT METHOD

2.1 Introduction

In this chapter, we first briefly review the formulations of TDG-based STFEM and its enriched version. Then the corresponding numerical implementations are established. Finally, several numerical examples are provided to demonstrate the advantages of STFEM/XTFEM over the traditional semi-discrete FEM.

2.2 Time-Discontinuous Galerkin Method

We start by briefly reviewing the TDG formulations for elastodynamics developed by Hughes and Hulbert (Hughes and Hulbert, 1988; Hulbert and Hughes, 1990; Hulbert, 1992). More details can be found in (Hulbert and Hughes, 1990; Hulbert, 1992; Yang et al., 2012; Bhamare et al., 2014)

2.2.1 Strong form of the governing equations

Let us consider the initial/boundary value problem (IBVP) defined over a spatial region Ω and the corresponding temporal domain $I =]0, T[$. The spatial region Ω is bounded by $\Gamma = \Gamma_t \cup \Gamma_u$, where Γ_t and Γ_u are the non-overlapping traction (Neumann) and essential

---

The following articles were reused in this chapter with permissions from the publishers:

1. Zhang, R., S. Naboulsi, T. Eason, and D. Qian (2019). A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing. *Finite Elements in Analysis and Design 166*, 103320. Reuse with permission from *Elsevier*.
2. Zhang, R., L. Wen, J. Xiao, and D. Qian (2019). An efficient solution algorithm for space-time finite element method. *Computational Mechanics 63*(3), 455–470. Reuse with permission from *Springer Nature*.
3. Zhang, R., L.Wen, S. Naboulsi, T. Eason, V. K. Vasudevan, and D. Qian (2016). Accelerated multiscale space-time finite element simulation and application to high cycle fatigue life prediction. *Computational Mechanics 58*(2), 329–349. Reuse with permission from *Springer Nature*.

(Dirichlet) boundaries, respectively. The strong form of governing equations is given as,

$$\rho\ddot{\boldsymbol{u}} = \nabla \cdot \boldsymbol{\sigma}(\nabla \boldsymbol{u}) + \boldsymbol{f} \qquad \text{on } Q \equiv \Omega \times]0, T[\tag{2.1}$$

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \qquad \text{on } \Upsilon_u \equiv \Gamma_u \times]0, T[\tag{2.2}$$

$$\boldsymbol{n} \cdot \boldsymbol{\sigma}(\nabla \boldsymbol{u}) = \boldsymbol{t} \qquad \text{on } \Upsilon_t \equiv \Gamma_t \times]0, T[\tag{2.3}$$

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}) \qquad \text{for } \boldsymbol{x} \in \Omega \tag{2.4}$$

$$\dot{\boldsymbol{u}}(\boldsymbol{x}, 0) = \boldsymbol{v}_0(\boldsymbol{x}) \qquad \text{for } \boldsymbol{x} \in \Omega \tag{2.5}$$

where $\rho = \rho(\boldsymbol{x})$ is the volumetric mass density, $\boldsymbol{u}$ represents the displacement vector, $\boldsymbol{f}$ is the body force per unit volume, $\boldsymbol{n}$ is the unit outward normal vector to traction surface Γ_t, $\boldsymbol{\sigma}(\nabla \boldsymbol{u}) = \boldsymbol{C} : \boldsymbol{\varepsilon}$ under the assumption of linear elasticity and $\boldsymbol{C}$ is the constitutive matrix, $\bar{\boldsymbol{u}}$ and $\boldsymbol{t}$ are the prescribed boundary displacement and traction, $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$ denote the initial displacement and velocity. A superposed dot indicates the partial differentiation with respect to time.

2.2.2 Space-Time discretization

In the TDG approach, the space-time domain $Q = \Omega \times I$ is first divided into multiple segments called space-time slabs and the n-th space-time slab is given as $Q_n = \Omega \times I_n$ where $I_n =]t_{n-1}, t_n[$. Displacement and traction boundary conditions are respectively defined on $(\Upsilon_u)_n = \Gamma_u \times I_n$ and $(\Upsilon_t)_n = \Gamma_t \times I_n$. 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 e-th 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 $Q_n^\Sigma = \bigcup_{e=1}^{(n_{el})_n} Q_n^e$ and $\Upsilon_n^\Sigma = \bigcup_{e=1}^{(n_{el})_n} \Upsilon_n^e - \Upsilon_n$ respectively. Figure 2.1 shows an example of the TDG space-time discretization described above.

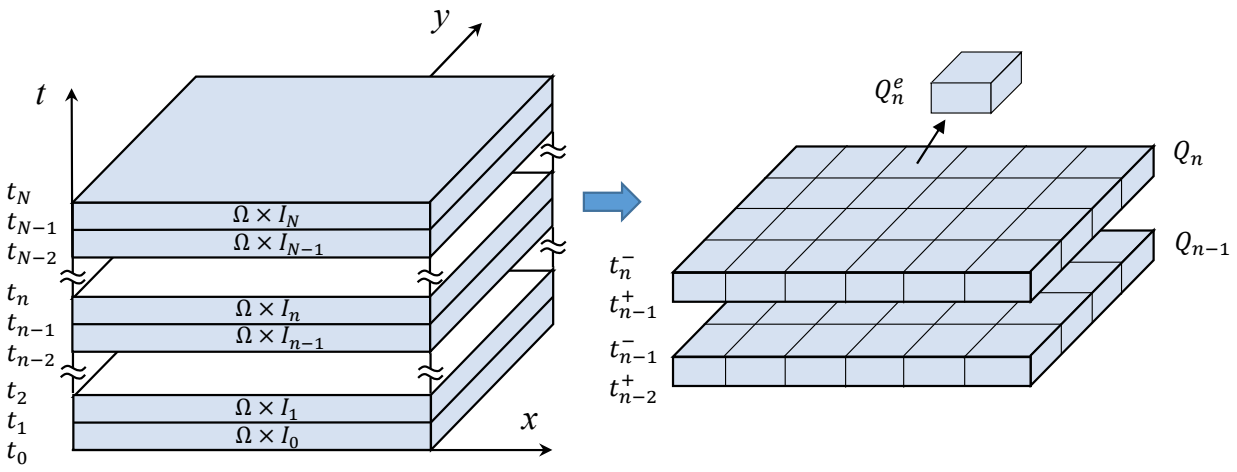


Figure 2.1. An illustration of TDG space-time discretization for 2D case

The following inner product notations are defined for deriving the TDG formulation,

$$\left(\boldsymbol{w}^h, \boldsymbol{u}^h\right)_{\Omega} = \int_{\Omega} \boldsymbol{w}^h \cdot \boldsymbol{u}^h d\Omega \tag{2.6}$$

$$a\left(\boldsymbol{w}^h, \boldsymbol{u}^h\right)_{\Omega} = \int_{\Omega} \nabla \boldsymbol{w}^h \cdot \boldsymbol{\sigma}(\nabla \boldsymbol{u}^h) d\Omega \tag{2.7}$$

$$\left(\boldsymbol{w}^h, \boldsymbol{u}^h\right)_{Q_n} = \int_{Q_n} \boldsymbol{w}^h \cdot \boldsymbol{u}^h dQ \tag{2.8}$$

$$a\left(\boldsymbol{w}^h, \boldsymbol{u}^h\right)_{Q_n} = \int_{Q_n} \nabla \boldsymbol{w}^h \cdot \boldsymbol{\sigma}(\nabla \boldsymbol{u}^h) dQ \tag{2.9}$$

$$\left(\boldsymbol{w}^h, \boldsymbol{u}^h\right)_{Q_n^{\Sigma}} = \int_{Q_n^{\Sigma}} \boldsymbol{w}^h \cdot \boldsymbol{u}^h dQ \tag{2.10}$$

$$\left(\boldsymbol{w}^h, \boldsymbol{u}^h\right)_{\Upsilon_n^{\Sigma}} = \int_{\Upsilon_n^{\Sigma}} \boldsymbol{w}^h \cdot \boldsymbol{u}^h d\Upsilon \tag{2.11}$$

$$\left(\boldsymbol{w}^h, \boldsymbol{u}^h\right)_{(\Upsilon_t)_n} = \int_{(\Upsilon_t)_n} \boldsymbol{w}^h \cdot \boldsymbol{u}^h d\Upsilon \tag{2.12}$$

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

$$[\![\boldsymbol{u}(t_n)]\!] = \boldsymbol{u}(t_n^+) - \boldsymbol{u}(t_n^-) \tag{2.13}$$

$$[\![\boldsymbol{u}(\boldsymbol{x})]\!] = \boldsymbol{u}(\boldsymbol{x}^+) - \boldsymbol{u}(\boldsymbol{x}^-) \tag{2.14}$$

in which

$$\boldsymbol{u}(t_n^{\pm}) = \lim_{\epsilon \to 0^{\pm}} \boldsymbol{u}(t_n + \epsilon) \tag{2.15}$$

$$\boldsymbol{u}(\boldsymbol{x}^{\pm}) = \lim_{\epsilon \to 0^{\pm}} \boldsymbol{u}(\boldsymbol{x} + \epsilon \boldsymbol{n}) \tag{2.16}$$

$$\boldsymbol{n} = \boldsymbol{n}^{+} = -\boldsymbol{n}^{-} \tag{2.17}$$

2.3 Space-Time FEM

2.3.1 Single-field formulation

Displacements are chosen as the basic unknowns in the single-field formulation. The weak form is derived by introducing the displacement trial functions $\boldsymbol{u}^h(\boldsymbol{x}, t)$ and test functions $\delta\boldsymbol{u}^h(\boldsymbol{x}, t)$ to be C^0 continuous within each space-time slab. Trial and test functions can have discontinuities across the space-time slabs. The spaces of the trial function and test function are given as

$$\boldsymbol{u}^h(\boldsymbol{x}, t) \in \boldsymbol{U} \qquad \boldsymbol{U} = \{\boldsymbol{u}^h(\boldsymbol{x}, t) \mid \boldsymbol{u}^h \in C^0(\bigcup_{n=1}^{N} Q_n),\ \boldsymbol{u}^h = \bar{\boldsymbol{u}} \text{ on } \Gamma_u\} \tag{2.18}$$

$$\delta\boldsymbol{u}^h(\boldsymbol{x}, t) \in \boldsymbol{U}_0 \qquad \boldsymbol{U}_0 = \{\delta\boldsymbol{u}^h(\boldsymbol{x}, t) \mid \delta\boldsymbol{u}^h \in C^0(\bigcup_{n=1}^{N} Q_n),\ \delta\boldsymbol{u}^h = \boldsymbol{0} \text{ on } \Gamma_u\} \tag{2.19}$$

With these definitions, the weak form of single-field formulation is expressed in a bilinear form. For the n-th space-time slab, it is given as

$$B_{DG}\left(\delta\boldsymbol{u}^h, \boldsymbol{u}^h\right)_n = L_{DG}\left(\delta\boldsymbol{u}^h\right)_n, \qquad n = 1, 2, ... \tag{2.20}$$

where

$$\begin{aligned} B_{DG}\left(\delta\boldsymbol{u}^h, \boldsymbol{u}^h\right)_n = &\left(\delta\dot{\boldsymbol{u}}^h, \rho\ddot{\boldsymbol{u}}^h\right)_{Q_n} + a\left(\delta\dot{\boldsymbol{u}}^h, \boldsymbol{u}^h\right)_{Q_n} + \left(\delta\dot{\boldsymbol{u}}^h(t_{n-1}^+), \rho\dot{\boldsymbol{u}}^h(t_{n-1}^+)\right)_{\Omega} \\ &+ a\left(\delta\boldsymbol{u}^h(t_{n-1}^+), \boldsymbol{u}^h(t_{n-1}^+)\right)_{\Omega} \end{aligned} \tag{2.21}$$

$$\begin{aligned} L_{DG}\left(\delta\boldsymbol{u}^h\right)_n = &\left(\delta\dot{\boldsymbol{u}}^h, \boldsymbol{f}\right)_{Q_n} + \left(\delta\dot{\boldsymbol{u}}^h, \boldsymbol{t}\right)_{(\Upsilon_t)_n} + \left(\delta\dot{\boldsymbol{u}}^h(t_{n-1}^+), \rho\dot{\boldsymbol{u}}^h(t_{n-1}^-)\right)_{\Omega} \\ &+ a\left(\delta\boldsymbol{u}^h(t_{n-1}^+), \boldsymbol{u}^h(t_{n-1}^-)\right)_{\Omega} \end{aligned} \tag{2.22}$$

2.3.2 Two-field formulation

Both displacement and velocity are taken as unknown fields in the two-field formulation. The weak form is derived by introducing the trial functions $\mathbf{U}^h(\boldsymbol{x},t) = \left\{\boldsymbol{u}^h, \boldsymbol{v}^h\right\}$ and test functions $\delta\mathbf{U}^h(\boldsymbol{x},t) = \left\{\delta\boldsymbol{u}^h, \delta\boldsymbol{v}^h\right\}$ to be C^0 continuous within each space-time slab. Similarly, trial and test functions can have discontinuities across the space-time slabs. The spaces of the trial functions are given as

$$\boldsymbol{u}^h(\boldsymbol{x},t) \in \boldsymbol{U} \qquad \boldsymbol{U} = \{\boldsymbol{u}^h(\boldsymbol{x},t) \mid \boldsymbol{u}^h \in C^0(\bigcup_{n=1}^{N} Q_n),\ \boldsymbol{u}^h = \bar{\boldsymbol{u}} \text{ on } \Gamma_u\} \tag{2.23}$$

$$\boldsymbol{v}^h(\boldsymbol{x},t) \in \boldsymbol{V} \qquad \boldsymbol{V} = \{\boldsymbol{v}^h(\boldsymbol{x},t) \mid \boldsymbol{v}^h \in C^0(\bigcup_{n=1}^{N} Q_n),\ \boldsymbol{v}^h = \dot{\bar{\boldsymbol{u}}} \text{ on } \Gamma_u\} \tag{2.24}$$

and the spaces of the test functions are

$$\delta\boldsymbol{u}^h(\boldsymbol{x},t) \in \boldsymbol{U}_0 \qquad \boldsymbol{U}_0 = \{\delta\boldsymbol{u}^h(\boldsymbol{x},t) \mid \delta\boldsymbol{u}^h \in C^0(\bigcup_{n=1}^{N} Q_n),\ \delta\boldsymbol{u}^h = \boldsymbol{0} \text{ on } \Gamma_u\} \tag{2.25}$$

$$\delta\boldsymbol{v}^h(\boldsymbol{x},t) \in \boldsymbol{V}_0 \qquad \boldsymbol{V}_0 = \{\delta\boldsymbol{v}^h(\boldsymbol{x},t) \mid \delta\boldsymbol{v}^h \in C^0(\bigcup_{n=1}^{N} Q_n),\ \delta\boldsymbol{v}^h = \boldsymbol{0} \text{ on } \Gamma_u\} \tag{2.26}$$

With these definitions, the weak form of two-field formulation for the n-th space-time slab can be expressed as

$$B_{DG}\left(\delta\mathbf{U}^h, \mathbf{U}^h\right)_n = L_{DG}\left(\delta\mathbf{U}^h\right)_n, \qquad n = 1, 2, ... \tag{2.27}$$

where

$$\begin{aligned} B_{DG}\left(\delta\mathbf{U}^h, \mathbf{U}^h\right)_n = &\left(\delta\boldsymbol{v}^h, \rho\dot{\boldsymbol{v}}^h\right)_{Q_n} + a\left(\delta\boldsymbol{v}^h, \boldsymbol{u}^h\right)_{Q_n} + a\left(\delta\boldsymbol{u}^h, \left(\dot{\boldsymbol{u}}^h - \boldsymbol{v}^h\right)\right)_{Q_n} \\ &+ \left(\delta\boldsymbol{v}^h(t_{n-1}^+), \rho\boldsymbol{v}^h(t_{n-1}^+)\right)_\Omega + a\left(\delta\boldsymbol{u}^h(t_{n-1}^+), \boldsymbol{u}^h(t_{n-1}^+)\right)_\Omega \end{aligned} \tag{2.28}$$

$$\begin{aligned} L_{DG}\left(\delta\mathbf{U}^h\right) = &\left(\delta\boldsymbol{v}^h, \boldsymbol{f}\right)_{Q_n} + \left(\delta\boldsymbol{v}^h, \boldsymbol{t}\right)_{(\Upsilon_t)_n} + \left(\delta\boldsymbol{v}^h(t_{n-1}^+), \rho\boldsymbol{v}^h(t_{n-1}^-)\right)_\Omega \\ &+ a\left(\delta\boldsymbol{u}^h(t_{n-1}^+), \boldsymbol{u}^h(t_{n-1}^-)\right)_\Omega \end{aligned} \tag{2.29}$$

2.3.3 Space-time shape function and stiffness matrix

Space-time shape function $\boldsymbol{N}(\boldsymbol{x},t)$ is constructed in a multiplicative form so that the temporal and spatial domains are approximated independently, i.e.,

$$\boldsymbol{N}(\boldsymbol{x},t) = \boldsymbol{N}_t \otimes \boldsymbol{N}_{\boldsymbol{x}} = \begin{bmatrix} N_{t_1}\boldsymbol{N}_{\boldsymbol{x}} & \cdots & N_{t_i}\boldsymbol{N}_{\boldsymbol{x}} & \cdots & N_{t_k}\boldsymbol{N}_{\boldsymbol{x}} \end{bmatrix} \tag{2.30}$$

where $\boldsymbol{N}_{\boldsymbol{x}}$ and $\boldsymbol{N}_t$ are the spatial and temporal shape functions respectively, symbol $\otimes$ denotes the Kronecker product, subscript i represents the i-th temporal node. Shape functions from standard FEM can be employed for the spatial shape function $\boldsymbol{N}_{\boldsymbol{x}}$.

2.3.3.1 Single-field formulation

For the single-field formulation, a 3-node quadratic shape function is employed for $\boldsymbol{N}_t$ in this work and is given by

$$\boldsymbol{N}_t = \begin{bmatrix} \frac{2(t_n - t)(t_{n-1/2} - t)}{\Delta t^2} & \frac{-4(t_n - t)(t_{n-1} - t)}{\Delta t^2} & \frac{2(t_{n-1} - t)(t_{n-1/2} - t)}{\Delta t^2} \end{bmatrix} \tag{2.31}$$

in which Δt is the size of the temporal mesh (time step size). The three nodes at t_{n-1}, $t_{n-1/2}$ and t_n are equally spaced along the time axis for each space-time slab.

For elastodynamic problems with linear elastic material and small deformation, the weak form Eq. (2.20) eventually leads to a discretized stiffness matrix equation for each space-time slab

$$\boldsymbol{\mathcal{K}}\boldsymbol{d} = \boldsymbol{\mathcal{F}} \tag{2.32}$$

where $\boldsymbol{\mathcal{K}}$ is space-time stiffness matrix, $\boldsymbol{d}$ and $\boldsymbol{\mathcal{F}}$ are unknown nodal displacement and prescribed external force vectors respectively. It can be shown that

$$\begin{aligned} \boldsymbol{\mathcal{K}} = & \int_{Q_n} \dot{\boldsymbol{N}}^T \rho \ddot{\boldsymbol{N}} dQ + \int_{Q_n} \dot{\boldsymbol{N}}_{,\boldsymbol{x}}^T \boldsymbol{C} \boldsymbol{N}_{,\boldsymbol{x}} dQ \\ & + \int_{\Omega} \dot{\boldsymbol{N}}^T \left(t_{n-1}^+\right) \rho \dot{\boldsymbol{N}} \left(t_{n-1}^+\right) d\Omega + \int_{\Omega} \boldsymbol{N}_{,\boldsymbol{x}}^T \left(t_{n-1}^+\right) \boldsymbol{C} \boldsymbol{N}_{,\boldsymbol{x}} \left(t_{n-1}^+\right) d\Omega \end{aligned} \tag{2.33}$$

and

$$\begin{aligned}\boldsymbol{\mathcal{F}} = &\int_{(\Upsilon_t)_n} \dot{\boldsymbol{N}}^T \boldsymbol{t} d\Upsilon + \int_{Q_n} \dot{\boldsymbol{N}}^T \boldsymbol{f} dQ \\ &+ \left[\int_\Omega \dot{\boldsymbol{N}}^T \left(t_{n-1}^+\right) \rho \dot{\boldsymbol{N}} \left(t_{n-1}^-\right) d\Omega + \int_\Omega \boldsymbol{N}_{,\boldsymbol{x}}^T \left(t_{n-1}^+\right) \boldsymbol{C} \boldsymbol{N}_{,\boldsymbol{x}} \left(t_{n-1}^-\right) d\Omega\right] \boldsymbol{d}_{n-1}\end{aligned} \tag{2.34}$$

Note that terms on the second lines of Eqs. (2.33) and (2.34) are contributed by the weak enforcement of temporal discontinuities across the adjacent space-time slabs, or the so-called *jump terms*.

Based on the space-time shape function defined in Eq. (2.30), integrations over the spatial domain can be done independently from the ones over the temporal domain. Therefore, the 1st and 2nd terms on right-hand side of Eq. (2.21)) are given as

$$\left(\delta \dot{\boldsymbol{u}}^h, \rho \ddot{\boldsymbol{u}}^h\right)_{Q_n} = \delta \boldsymbol{d}^T \left(\int_{Q_n} \dot{\boldsymbol{N}}^T \rho \ddot{\boldsymbol{N}} dQ\right) \boldsymbol{d} = \delta \boldsymbol{d}^T \left[\left(\int_{I_n} \dot{\boldsymbol{N}}_t^T \ddot{\boldsymbol{N}}_t dt\right) \otimes \boldsymbol{M}\right] \boldsymbol{d} \tag{2.35}$$

$$a\left(\delta \dot{\boldsymbol{u}}^h, \boldsymbol{u}^h\right)_{Q_n} = \delta \boldsymbol{d}^T \left(\int_{Q_n} \dot{\boldsymbol{N}}_{,\boldsymbol{x}}^T \boldsymbol{C} \boldsymbol{N}_{,\boldsymbol{x}} dQ\right) \boldsymbol{d} = \delta \boldsymbol{d}^T \left[\left(\int_{I_n} \dot{\boldsymbol{N}}_t^T \boldsymbol{N}_t dt\right) \otimes \boldsymbol{K}\right] \boldsymbol{d} \tag{2.36}$$

in which $\delta \boldsymbol{d}$ is arbitrary virtual displacement that can be dropped from both sides of Eq. (2.20), $\boldsymbol{d}$ is nodal displacement vector, $\boldsymbol{K}$ and $\boldsymbol{M}$ are spatial stiffness and mass matrices in standard FEM, respectively. Thus, the space-time stiffness matrix $\boldsymbol{\mathcal{K}}$ in Eq. (2.32) can be express as

$$\boldsymbol{\mathcal{K}} = \boldsymbol{\Phi} \otimes \boldsymbol{K} + \boldsymbol{\Psi} \otimes \boldsymbol{M} \tag{2.37}$$

where the temporal matrices are

$$\boldsymbol{\Phi} = \int_{I_n} \dot{\boldsymbol{N}}_t^T \boldsymbol{N}_t dt + \boldsymbol{N}_t^T(t_{n-1}^+) \boldsymbol{N}_t(t_{n-1}^+) \tag{2.38}$$

$$\boldsymbol{\Psi} = \int_{I_n} \dot{\boldsymbol{N}}_t^T \ddot{\boldsymbol{N}}_t dt + \dot{\boldsymbol{N}}_t^T(t_{n-1}^+) \dot{\boldsymbol{N}}_t(t_{n-1}^+) \tag{2.39}$$

Given the quadratic temporal shape functions for the single-field formulation (Eq. (2.31)), integrations in temporal domain in Eqs. (2.38) and (2.39) can be evaluated analytically.

Therefore, the stiffness matrix $\boldsymbol{\mathcal{K}}$ in Eq. (2.32) can be expressed as

$$\boldsymbol{\mathcal{K}} = \begin{bmatrix} \frac{5\boldsymbol{M}}{\Delta t^2} + \frac{\boldsymbol{K}}{2} & -\frac{4\boldsymbol{M}}{\Delta t^2} - \frac{2\boldsymbol{K}}{3} & -\frac{\boldsymbol{M}}{\Delta t^2} + \frac{\boldsymbol{K}}{6} \\ -\frac{12\boldsymbol{M}}{\Delta t^2} + \frac{2\boldsymbol{K}}{3} & \frac{16\boldsymbol{M}}{\Delta t^2} & -\frac{4\boldsymbol{M}}{\Delta t^2} - \frac{2\boldsymbol{K}}{3} \\ \frac{7\boldsymbol{M}}{\Delta t^2} - \frac{\boldsymbol{K}}{6} & -\frac{12\boldsymbol{M}}{\Delta t^2} + \frac{2\boldsymbol{K}}{3} & \frac{5\boldsymbol{M}}{\Delta t^2} + \frac{\boldsymbol{K}}{2} \end{bmatrix} \tag{2.40}$$

Eq. (2.40) clearly shows that the space-time stiffness matrix of single-field formulation is fully coupled, i.e., all the terms are featured by the combinations of $\boldsymbol{K}$ and $\boldsymbol{M}$ with the exception of $\boldsymbol{\mathcal{K}}_{22}$. It can also be expressed in the form of Kronecker products as shown below

$$\boldsymbol{\mathcal{K}} = \frac{1}{6}\begin{bmatrix} 3 & -4 & 1 \\ 4 & 0 & -4 \\ -1 & 4 & 3 \end{bmatrix} \otimes \boldsymbol{K} + \frac{1}{\Delta t^2}\begin{bmatrix} 5 & -4 & -1 \\ -12 & 16 & -4 \\ 7 & -12 & 5 \end{bmatrix} \otimes \boldsymbol{M} \tag{2.41}$$

2.3.3.2 Two-field formulation

The same decomposition of space-time shape function (Eq. (2.30)) is employed in the two-field formulation. Linear temporal shape functions are defined for both the displacement and the velocity fields, i.e. the P1-P1 element in (Hulbert and Hughes, 1990; Hulbert, 1992) is adopted. Similarly, by employing analytical temporal integration, the corresponding space-time stiffness matrix is obtained as

$$\boldsymbol{\mathcal{K}} = \begin{bmatrix} \frac{1}{2}\boldsymbol{K} & \frac{1}{2}\boldsymbol{K} & -\frac{\Delta t}{3}\boldsymbol{K} & -\frac{\Delta t}{6}\boldsymbol{K} \\ -\frac{1}{2}\boldsymbol{K} & \frac{1}{2}\boldsymbol{K} & -\frac{\Delta t}{6}\boldsymbol{K} & -\frac{\Delta t}{3}\boldsymbol{K} \\ \frac{\Delta t}{3}\boldsymbol{K} & \frac{\Delta t}{6}\boldsymbol{K} & \frac{1}{2}\boldsymbol{M} & \frac{1}{2}\boldsymbol{M} \\ \frac{\Delta t}{6}\boldsymbol{K} & \frac{\Delta t}{3}\boldsymbol{K} & -\frac{1}{2}\boldsymbol{M} & \frac{1}{2}\boldsymbol{M} \end{bmatrix} \tag{2.42}$$

Eq. (2.42) can also be rewritten as

$$\boldsymbol{\mathcal{K}} = \frac{1}{6}\begin{bmatrix} 3 & 3 & -2\Delta t & -\Delta t \\ -3 & 3 & -\Delta t & -2\Delta t \\ 2\Delta t & \Delta t & 0 & 0 \\ \Delta t & 2\Delta t & 0 & 0 \end{bmatrix} \otimes \boldsymbol{K} + \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \otimes \boldsymbol{M} \tag{2.43}$$

As shown in Eq. (2.43), the space-time stiffness matrix of the two-field formulation is weakly coupled, i.e., each term involves either $\boldsymbol{K}$ or $\boldsymbol{M}$ but not both. Hence, the corresponding matrix equation can be recast into a partially decoupled form, which can be solved by a family of iterative predictor/multi-corrector solution algorithms (Li and Wiberg, 1996; Chien and Wu, 2000; Kunthong and Thompson, 2005).

2.4 Extended Space-Time FEM

For certain class of problems, the polynomials-based shape function employed by STFEM may not provide the ideal basis for interpolation. In such cases, the predictive capability of STFEM can be further improved by introducing an enrichment function that represents the problem physics (Chirputkar and Qian, 2008; Yang et al., 2012; Qian and Chirputkar, 2014; Bhamare et al., 2014). The resulting formulation is termed as XTFEM, and referred to as either the GFEM (Strouboulis et al., 2000) or XFEM (Moës et al., 1999) if discretization is applied only to the spatial domain. Both GFEM and XFEM are based on the partition-of-unity concept (PUM) (Melenk and Babuška, 1996). In this work, the XTFEM is derived based on the single-field STFEM.

In XTFEM, the unknown displacement field is approximated by

$$\boldsymbol{u}(\boldsymbol{x},t) = \sum_I \bar{\boldsymbol{N}}_I(\boldsymbol{x},t)\bar{\boldsymbol{d}}_I + \sum_J \tilde{\boldsymbol{N}}_J(\boldsymbol{x},t)\tilde{\boldsymbol{d}}_J \tag{2.44}$$

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

For the J-th node, the enriched space-time shape function is given by

$$\tilde{\boldsymbol{N}}_J(\boldsymbol{x},t) = \boldsymbol{N}_J(\boldsymbol{x},t)\Phi_J(\boldsymbol{x},t) \tag{2.45}$$

where the enrichment function $\Phi(\boldsymbol{x},t)$ that represents the problem physics is defined as

$$\Phi_J(\boldsymbol{x},t) = \Phi(\boldsymbol{x},t) - \Phi(\boldsymbol{x}_J,t_J) \tag{2.46}$$

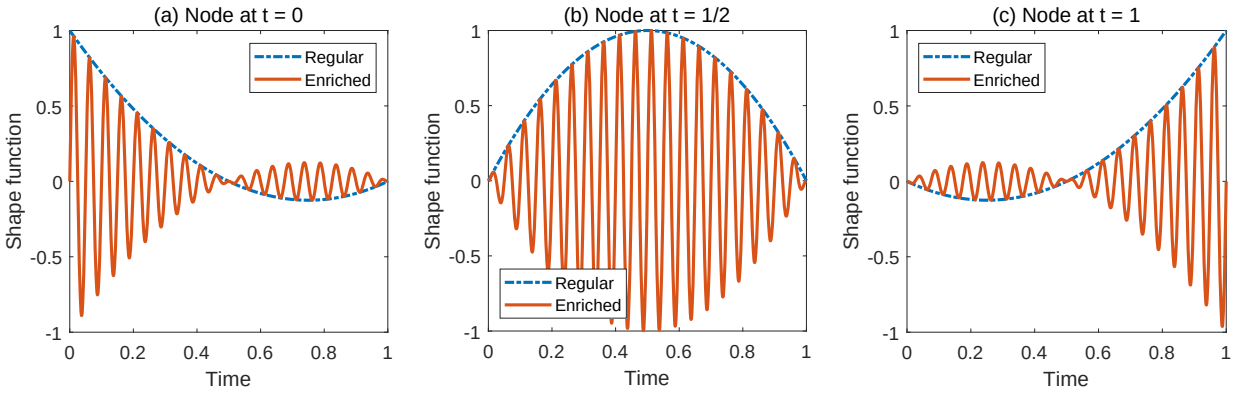


Figure 2.2. Comparison between regular and harmonic enriched temporal shape functions

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

$$\Phi_J(t) = \Phi(t) - \Phi(t_J) = \sin(\omega t) - \sin(\omega t_J) \tag{2.47}$$

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

Examples of the regular and the harmonic enriched temporal shape functions are shown in Figure 2.2. In this example, a quadratic function is chosen as the regular temporal shape function with three evenly spaced nodes on the time axis from 0 to 1. The enrichment function for the harmonic enriched temporal shape function is the same as Eq. (2.47) with a frequency of 20 Hz. It can be seen that this combination effectively captures both the slow and fast time scale components. This type of multiscale approximation enables rapid HCF simulations by using large time steps without compromising accuracy, which is critical for practical HCF applications. Previous studies by Yang et al. (Yang et al., 2012) and Bhamare et al. (Bhamare et al., 2014) have extensively demonstrated the capability of XTFEM in handling the multiple temporal scales for problems involving linear elastodynamics and concurrent multiscale simulations.

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

$$\boldsymbol{u}(\boldsymbol{x}, t) = \sum_{I} \boldsymbol{N}_I(\boldsymbol{x}, t)\boldsymbol{d}_I \tag{2.48}$$

which has the same form as the single-field formulation.

With the introduction of enrichment function, the XTFEM stiffness matrix can be generally expressed as

$$\boldsymbol{\mathcal{K}}_e = \begin{bmatrix} \boldsymbol{\mathcal{K}} & \boldsymbol{\mathcal{K}}_{ea} \\ \boldsymbol{\mathcal{K}}_{eb} & \boldsymbol{\mathcal{K}}_{ee} \end{bmatrix} \tag{2.49}$$

where $\boldsymbol{\mathcal{K}}$ is the same space-time matrix as in Eq. (2.40) of the single-field formulation, $\boldsymbol{\mathcal{K}}_{ea}$ and $\boldsymbol{\mathcal{K}}_{eb}$ reflect the coupling between enriched and regular DOFs, $\boldsymbol{\mathcal{K}}_{ee}$ reflects the coupling between enriched DOFs. Eq. (2.49) can also be written in the form of Kronecker product as shown in Eq. (2.41). However, the temporal submatrices are more complex due to the enrichment functions. In the current implementation, these temporal submatrices are also derived analytically.

2.5 Numerical implementation

Numerical implementation of STFEM or XTFEM generally follows the steps shown below:

1. Discretize the spatial domain the same way as in the standard FEM
2. Calculate the spatial stiffness and mass matrices using Gauss quadrature rules
3. Start the loop over space-time slabs (time stepping)
   (a) Calculate the temporal matrices based on either closed-form integration or numerical quadrature
   (b) Construct the space-time stiffness matrix
   (c) Calculate external force vector due to body force, traction, and the jump terms

(d) Apply essential boundary conditions

(e) Solve the stiffness matrix equation

(f) Evaluate strain, stress, damage, etc.

4. Post-processing

2.6 Numerical examples

2.6.1 Wave propagation in 1D bar

In the first example, we study a simple 1D bar with one end fixed while the other end subjected to constant stress. Closed-form solution to this problem can be found by using the d'Alembert's solution to the 1D wave equation under the given boundary conditions:

$$u(x,t) = \frac{\sigma_0}{E}\sum_{k=1}^{\infty}(-1)^{k-1}\left[\langle ct - x - 2(k-1)L\rangle - \langle ct + x - 2kL\rangle\right] \tag{2.50}$$

$$v(x,t) = \frac{\sigma_0 c}{E}\sum_{k=1}^{\infty}(-1)^{k-1}\left[H(ct - x - 2(k-1)L) - H(ct + x - 2kL)\right] \tag{2.51}$$

where σ_0 is the applied stress, E is the Young's modulus, $c = \sqrt{E/\rho}$ is the wave speed, L is the length of the bar, $\langle\cdot\rangle$ denotes the Macaulay bracket and $H(\cdot)$ is the Heaviside function. Note that $x = L$ is the fixed boundary while $x = 0$ is the boundary subjected to load.

Here we employ the traditional semi-discrete schemes, i.e. explicit central difference and implicit Newmark-β ($\gamma = 0.5, \beta = 0.25$) (Newmark, 1959) methods, as well as the TDG approaches (both single-field and two-field formulations) to simulate this problem. For simplicity, the parameters are chosen as $L = 1, E = 100, \rho = 1$, and $\sigma_0 = 1$. The spatial domain is uniformly discretized by 100 two-node linear bar element. The critical time step for central difference method is

$$\Delta t_c = L^e/c \tag{2.52}$$

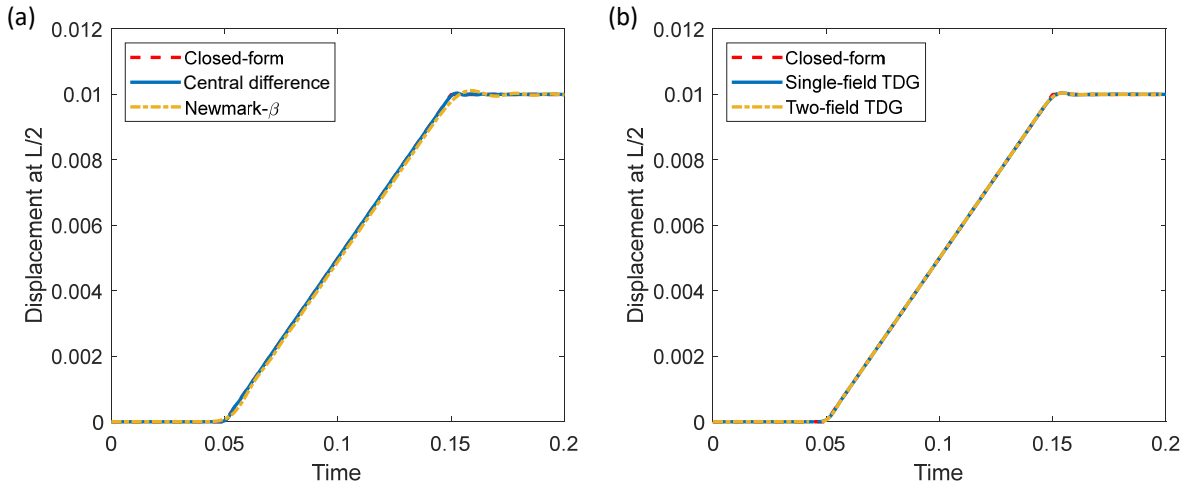


Figure 2.3. Displacement at $x = L/2$ from (a) semi-discrete schemes and (b) TDG approaches

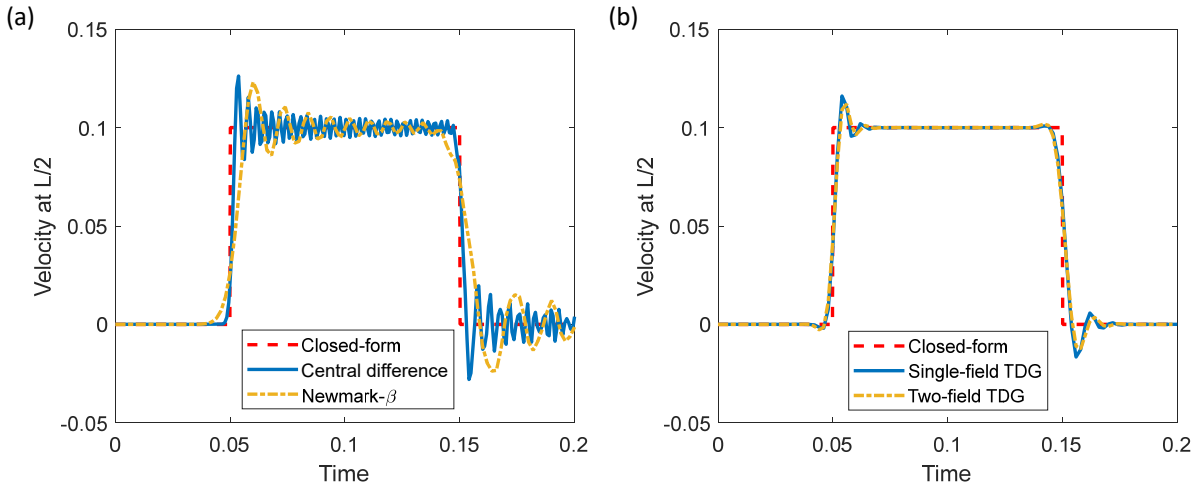


Figure 2.4. Velocity at $x = L/2$ from (a) semi-discrete schemes and (b) TDG approaches

where L^e is the element length. By substituting the parameters, the time step for central difference method is obtained as 8.5×10^{-4}. For the other methods we choose a time step size of 1.0×10^{-3} and the total simulation time is 0.2.

Displacement histories at $x = L/2$ obtained by the above methods are plotted in Figure 2.3. It can be seen that the displacement solutions from all the numerical methods agree well with the closed-form solution. However, as illustrated in Figure 2.4, significant artificial

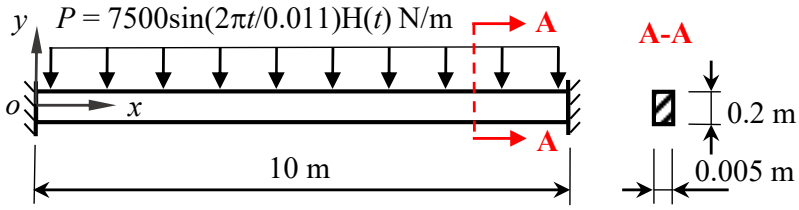


Figure 2.5. Dimensions and boundary conditions of the straight beam problem

oscillations near the temporal discontinuities in velocity history are observed in solutions obtained from the traditional semi-discrete schemes. On the contrary, TDG approaches significantly reduced such oscillations and the numerical solutions rapidly converged to the closed-form solution despite the large time step employed, which demonstrated exceptional accuracy and stability of the methods.

2.6.2 Transverse vibration in 2D beam

To demonstrate STFEM on 2D applications, we study the vibration of a straight beam under dynamic transverse loading. The material of the beam is assumed to be isotropic linear elastic and the material properties are given as density $\rho = 7860$ kg/m^3, Young's modulus $E = 200$ GPa and Poisson's ratio $\nu = 0.3$. Figure 2.5 shows the geometrical dimensions and boundary conditions of this example. The beam is clamped at both ends and subjected to a uniformly distributed cyclic load $P(t) = 7500\sin(2\pi t/0.011)H(t)$ N/m on the top surface. This problem is analyzed using the single-field STFEM. The spatial domain is modeled by plane stress formulation and discretized by 195 quadratic quadrilateral (Q9) elements. The temporal domain is divided into 460 space-time slabs with a time step of 5.0×10^{-4} s. The number of DOFs for this problem is 5,502.

For the purpose of verification, a closed-form solution (Clough and Penzien, 1993; Paz and Leigh, 2004) is obtained by utilizing the mode-superposition method, given as

$$y(x,t) = \sum_{n=1}^{\infty} \Phi_n(x) I_n \frac{P_0}{M\left(\omega_n^2 - \omega_f^2\right)} \left(-\frac{\omega_f}{\omega_n}\sin(\omega_n t) + \sin(\omega_f t) \right) \tag{2.53}$$

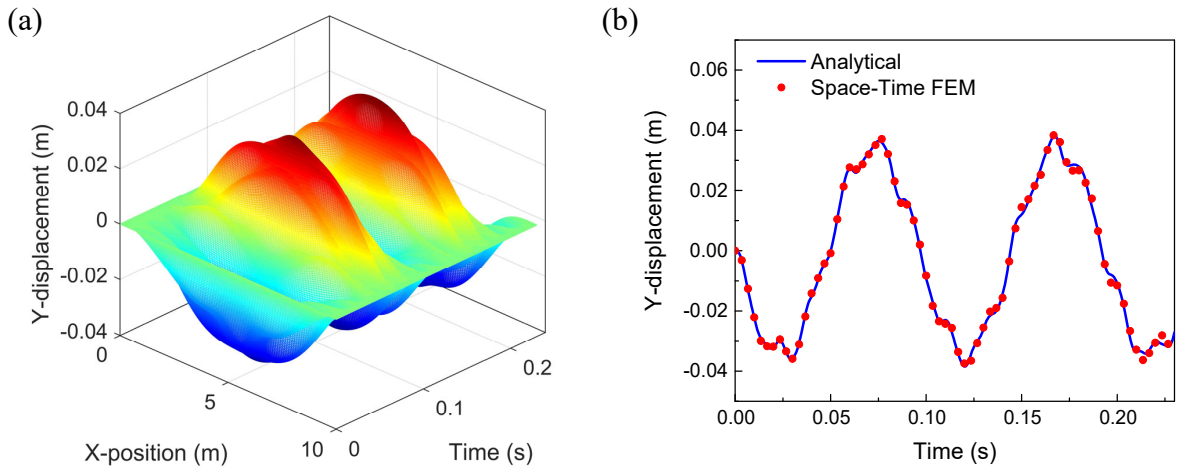


Figure 2.6. Space-time FEM results for straight beam problem: (a) transverse displacement at mid-surface vs. time; (b) transverse displacement at mid-point vs. time

where M is the mass per unit length, ω_f is the angular loading frequency and P_0 is the amplitude, and

$$I_n = \int_0^L \Phi_n(x)dx \Big/ \int_0^L \Phi_n^2(x)dx \tag{2.54}$$

in which L is the span of beam.

The mode shape is

$$\Phi_n(x) = \cosh(a_n x) - \cos(a_n x) - \sigma_n \left(\sinh(a_n x) - \sin(a_n x)\right) \tag{2.55}$$

where $\sigma_n = \left(\cos\left(a_n L\right) - \cosh\left(a_n L\right)\right) / \left(\sin\left(a_n L\right) - \sinh\left(a_n L\right)\right)$, the corresponding natural frequency is given by

$$\omega_n = a_n^2 \sqrt{EI/M} \tag{2.56}$$

in which I represents the second moment of area of the beam's cross-section, and finally a_n is the solution of

$$\cos(a_n L)\cosh(a_n L) = 1 \tag{2.57}$$

The analytical solution (Eq. 2.53) is approximated by using the first nine modes. Figure 2.6 shows the displacement responses at the neutral axis and at the centroid of the

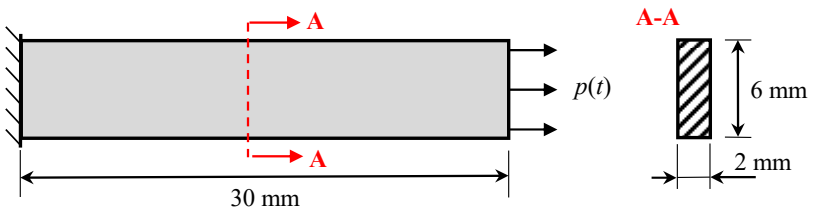


Figure 2.7. Dimensions and boundary conditions of the thin plate problem

beam. A good agreement between results obtained by the accelerated space-time FEM and the analytical solution is clearly demonstrated in Figure 2.6 (b).

2.6.3 Dynamic response of a 3D plate subjected to surface traction

In the last example, we study the problem of a rectangular thin plate with one end fixed and the other end subjected to a uniformly distributed surface traction $p(t)$. Detailed geometric dimensions and boundary conditions of the problem are illustrated in Figure 2.7. 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 8-node linear brick element with uniform element size of 0.5 mm, which leads to 2,880 elements and 3,965 nodes. To demonstrate the accuracy and robustness of the TDG methods, we compare the displacement and velocity solutions at the right end of the plate with that obtained from the semi-discrete schemes such as the explicit central difference method and the implicit Newmark-β. The TDG formulations, i.e. the single-field, two-field and enriched formulations are denoted as *TDG-1*, *TDG-2*, and *TDG-e*, respectively.

First, we consider the loading condition of a constant pressure, i.e. $p(t) = 100H(t)$ MPa. The time steps for central difference, Newmark-β, *TDG-1* and *TDG-2* are 1.2×10^{-9} s, 6.0×10^{-8} s, 1.2×10^{-6} s and 1.2×10^{-6} s, respectively.

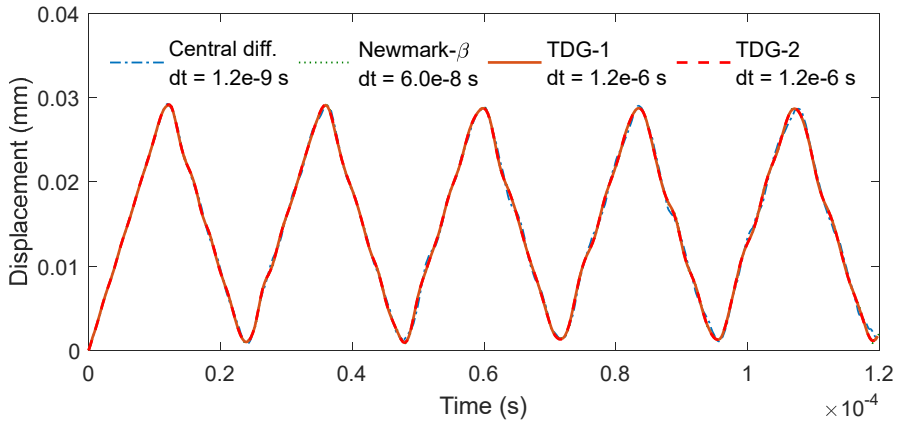


Figure 2.8. Comparison of the displacement solutions at the right end of the plate

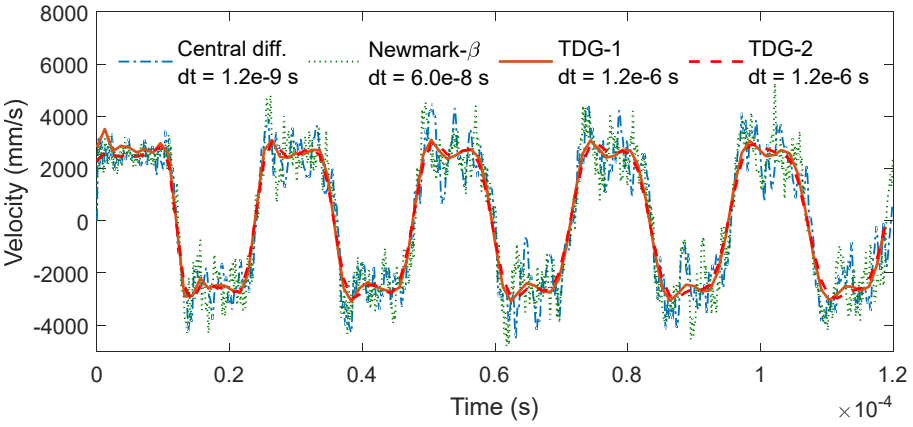


Figure 2.9. Comparison of the velocity solutions at the right end of the plate

Figure 2.8 shows the displacement solutions at the right end of the plate. Despite the large time steps that are used by the TDG methods, their displacement solutions show good agreement with that of the semi-discrete methods. The velocity solutions at the right end of the plate are shown in Figure 2.9. It can be seen that TDG methods significantly reduce the artificial oscillations near the discontinuities. Also, the TDG solutions are more stable than that of the semi-discrete schemes as the stress wave propagates in the plate.

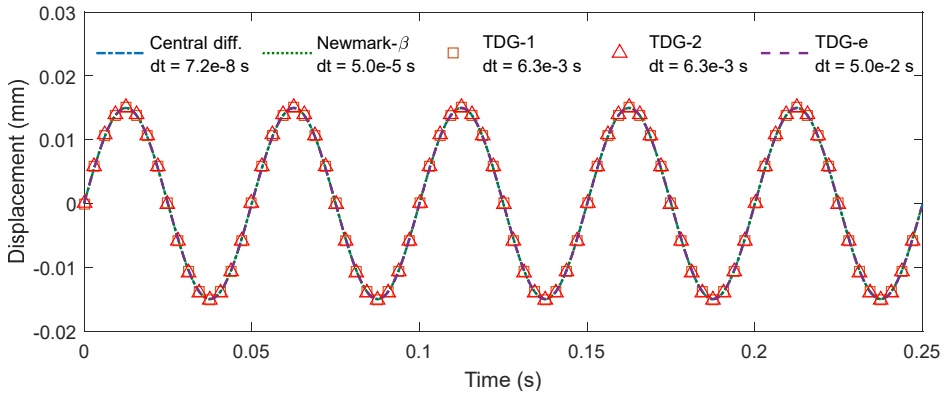


Figure 2.10. Comparison of the displacement solutions under cyclic load

Next, a fully-reversed cyclic load is considered, i.e. $p(t) = 100 \sin(40\pi t) H(t)$ MPa. The time steps for central difference, Newmark-β, *TDG-1*, *TDG-2*, and *TDG-e* methods are 7.2×10^{-8} s, 5.0×10^{-5} s, 6.3×10^{-3} s, 6.3×10^{-3} s, and 5.0×10^{-2} s, respectively. For the TDG-e method, a shifted sinusoidal function is employed to enrich the space-time shape function, i.e.,

$$\Phi_J(t) = \sin(\omega t) - \sin(\omega t_J) \tag{2.58}$$

in which the frequency $\omega = 40\pi$ is the same as the loading frequency. The displacement solutions are shown in Figure 2.10. Although TDG methods use very large time steps compared with semi-discrete methods, they accurately captured the oscillating structural response.

Finally, a cyclic load with dual modes is applied to further demonstrate the robustness of XTFEM. It contains a high frequency component at 20 Hz and a low frequency component at 1 Hz. The corresponding amplitudes are respectively 20 MPa and 100 MPa, i.e. $p(t) = 20 \sin(40\pi t) + 100 \sin(2\pi t)$ MPa. The displacement solutions are shown in Figure 2.11. It shows that XTFEM accurately captures both the low and high frequency components in the oscillating structural response without significantly reducing the time step.

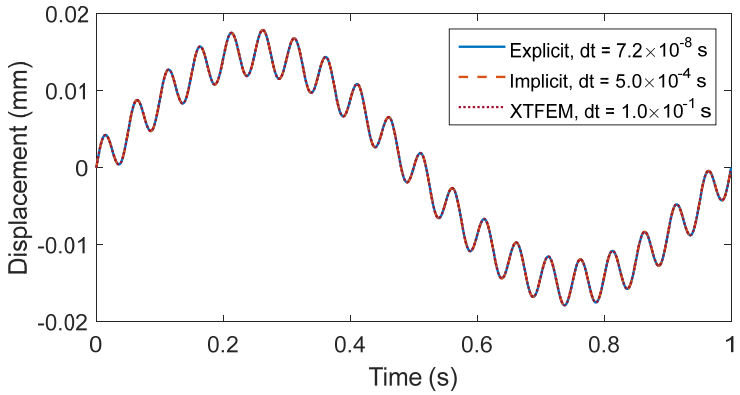


Figure 2.11. Comparison of semi-discrete and XTFEM solutions under complex cyclic load

2.7 Summary

In this chapter, we briefly reviewed the theory of TDG-based space-time FEM and its enriched version. Space-time stiffness matrices are derived for various TDG formulations under the assumptions of linear elastic material and small deformation. Those matrices exhibit a unique decomposition based on Kronecker product of temporal and spatial components, which can be further exploited to improve the efficiency in solving space-time stiffness matrix equations. Numerical examples range from 1D to 3D problems are provided to demonstrate the exceptional stability and accuracy of STFEM. The capability of XTFEM in capturing problem physics, e.g. oscillations in mechanical responses under fatigue loading, is also validated. More extensive demonstrations of the robustness of space-time FEM can be found in (Alpert, 2009; Bhamare, 2012; Alpert, 2013; Bhamare et al., 2014). Note that STFEM and XTFEM are also applicable for general cases involving nonlinear material behaviors and geometric nonlinearity, e.g. large deformation problems. Applications on rubber materials considering hyperelasticity, viscoelasticity and damage can be found in (Wada, 2017; Wada et al., 2018).

CHAPTER 3

EFFICIENT SOLUTION METHODS TO SPACE-TIME FEM

3.1 Introduction

Although space-time FEM shows excellent stability and accuracy over traditional semi-discrete FEM, as demonstrated in the previous chapter, the size of the space-time stiffness matrix equations are much larger due to the added dimension of time. Directly solving such a system of matrix equations leads to much higher computational cost than the standard FEM, especially for 3D and large problems in practical applications. Therefore, an efficient solution method to space-time FEM is critical and highly desired.

This chapter focuses on addressing the above issue and developing efficient solution methods. First, we conduct a detailed computational cost analysis of space-time FEM to identify the efficiency bottlenecks. Next, sparse matrix storage and operation algorithms are introduced. After that, we introduce the iterative solver for system of linear equations and discuss several conventional preconditioning techniques (Zhang et al., 2016). With these preparations ready, we first propose a novel Kronecker preconditioner based on the unique feature of space-time stiffness matrix that can significantly improve the computational performance of the iterative solver (Zhang et al., 2019). Furthermore, we propose an advanced hybrid iterative/direct sparse solver for space-time stiffness matrix equations based on the Kronecker

---

The following articles were reused in this chapter with permissions from the publishers:

1. Zhang, R., S. Naboulsi, T. Eason, and D. Qian (2019). A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing. *Finite Elements in Analysis and Design 166*, 103320. Reuse with permission from *Elsevier*.
2. Zhang, R., L. Wen, J. Xiao, and D. Qian (2019). An efficient solution algorithm for space-time finite element method. *Computational Mechanics 63*(3), 455–470. Reuse with permission from *Springer Nature*.
3. Zhang, R., L.Wen, S. Naboulsi, T. Eason, V. K. Vasudevan, and D. Qian (2016). Accelerated multiscale space-time finite element simulation and application to high cycle fatigue life prediction. *Computational Mechanics 58*(2), 329–349. Reuse with permission from *Springer Nature*.

preconditioning technique (Zhang et al., 2019). Finally, numerical examples are provided to demonstrate both the computational efficiency and performance of the proposed solution methods.

3.2 Computational cost analysis

Numerical implementations of space-time FEM are given in Section 2.5 and summarized by the pseudocode shown in Table 3.1. At first, spatial stiffness and mass matrices are formed outside the loop over the space-time slabs, which cost $\mathcal{O}(n_s)$ in time and $\mathcal{O}(n_s^2)$ in memory respectively, where n_s is the number of spatial DOFs. Let N denote the number of space-time DOFs, then $N = 3n_s$, $4n_s$, and $6n_s$ respectively for single-field, two-field and enriched space-time FEM. Within each space-time slab, assembly of the space-time matrix requires both $\mathcal{O}(N^2)$ in operations and storage if full matrix storage format is employed. Matrix-vector multiplication involved with the jump terms in right-hand-side vector costs $\mathcal{O}(N^2)$ in time while applying boundary conditions requires $\mathcal{O}(N)$ operations. Solution of the resulting space-time stiffness equations with direct solver, e.g. a Gaussian elimination type solver, requires $\mathcal{O}(N^3)$ in time. Finally, stress/strain calculation and postprocessing generally cost $\mathcal{O}(N)$ in time. Note that the cost of calculating the temporal matrices, even by numerical quadrature instead of analytical integration, is negligible due to its small size.

The above cost analysis clearly shows that, for space-time FEM implementation, the largest memory cost is of $\mathcal{O}(N^2)$ for the storage of space-time matrices. The most time-consuming part is the direct solution of the space-time stiffness equation, which requires $\mathcal{O}(N^3)$ in time. In space-time formulation, N accounts for both spatial and temporal discretizations, leading to the prohibitive computational expense.

To further illustrate this significant increase in matrix size, we consider here the example of a thin plate discussed in Section 2.6.3. For spatial discretization, a 3D structured mesh (2,880 8-node brick elements, 3,965 nodes) is generated and leads to the banded pattern of

Table 3.1. Pseudocode for space-time FEM

Line number	**Operation**
1	Discretize the spatial domain
2	Integrate and assemble spatial matrices $\boldsymbol{K}$ and $\boldsymbol{M}$
3	Loop over space-time slabs
4	Discretize the temporal domain
5	Integrate the temporal matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$
6	Assemble the space-time stiffness matrix $\boldsymbol{\mathcal{K}}$
7	Evaluate external force and jump term $\boldsymbol{\mathcal{F}}$
8	Apply essential boundary conditions
9	Solve $\boldsymbol{\mathcal{K}d} = \boldsymbol{\mathcal{F}}$
10	Evaluate strain, stress, etc.
11	End loop
12	Postprocessing

stiffness matrix for standard FEM as shown Figure 3.1 (a). The number of spatial DOFs in standard FEM is $n_s = 11,895$. The numbers of space-time DOFs are $N = 3n_s = 35,685$, $N = 4n_s = 47,680$ and $N = 6n_s = 71,370$ for single-field and two-field STFEM and XTFEM, respectively. Figure 3.1 (b) $\sim$ (d) illustrate the sparsity patterns from the stiffness matrices that are obtained from single-field and two-field STFEM and XTFEM for the same number of spatial nodes. Note that, in Figure 3.1, dashed boxes indicate the size of the standard FE stiffness matrix and nz represents the number of non-zero elements in the sparse matrix. As can be seen, direct solution of space-time stiffness matrix equation leads to computational cost that is several orders of magnitude higher comparing to that of solving the corresponding stiffness matrix equation in standard FEM, which becomes a critical barrier for its extensive and practical applications. Therefore, this computational analysis highlights the key role of efficient approaches for space-time FEM.

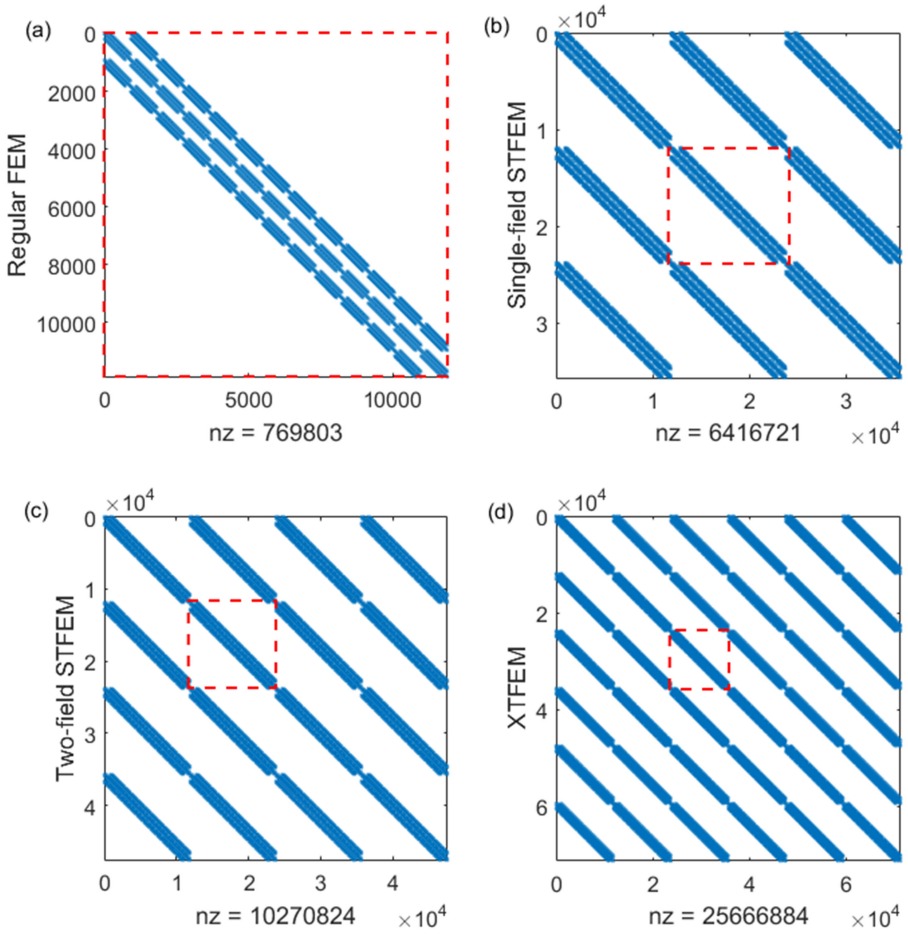


Figure 3.1. Comparison among the sparse pattern of the stiffness matrices formed by (a) standard FEM, (b) single-field STFEM, (c) two-field STFEM and (d) XTFEM

3.3 Sparse matrix storage and operations

Since the approximations are spatially localized and temporally nonlocal, the system matrices formulated by space-time FEM and its enriched version contain blocks of sparse matrices that are distributed non-sparsely as shown in Figure 3.1. By utilizing this particular property, the space-time matrices can be compressed to reduce storage, which is also commonly practiced in standard FEM. In this study, two of the most widely used sparse matrix storage formats are adopted, i.e., the Coordinate (COO) format and the Compressed Sparse Row (CSR) format, which are illustrated by an example shown in Figure 3.2. COO contains three one-

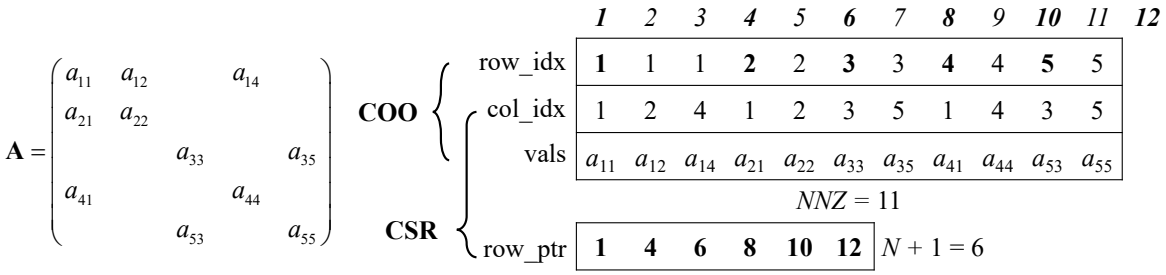


Figure 3.2. Illustrations of the COO and CSR format for storing a sparse matrix

dimensional arrays to store row indices, column indices and values of the nonzero entries respectively. The length of these arrays is the number of nonzero entries (NNZ). COO is well suited for incremental matrix construction and easy to access. Its storage can be further compressed to save more memory. In CSR format, the array of row indices in COO are compressed by array of row pointers, which is a list of indices where each row starts. Then the length of this array is reduced to the number of DOFs plus one, i.e., $(N+1)$. CSR is adopted by many sparse linear system solvers for its efficiency in arithmetic operations and matrix-vector multiplications.

Compared with COO, the CSR format is more expensive and complicated for operations that change the sparsity structure, such as matrix assembly. Thus, the spatial element matrices are first assembled into a global matrix in COO without summing up duplicate entries. Subsequently this triplet format is transformed into CSR by the following steps: (1) compressing the row indices; (2) sorting row and column indices; (3) summing up duplicate entries and (4) removing zero entries. The detailed sparse matrix manipulation algorithms involved in the above steps can be found in (Davis, 2006). Once spatial assembly finished, the number of nonzero entries in the space-time matrix can be easily estimated, which facilitates memory allocation. The spatial stiffness and mass matrices are further combined with the coefficients to form the space-time stiffness matrix in a COO format. Finally, the COO-formatted space-time stiffness matrix is transformed into a CSR format by the same

four steps mentioned above. By employing those sparse formats, not only the memory is saved, the time cost of matrix related operations, e.g. assembly and matrix-vector product, is also reduced to $\mathcal{O}(NNZ)$, where NNZ is $\sim \mathcal{O}(N)$ for large sparse matrices formulated by space-time FEM.

3.4 Iterative solver and preconditioning techniques

As we already shown in Chapter 2, for each space-time slab, the system of linear equations derived from TDG space-time FEM is given as

$$\boldsymbol{\mathcal{K}}\boldsymbol{d} = \boldsymbol{\mathcal{F}} \tag{3.1}$$

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

In standard 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 (Davis, 2006), which is robust and efficient particularly for equations with multiple right-hand-side vectors. The second type is iterative solver (Barrett et al., 1994; Saad, 2003), such as the *Conjugate Gradient* (CG) method and its preconditioned version (PCG). Iterative solvers are generally less robust for systems of linear equations that are not well conditioned but more flexible. They can achieve much higher efficiency than the direct sparse solver for particular problems.

In space-time FEM, stationary iterative solvers such as *Gauss-Jacobi* and *Gauss-Seidel* methods have been adopted to reduce the computational cost for the two-field formulation (Li and Wiberg, 1996; Chien and Wu, 2000; Kunthong and Thompson, 2005; Zhang et al., 2016), see (Zhang et al., 2019) for a brief review. Although the stationary iterative methods are simple to derive and easy to implement, their convergence may not good compared to the more general *Krylov subspace* methods. Examples of *Krylov subspace* methods including the above-mentioned CG and PCG iterative solvers. However, they are only applicable to

system of linear equations with symmetric coefficient matrix, which is the case in standard FEM but not true for space-time FEM. The generalization of CG to non-symmetric matrices leads to the *Biconjugate Gradient* (BiCG) method and eventually a more popular variant called *Biconjugate Gradient Stabilized* (BiCGSTAB) method (Vorst, 1992).

In this study, we also employ the *Krylov subspace* iterative approach to solve Eq. (3.1). It is well known that the efficiency and robustness of iterative methods largely depend on the quality of preconditioner (Barrett et al., 1994; Saad, 2003). Therefore, a preconditioner is constructed and the original system is modified as

$$\boldsymbol{\mathcal{P}}^{-1}\boldsymbol{\mathcal{K}}\boldsymbol{d} = \boldsymbol{\mathcal{P}}^{-1}\boldsymbol{\mathcal{F}} \tag{3.2}$$

in which matrix $\boldsymbol{\mathcal{P}}$ is the preconditioner and Eq. (3.2) is called left-preconditioned system.

3.4.1 Preconditioned iterative solver

Due to the asymmetry of space-time stiffness matrix, we employ the *Generalized Minimum Residual* (GMRES) method as the linear system solver, which was proposed by Saad and Schultz (Saad and Schultz, 1986) and widely used for system of linear equations with a non-symmetric coefficient matrix. The preconditioned GMRES algorithm we implemented here is summarized in Table 3.2.

The GMRES algorithm starts with an initial guess of the solution, i.e. $\boldsymbol{d}_0$, and the initial residual is calculated by $\boldsymbol{r}_0 = \boldsymbol{\mathcal{P}}^{-1}\left(\boldsymbol{\mathcal{F}} - \boldsymbol{\mathcal{K}}\boldsymbol{d}_0\right)$. The exact solution is then approximated by

$$\boldsymbol{d}^{(i)} = \boldsymbol{d}^{(0)} + y_1\boldsymbol{v}^{(1)} + \cdots + y_i\boldsymbol{v}^{(i)} \tag{3.3}$$

in which $\boldsymbol{v}^{(i)}$ are orthonormal bases of the left-preconditioned Krylov subspace of order m,

$$\mathcal{K}_m(\boldsymbol{\mathcal{P}}^{-1}\boldsymbol{\mathcal{K}}, \boldsymbol{r}_0) = span\left\{\boldsymbol{r}_0, \boldsymbol{\mathcal{P}}^{-1}\boldsymbol{\mathcal{K}}\boldsymbol{r}_0, \left(\boldsymbol{\mathcal{P}}^{-1}\boldsymbol{\mathcal{K}}\right)^2\boldsymbol{r}_0, ..., \left(\boldsymbol{\mathcal{P}}^{-1}\boldsymbol{\mathcal{K}}\right)^{m-1}\boldsymbol{r}_0\right\} \tag{3.4}$$

and y_i is solved from a linear least-squares problem which minimizes the residual. The bases are generated by modified Gram-Schmidt orthogonalization procedure in the Arnoldi

Table 3.2. Pseudocode for left-preconditioned GMRES algorithm

Line number	Operation
1	Given $\boldsymbol{\mathcal{P}}$, $\boldsymbol{\mathcal{K}}$, $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{d}_0$
2	**DO** $k = 1, 2, ...$
3	Solve $\boldsymbol{r}_0$ from $\boldsymbol{\mathcal{P}}\boldsymbol{r}_0 = \boldsymbol{\mathcal{F}} - \boldsymbol{\mathcal{K}}\boldsymbol{d}_0$
4	Calculate $\beta = \left\|\boldsymbol{r}_0\right\|_2$ and $\boldsymbol{v}^{(1)} = \boldsymbol{r}_0/\beta$
5	**DO** $j = 1, 2, ..., m$
6	Solve $\boldsymbol{w}$ from $\boldsymbol{\mathcal{P}}\boldsymbol{w} = \boldsymbol{\mathcal{F}} - \boldsymbol{\mathcal{K}}\boldsymbol{v}^{(j)}$
7	**DO** $i = 1, 2, ..., j$
8	Calculate $h_{i,j} = \left(\boldsymbol{w},\boldsymbol{v}^{(i)}\right)$ and $\boldsymbol{w} = \boldsymbol{w} - h_{i,j}\boldsymbol{v}^{(i)}$
9	**END DO**
10	Calculate $h_{j+1,j} = \left\|\boldsymbol{w}\right\|_2$ and $\boldsymbol{v}^{(j+1)} = \boldsymbol{w}/h_{j+1,j}$
11	**END DO**
12	Define $\boldsymbol{V}_m = \left[\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(m)}\right]$, $\bar{\boldsymbol{H}}_m = \left\{h_{i,j}\right\}_{1\leqslant i\leqslant j+1; 1\leqslant j\leqslant m}$
13	Solve $\boldsymbol{y}_m$ from $\arg\min_{\boldsymbol{y}}\left\|\beta\boldsymbol{e}_1 - \bar{\boldsymbol{H}}_m\boldsymbol{y}\right\|_2$
14	Calculate $\boldsymbol{d}_m = \boldsymbol{d}_0 + \boldsymbol{V}_m\boldsymbol{y}_m$
15	**IF** $\boldsymbol{d}_m$ is satisfied **THEN**
16	**STOP**
17	**ELSE**
18	Set $\boldsymbol{d}_0 = \boldsymbol{d}_m$
19	**END IF**
20	**END DO**

iteration. The iterative procedure is stopped once the residual satisfies the convergence criteria.

To save storage and computing resources, the GMRES is restarted every m iterations. Compared to direct solver, GMRES reduces the computational complexity to $\mathcal{O}(N^2)$ for solving general dense matrix systems while matrix-vector multiplication in every iteration is the most time-consuming part for large system of linear equations. Furthermore, for the sparse systems in our case, the computational complexity can be further reduced to $\mathcal{O}(NNZ)$ as discussed in Section 3.3.

Overall computational cost of GMRES solver depends on two factors. The first is the number of iterations that is required to satisfy the convergence criterion. The second is the

cost of each iteration. The total cost is the product of these two factors. Therefore, good preconditioners need to be constructed to minimize both the number of iterations and the most computing-intensive operations, i.e. the matrix operations as outlined in lines 3 and 6 in Table 3.2.

3.4.2 Conventional preconditioners

In general, preconditioners are built by approximating the space-time stiffness matrix or its inverse. For example, the simplest Jacobi preconditioner consists only the diagonal of the matrix. A good preconditioner makes the modified system of linear equations easier to solve and the computational saving gained far outweighs the extra cost of preconditioning itself. There are many preconditioners readily available, such as *Block Jacobi*, *Successive Over-Relaxation* (SOR), *Sparse Approximate Inverse* (SPAI), *Incomplete Factorization*, etc.

As an illustration of preconditioning techniques, here we employ a preconditioner based on the incomplete factorization of the space-time stiffness matrix. Note that incomplete factorization itself is a broad class of preconditioners. For example, *incomplete Cholesky factorization* (ICHOL) of symmetric positive definite (SPD) matrix is widely employed as a preconditioner for PCG solver. Since the space-time stiffness matrix is non-symmetric, we employ the more general *incomplete LU factorization* (ILU) method as the preconditioner for GMRES solver.

The ILU factorization of space-time stiffness matrix $\boldsymbol{\mathcal{K}}$ is given as

$$\boldsymbol{\mathcal{K}} = \boldsymbol{LU} - \boldsymbol{R} \tag{3.5}$$

where $\boldsymbol{L}$ is the lower unitriangular matrix, $\boldsymbol{U}$ is the upper triangular matrix and $\boldsymbol{R}$ is the residual matrix. During factorization, some entries in the original matrix will change from zero to nonzero values. This is generally referred to as "fill-in" and the preconditioning matrix is thus less sparse. Based on the fill-in control strategies, two variants of the ILU preconditioner are adopted:

1. ILUTP preconditioner, where "T" stands for threshold strategy and "P" stands for column pivoting. It is one of the most robust and efficient ILU preconditioners. In ILUTP, a non-negative scalar parameter *droptol* sets the threshold for dropping small terms in the $\boldsymbol{L}$, $\boldsymbol{U}$ factors and another tolerance ratio parameter *permtol* is used to determine whether or not to permute two columns. The columns are permuted to avoid failure of the factorization when encountering a zero pivot, and furthermore, to improve the stability of resulting preconditioner.
2. ILU(0) preconditioner, where "0" means fill-in is not allowed. It is the simplest and cheapest ILU preconditioner. However, it is not accurate and generally not recommend for practical problems. We employ it here only for the purpose of comparison.

3.4.3 Stiffness matrix reordering/permutation

While ILUTP preconditioner can efficiently reduce the number of iterations to a converge threshold required by GMRES, however, it often leads to much less sparse $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices, which makes the overall cost too expensive. Thus, we utilize matrix reordering algorithms based on the Graph theory to minimize the fill-in during factorization, which is commonly practiced in direct sparse solution methods. Computational complexity of finding the optimal reordering is NP-complete, thus heuristic algorithms are used instead. Two reordering algorithms with an $\mathcal{O}(NNZ)$ cost, namely, the Approximate Minimum Degree (AMD) algorithm (Amestoy et al., 1996) and the Reversed Cuthill-McKee (RCM) algorithm (Chan and George, 1980; George et al., 1994) are adopted here.

Here we illustrate the effects of these two reordering algorithms by an example shown in Figure 3.3, and their numerical performances will be demonstrated later. The space-time matrix formulated by XTFEM for the 2D problem described in Section 2.6.2 is employed as an input. Its sparsity pattern with original ordering is shown in Figure 3.3 (a). The results of matrix permutation by AMD and RCM algorithms are respectively shown in Figure 3.3

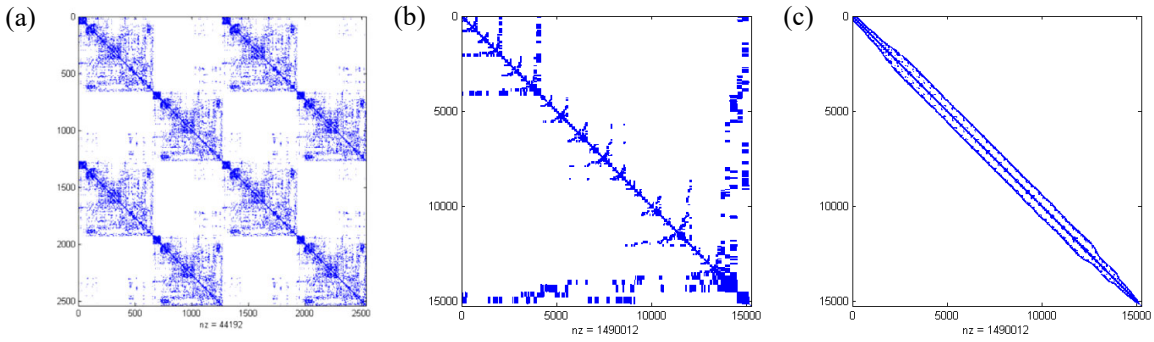


Figure 3.3. Illustrations of matrix reordering: (a) the original matrix and the permuted matrices by (b) AMD and (c) RCM reordering algorithms

(b) and (c). It is worth noting that the entries in the matrix are clustered around the main diagonal for RCM ordering since it is designed to reduce the bandwidth. A study of the effects of reordering algorithms used in implicit FEM for structural mechanics problems indicates that AMD ordering is more suitable to ill-posed problems such as thin shell elements or poor quality mesh, while RCM ordering shows better performance for the hexahedral topology problem (Kilic, 2012). Since the output of these reordering algorithms depends solely on the sparsity pattern (graph) of the system matrix, which is determined by the connectivity of FE mesh, the calculation of permutation vector is only required at the first time increment or after the deletion of damaged elements in HCF applications.

In summary, the conventional iterative solution approach introduced here solves modified space-time stiffness equation (3.2) by the following steps:

1. Permute the original system matrix by either AMD or RCM algorithm;
2. Factorize the reordered matrix by ILU method;
3. Solve the system of linear equations by GMRES and
4. Permute the solution vector back to the original ordering.

3.5 A novel Kronecker preconditioner

3.5.1 The Kronecker preconditioner

Since the choice and quality of the preconditioner greatly depend on specific system of linear equations, further improvements can be made by taking advantage of the special features of the space-time stiffness matrix. Considering the unique block structure of space-time matrix, Eq. (3.1) can be rewritten as

$$(\boldsymbol{\Phi} \otimes \boldsymbol{K} + \boldsymbol{\Psi} \otimes \boldsymbol{M})\, \boldsymbol{d} = \boldsymbol{\mathcal{F}} \tag{3.6}$$

Several observations can be made from the space-time stiffness matrix shown above and those derived in Sections 2.3 and 2.4:

1. The determinant of the matrix $\boldsymbol{\Psi}$, i.e., the temporal matrix corresponding to spatial mass matrix $\boldsymbol{M}$, is always zero. Since the determinant of Kronecker product of two matrices of $\boldsymbol{A}$ and $\boldsymbol{B}$ is given as $|\boldsymbol{A}_{n\times n} \otimes \boldsymbol{B}_{m\times m}| = |\boldsymbol{A}|^m |\boldsymbol{B}|^n$, the contribution from $\boldsymbol{M}$ to the space-time stiffness matrix $\boldsymbol{\mathcal{K}}$, i.e. the matrix $\boldsymbol{\Psi} \otimes \boldsymbol{M}$, is always singular;
2. The spatial stiffness matrix $\boldsymbol{K}$ is also singular. However, this singularity is eliminated by introducing the essential boundary conditions. Thus, the component derived from $\boldsymbol{K}$ matrix, i.e. the matrix $\boldsymbol{\Phi} \otimes \boldsymbol{K}$, is non-singular;
3. The matrix $\boldsymbol{\Phi} \otimes \boldsymbol{K}$ is the dominant component in the space-time stiffness matrix for most computational mechanics applications since the elements in $\boldsymbol{K}$ are usually several orders of magnitude larger than those of $\boldsymbol{M}$.

According to these observations, a block-structured preconditioner is constructed by approximating the dominant component in space-time matrix $\boldsymbol{\mathcal{K}}$, i.e. the matrix $\boldsymbol{\Phi} \otimes \boldsymbol{K}$. It is defined as

$$\boldsymbol{\mathcal{P}} = \boldsymbol{\Phi} \otimes \boldsymbol{P} \tag{3.7}$$

in which matrix $\boldsymbol{P}$ is derived by incomplete factorization of matrix $\boldsymbol{K}$. Since the spatial stiffness matrix $\boldsymbol{K}$ is SPD, both ICHOL and ILU methods mentioned in Section 3.4.2 can be used. Matrix $\boldsymbol{P}$ can be then expressed as

$$\boldsymbol{P} = \boldsymbol{L}\boldsymbol{U} = \boldsymbol{K} - \boldsymbol{R} \qquad \text{for ILU} \tag{3.8}$$

$$\boldsymbol{P} = \boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{K} - \boldsymbol{R} \qquad \text{for ICHOL} \tag{3.9}$$

where $\boldsymbol{R}$ is the residual matrix.

The advantages of using the preconditioner as proposed by Eq. (3.7) are twofold. First, the preconditioner is obtained by incomplete factorization of spatial stiffness matrix $\boldsymbol{K}$ which is SPD and better conditioned. Therefore, the computational cost is far less than the conventional approach, as shown in Section 3.4, of obtaining the preconditioner by incomplete factorization of the larger, coupled space-time matrix $\boldsymbol{\mathcal{K}}$ that is also non-symmetric. Second, the preconditioning operation, i.e., $\boldsymbol{\mathcal{P}}^{-1}\boldsymbol{v}$ with $\boldsymbol{v}$ an arbitrary vector involved in iterations, can be greatly simplified and accelerated by using properties of Kronecker product as shown later.

As mentioned in Section 3.4.1, the most computationally-intensive part of GMRES iterations is the following matrix operation

$$\boldsymbol{w} = \boldsymbol{\mathcal{P}}^{-1}\left(\boldsymbol{\mathcal{K}}\boldsymbol{v}\right) \tag{3.10}$$

which can be further decomposed into two matrix operations. The first is the matrix-vector multiplication of $\boldsymbol{\mathcal{K}}\boldsymbol{v}$. The second is the preconditioning operation $\boldsymbol{\mathcal{P}}^{-1}\boldsymbol{u}$, in which $\boldsymbol{u} = \boldsymbol{\mathcal{K}}\boldsymbol{v}$. Direct evaluations of these matrix operations are not efficient. They can be optimized and accelerated by using the block structure of the space-time stiffness matrix and the inverse property of Kronecker product as follows.

3.5.2 Optimization of matrix-vector multiplication

The general form of matrix-vector multiplication is

$$\boldsymbol{y} = \mathcal{K}\boldsymbol{x} \tag{3.11}$$

in which $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of size $N = n_t \times n_s$, where n_t and n_s are the dimensions of temporal and spatial matrices respectively. Note that n_t is usually much smaller than n_s. For example, $n_t = 3$, 4, and 6 for single-field, two-field STFEM and XTFEM, respectively.

By using the block structure of space-time matrix, Eq. (3.11) is recast as

$$\boldsymbol{y} = (\boldsymbol{\Phi} \otimes \boldsymbol{K} + \boldsymbol{\Psi} \otimes \boldsymbol{M})\, \boldsymbol{x} \tag{3.12}$$

The first step to simplify Eq. (3.12) is dividing vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ into smaller segments,

$$\boldsymbol{x} = \begin{Bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(n_\mathrm{t})} \end{Bmatrix} \quad \text{and} \quad \boldsymbol{y} = \begin{Bmatrix} \boldsymbol{y}^{(1)} \\ \boldsymbol{y}^{(2)} \\ \vdots \\ \boldsymbol{y}^{(n_\mathrm{t})} \end{Bmatrix} \tag{3.13}$$

Second, we introduce two intermediate vectors

$$\boldsymbol{u}^{(i)} = \boldsymbol{K}\boldsymbol{x}^{(i)} \quad \text{and} \quad \boldsymbol{v}^{(i)} = \boldsymbol{M}\boldsymbol{x}^{(i)}, \qquad i = 1, 2, ..., n_\mathrm{t} \tag{3.14}$$

Finally, the desired vector $\boldsymbol{y}$ is computed as

$$\boldsymbol{y}^{(i)} = \sum_{j=1}^{n_\mathrm{t}} \phi_{ij}\boldsymbol{u}^{(j)} + \sum_{j=1}^{n_\mathrm{t}} \psi_{ij}\boldsymbol{v}^{(j)}, \qquad i = 1, 2, ..., n_\mathrm{t} \tag{3.15}$$

It can be shown that direct evaluation of the matrix-vector product in Eq. (3.11) requires approximately $2N^2$ arithmetic operations, including $N(N-1)$ additions and N^2 multiplications, and $\mathcal{O}(N^2)$ memory assuming a full matrix storage scheme, where $N = n_t n_s$ is the size of space-time matrix. In the optimized implementation, the matrix-vector products in Eq. (3.14) require $4n_t n_s^2$ arithmetic operations and Eq. (3.15) requires only $\mathcal{O}(n_s)$

operations. The space and time matrices in Eqs. (3.14) and (3.15) require a storage cost of $\mathcal{O}(2n_s^2)$. Therefore, both operations and storage cost of the optimized implementation are reduced to only $2/n_t$ and $2/n_t^2$ of the direct evaluation method, respectively.

3.5.3 Optimization of preconditioning operation

The preconditioning operation is generally expressed as

$$\boldsymbol{y} = \boldsymbol{\mathcal{P}}^{-1}\boldsymbol{x} \tag{3.16}$$

in which $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of size $N = n_t \times n_s$.

By using the preconditioner defined in Eq. (3.7), we have

$$\boldsymbol{y} = (\boldsymbol{\Phi} \otimes \boldsymbol{P})^{-1}\boldsymbol{x} \tag{3.17}$$

The inverse of a Kronecker product is given by $(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$ if and only if both $\boldsymbol{A}$ and $\boldsymbol{B}$ are invertible. By using this property, Eq. (3.17) is further simplified as

$$\boldsymbol{y} = \left(\boldsymbol{\Phi}^{-1} \otimes \boldsymbol{P}^{-1}\right)\boldsymbol{x} \tag{3.18}$$

It can be seen that the inverse of the larger matrix $\boldsymbol{\mathcal{P}}$ is now replaced by the inverses of two smaller matrices $\boldsymbol{\Phi}$ and $\boldsymbol{P}$. The cost of inverting temporal matrix $\boldsymbol{\Phi}$ is negligible since its size is very small ($n_t \times n_t$). However, inverting matrix $\boldsymbol{P}$ explicitly can be very expensive for practical applications. To avoid that, we further decompose the evaluation of Eq. (3.18) into the following two steps:

First, we introduce an intermediate vector $\boldsymbol{z}$ that can be solved from

$$\boldsymbol{z}^{(i)} = \boldsymbol{P}^{-1}\boldsymbol{x}^{(i)}, \quad i = 1, 2, ..., n_\mathrm{t} \tag{3.19}$$

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

Then, the desired vector $\boldsymbol{y}$ is obtained by

$$\boldsymbol{y}^{(i)} = \sum_{j=1}^{n_\mathrm{t}} \varphi_{ij}\boldsymbol{z}^{(j)}, \quad i = 1, 2, ..., n_\mathrm{t} \tag{3.20}$$

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

In those two steps, Eq. (3.19) is the most computationally-intensive one. Cost of implicitly generating inverse of the preconditioner matrix $\boldsymbol{P}^{-1}$ mostly comes from incomplete factorization of the $\boldsymbol{K}$ matrix. It depends not only on the choice of incomplete factorization algorithm but also on property of the coefficient matrix to be factorized. However, this operation is the same as its counterpart in solving the corresponding static problem with standard FEM assuming that the same type of preconditioned iterative solver is used. Therefore, computational cost of preconditioner is reduced to the same order as the standard FEM.

As a brief summary of this section, we have demonstrated theoretically that the proposed Kronecker preconditioning algorithm reduces the computational cost of solving space-time linear system of equations to the same order of solving the corresponding static problems in standard FEM with iterative method. Furthermore, numerous matrix operations involved in this method are well-suited for parallel computing.

3.6 A hybrid iterative/direct sparse solver

3.6.1 Direct sparse solver for preconditioning

To further improve the numerical efficiency of the proposed Kronecker preconditioning approach and develop parallel implementation, the dominant component of space-time stiffness matrix, i.e. $\boldsymbol{\Phi}\otimes\boldsymbol{K}$, is employed in this section directly as the preconditioner instead of $\boldsymbol{\Phi}\otimes\boldsymbol{P}$ since the former already provides a good approximation of the space-time stiffness matrix. Therefore, we propose that

$$\boldsymbol{\mathcal{P}} = \boldsymbol{\Phi} \otimes \boldsymbol{K} \tag{3.21}$$

With this new preconditioner, the process of evaluating matrix $\boldsymbol{P}$ in Eq. (3.7) is completely avoided and computational effort is thus saved. Another key feature of the new

preconditioner is that the direct sparse solver can now be introduced into the preconditioning operation, which improves its robustness and efficiency. The new solution approach essentially forms an advanced hybrid iterative/direct sparse solver that takes advantage of both the efficiency of iterative method and the robustness of direct method.

The main difference of the hybrid method compared to the iterative method presented in Section 3.5 is that Eq. (3.19) can now be replaced by

$$\boldsymbol{z}^{(i)} = \boldsymbol{K}^{-1}\boldsymbol{x}^{(i)}, \quad i = 1, 2, ..., n_{\mathrm{t}} \tag{3.22}$$

which is equivalent to solving the corresponding standard FE static stiffness matrix equations. Hence, by using the proposed method, the computational cost of preconditioning is further reduced to the same as solving the spatial stiffness matrix equations in linear static FEM analysis with a direct sparse solver. Therefore, Eq. (3.22) can be handled efficiently by many well-established direct sparse solvers employed by standard FEM. It is also possible to use commercial FE packages to solve Eq. (3.22) and implement space-time FEM.

In this work, a parallel multi-frontal direct sparse solver named MUMPS (Amestoy et al., 2001, 2006) is employed to solve Eq. (3.22). In general, the direct sparse method solves a system of linear equations 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 these two steps are completed, the final solution step can be performed separately and repeatedly for the case of multiple right-hand-side vectors. Therefore, for linear elastodynamics the analysis and factorization steps of direct sparse solver is performed only once during the first time-increment. For nonlinear HCF simulations, the same solution strategy is employed unless the preconditioner $\boldsymbol{\Phi} \otimes \boldsymbol{K}$ no longer provides a good approximation of the XTFEM stiffness matrix $\boldsymbol{\mathcal{K}}$ due to the progressive evolution of spatial matrix $\boldsymbol{K}$ due to fatigue damage. In that case, the convergence rate of the main iterative solver slows down and triggers the re-evaluation of the preconditioner.

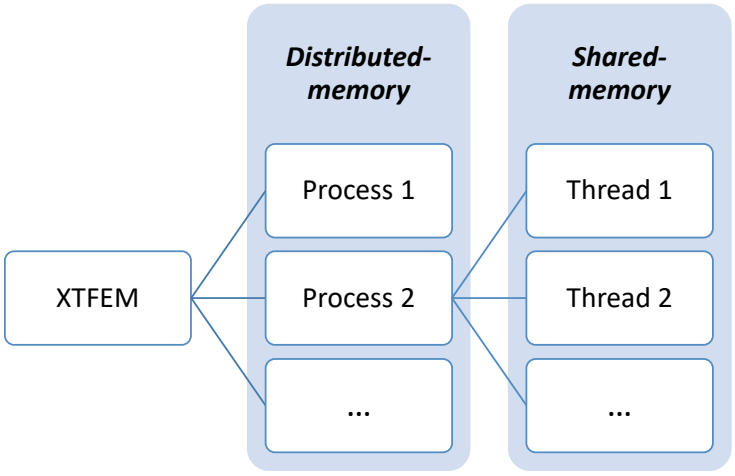


Figure 3.4. A two-level hierarchy of two types of parallelism

Thus, computational cost can be saved since the first two steps of the direct sparse solver are minimized.

3.6.2 HPC parallel implementation

To 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 sparse 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 Figure 3.4. In this way, parallel computing hardware can be used more efficiently.

3.6.2.1 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

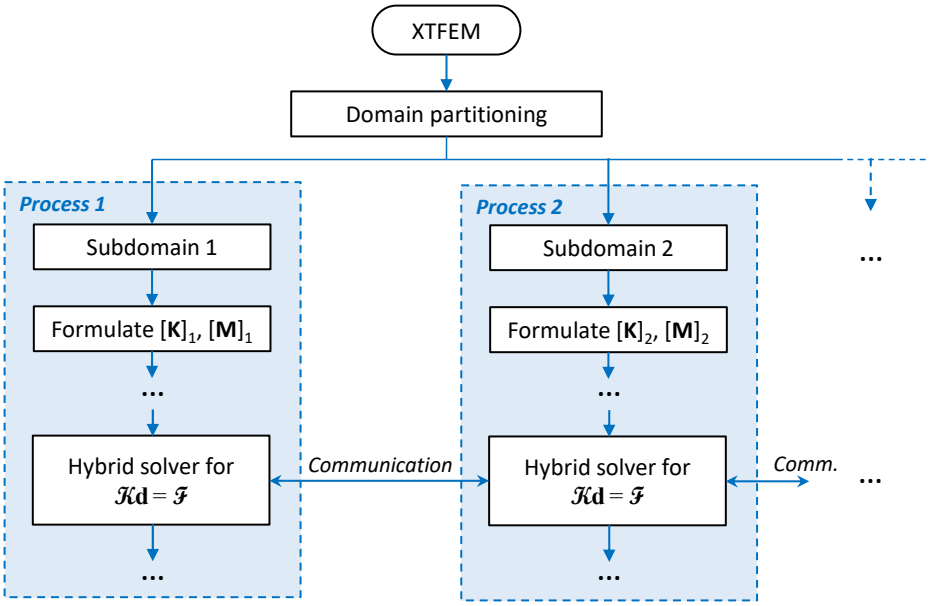


Figure 3.5. Distributed-memory parallelism of XTFEM

subdomain are then handled by a specific process as shown in Figure 3.5. Data on the interfacial nodes between subdomains are transferred through communication among processes using the *Message Passing Interface* (MPI) protocol (Gropp et al., 1999).

To achieve an 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 strategy 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 the di-

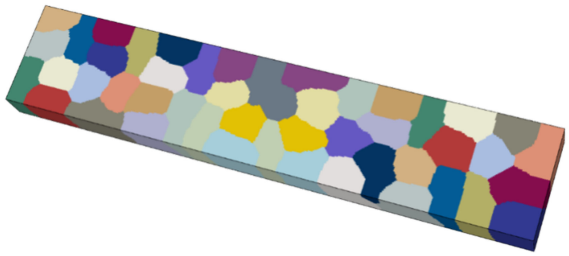


Figure 3.6. Domain partitioning by METIS

rect solver (Kumar et al., 1994). As such, domain partitioning is critical to the parallel implementation of the proposed hybrid solver.

Finite element mesh can be mathematically represented as graph based on connectivity. Partitioning such a graph is a NP-complete problem (Andreev and Räcke, 2004). Algorithms such as the spectral partitioning methods (Pothen et al., 1990, 1992; Hendrickson and Leland, 1995) and the multilevel spectral bisection (MSB) method (Barnard and Simon, 1994) are quite expensive though they produce a reasonably good partition. In this work, we employ a multilevel graph partitioning scheme developed by Karypis and Kumar (Karypis and Kumar, 1998a,b), also known as the METIS package. It partitions a graph using either the multilevel recursive bisection algorithm or the multilevel k-way algorithm. METIS is usually two orders of magnitude faster than the MSB algorithm and produces higher quality ordering that leads to substantially smaller fill-in than other frequently used algorithms (Karypis and Kumar, 1998a,b). Figure 3.6 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.

3.6.2.2 Shared-memory parallelism

As illustrated in Figure 3.7, the shared-memory parallelism of XTFEM is exploited from three computing-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)

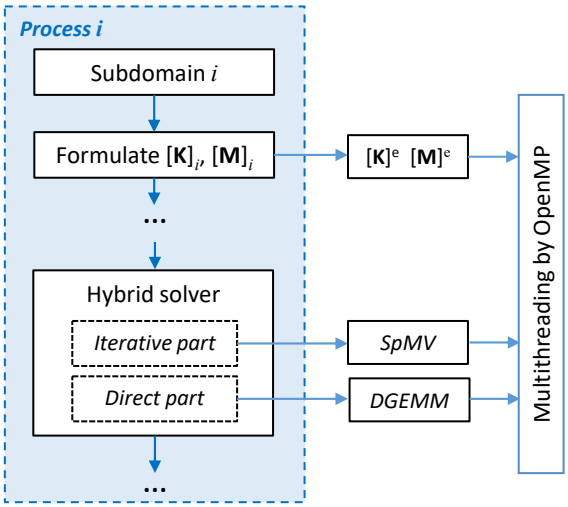


Figure 3.7. Shared-memory parallelism of XTFEM

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 COO sparse matrix. Then, the COO matrix is compressed to CSR matrix, which is the input format used for the hybrid solver. 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, the OpenMP (Dagum and Menon, 1998) 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 (Tian, 2014). Those subroutines are well-suited for SIMD-type 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

Table 3.3. Performance evolution of supercomputers (2009-2018)

Name	**Year**	**Peak performance (Tflops)**	**Memory bandwidth (GB/s)**	**Network bandwidth (GB/s)**
Summit[a]	2018	200,795	135.0	23
Titan[b]	2012	27,113	52.0	8
K[c]	2011	11,280	64.0	10
Jaguar[d]	2009	2,311	25.6	4

Sources: [a](ORNL, 2018a), [b](Tian, 2014; ORNL, 2018b), [c](RIKEN, 2018)
[d](Bland et al., 2010)

bound of the AMR of SpMV is 2:1 (flops/bytes) (Tian, 2014). Hence, SpMV subroutine is memory-intensive in the sense that its computing time is bounded by memory bandwidth of hardware rather than CPU speed. In contrast, DGEMM is a computing-intensive subroutine with an AMR of n:1, where n is the size of the dense frontal matrix (Tian, 2014).

Currently, newer HPC platforms typically have large AMR values and the trend of architecture evolves towards a higher AMR since improvement on processors (CPUs/GPUs) are usually much faster than that of memory/network bandwidth. For example, the peak performance of top supercomputer in the past decade (see Table 3.3) has 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 3.4). 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.

Table 3.4. Performance evolution of NVIDIA TESLA GPU (2009-2018)

Model	Year	DP performance (Tflops)	Memory bandwidth (GB/s)	flops/bytes
V100	2018	7,834	897	8.7
P100	2016	4,763	732	6.5
K40	2013	1,430	288	5.0
C2090	2011	656	177	3.7
S1070	2008	78	102	0.8

Source: (TechPowerUp, 2018)

3.7 Numerical examples

3.7.1 Performance of the conventional iterative solver

The problem statement of this example is given in Section 2.6.2 and here we mainly focus on evaluating the computational performance of stiffness matrix equation solvers. The spatial domain of this problem is discretized by 195 Q9 elements with plane stress formulation and the temporal domain is divided into 460 slabs. The total number of DOFs is 5,502.

The conventional iterative solver introduced in Section 3.4 is employed for this example and configured as follows: (1) AMD algorithm is used for reordering, (2) ILUTP parameters *droptol* and *permtol* are respectively set to 1.0e-5 and 0.01 as suggested in (Saad, 2003) and (3) GMRES convergence tolerance and the maximum number of iterations are set to 1.0e-8 and 1,000, respectively. The problem is solved with Intel Xeon E5-2667 CPU using single core. The total computing time used is about 8.0 seconds for 460 space-time slabs.

Next, we investigate the computational performance of different preconditioning and reordering techniques. Numerical accuracy, computational time and memory cost are the three metrics that are used to evaluate the performance. In terms of accuracy, the numerical error is calculated by L_2-norm with a reference solution obtained by the direct solver. The direct solver was implemented using the DGESV subroutine in Linear Algebra Package

library (LAPACK [1]), which has a computational complexity of $\mathcal{O}(N^3)$. Since the problem is linear elastic, reordering and preconditioning are only required for the first time-step, thus the total CPU time for the first step is chosen as the time cost metric. Due to its major contribution to storage cost, the ratio of the number of nonzero entries of $\boldsymbol{LU}$ factors to the number of nonzero entries of system matrix is defined as fill-in ratio to measure the memory usage.

For different preconditioners and reordering algorithms, the above performance metrics together with the number of iterations to converge are summarized in Table 3.5. Without any preconditioners, the GMRES solver fails to converge after it reaches the maximum number of iterations that allowed. Due to the large L_2-norm error finally obtained, it is unlikely to converge even with multiple restarts of iteration, which is typically not cost-effective. This also indicates that condition number of space-time stiffness matrix is high. By employing the simple ILU(0) preconditioner, solution converged after 342 iterations with a satisfied accuracy. Since fill-in is not allowed by ILU(0), the fill-in ratio equals one. The number of iterations is further reduced to 25 by utilizing ILUTP preconditioner. However, the time and memory cost are respectively increased almost 6 and 9 times compared with those from ILU(0). The increasing in computational cost can be effectively reduced by either AMD or RCM reordering algorithms in this example, which is clearly demonstrated by the performance metrics shown in Table 3.5.

Now we further study the influences of ILUTP parameters with AMD reordering algorithm. The *droptol* and *permtol* are chosen as [1.0e-3, 1.0e-4, 1.0e-5] and [0.0, 0.01, 0.5] respectively, thus totally nine sets of results are given in Table 3.6. Since the smaller *droptol* leads to a more accurate ILU factorization, the iteration number is significantly reduced as well as the CPU time, and memory cost is increased due to more fill-in's that are allowed. However, there is no obvious correlation between the accuracy of GMRES solution and ILU

[1] See `http://www.netlib.org/lapack` for details

Table 3.5. Performance comparison between preconditioners and reordering algorithms

Preconditioner	**Reordering**	**Iterations**	**L_2-norm error**	**Time (s)**	**Fill-in ratio**
None	None	1000	6.52e-01	5.25	n/a
ILU(0)	None	342	1.98e-07	1.46	1.0
ILUTP	None	25	3.06e-06	9.85	8.8
ILUTP	AMD	17	2.17e-07	0.15	1.2
ILUTP	RCM	21	2.02e-07	0.20	1.5

Table 3.6. Parameter study of the ILUTP preconditioner

droptol	***permtol***	**Iterations**	**L_2-norm error**	**Time (s)**	**Fill-in ratio**
	0.00	277	3.98e-05	1.25	1.08
1.0e-03	0.01	149	2.20e-06	0.55	1.04
	0.50	166	7.92e-06	0.63	1.06
	0.00	107	1.24e-04	0.47	1.20
1.0e-04	0.01	51	1.33e-06	0.24	1.16
	0.50	58	4.94e-07	0.25	1.16
	0.00	43	4.11e-05	0.22	1.22
1.0e-05	0.01	17	2.17e-07	0.15	1.18
	0.50	22	2.71e-07	0.16	1.18

factorization. Pivoting of columns clearly shows an improvement on both efficiency of the ILU preconditioner and accuracy of the GMRES solver. Values of *permtol* between 0.01 and 0.5 lead to similar results.

Finally, we study the performance of the conventional preconditioned iterative solution method in terms of its scaling with the number of DOFs, which is denoted by N. By refining the spatial discretization with a constant element aspect ratio, the maximum number of DOFs is obtained as almost 1 *million*. The results are shown in Figure 3.8. Performance of direct solver (LAPACK/DGESV) is also provided for the purpose of comparison. Time costs shown in Figure 3.8 (a) are the total CPU time for the first space-time slab. Since the computational cost of ILUTP preconditioner depends on system matrices, the overall

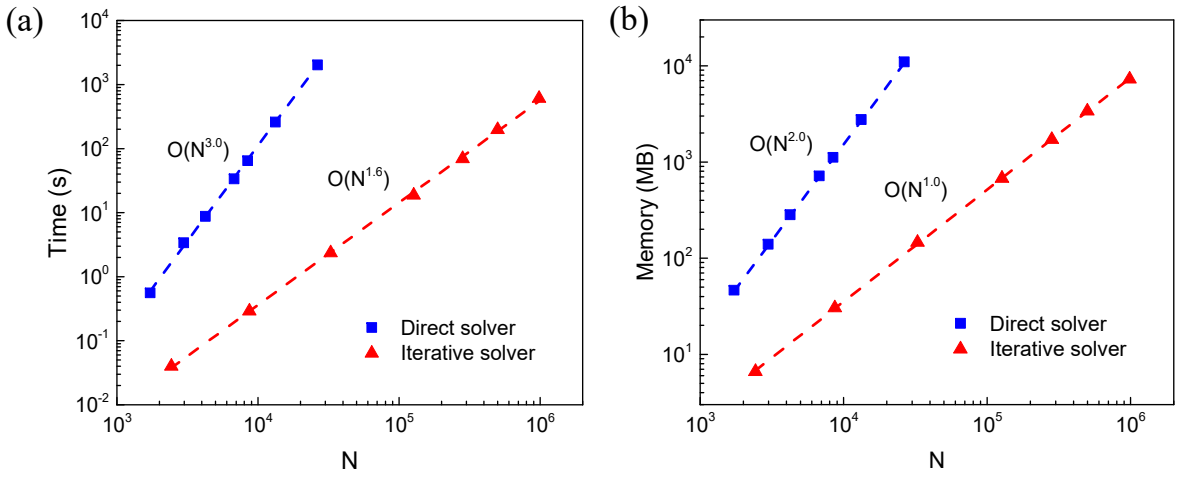


Figure 3.8. Performance of iterative and direct solvers: (a) time and (b) memory costs

time cost of the proposed method is problem-dependent. For this particular example, the iterative method achieves a time cost of $\mathcal{O}(N^{1.6})$, which is much smaller than the $\mathcal{O}(N^3)$ cost by the direct solver. By utilizing the sparse matrix format and reordering algorithms, the memory cost, which is mainly contributed by system matrix and preconditioner, is linearly proportional to the number of DOFs as shown in Figure 3.8 (b). The memory cost is less than 10 Gigabytes for the largest problem with almost 1 *million* unknowns. This computational performance is comparable to the case of standard FEM. Therefore, the conventional preconditioned iterative solution method enables solutions of relatively large 2D problems on a single PC using space-time FEM.

3.7.2 Performance of the Kronecker solver

The problem statement of the second example is given in Section 2.6.3 with the exception that both 2D and 3D cases are studied here with a focus on evaluating the computational performance of the proposed Kronecker iterative solution algorithm with the following configurations.

First of all, both 2D and 3D discretizations are established for the spatial domain of the thin plate for comparison purposes. For 2D discretizations, bilinear quadrilateral (Q4) elements with plane stress formulation are employed. Eight-node hexahedral elements with full integration are used for 3D cases. Two sets of structured mesh grids are created by refining the element size. All the TDG formulations, i.e., *TDG-1*, *TDG-2* and *TDG-e*, are employed to discretize the temporal domain.

In addition the proposed Kronecker iterative algorithm, a direct sparse solver (UMFPACK[1], also a multi-frontal method (Davis, 2004)) and the conventional preconditioned iterative solver introduced in Section 3.4 are also employed for the purpose of comparison. For convenience, we denote the proposed method as the *Kronecker* solver, while the other two are denoted simply as the *Direct* and *Iterative* solvers. Note that reversed Cuthill-Mckee (RCM) reordering algorithm (Chan and George, 1980) and ILUTP preconditioning algorithm are used in both the Kronecker and Iterative solvers. Same parameters associated with these two iterative solvers are used, see the previous Section 3.7.1 or (Zhang et al., 2016) for details.

For comparison purposes, the same problem is also simulated in semi-discrete FEM with the Newmark-β method. Since Newmark-β method is an implicit time integration method, it formulates linear systems of equations with the spatial-only effective stiffness matrix, which is given by

$$\boldsymbol{K}_{\text{eff}} = \boldsymbol{K} + \frac{1}{\beta \Delta t^2} \boldsymbol{M} \tag{3.23}$$

The conventional preconditioned iterative solver is employed to solve the systems of linear equations with the stiffness matrix as shown in Eq. (3.23). Therefore, the result of theoretical

[1]UMFPACK is part of the SuiteSparse package developed by Professor Tim Davis at Texas A&M University (`http://faculty.cse.tamu.edu/davis/suitesparse.html`). As shown later, we did the tests in this section in MATLAB, which integrates the UMFPACK as a default solver for sparse linear systems in the form of a built-in routine for the backslash (\) and forward-slash (/) operators. To obtain and control the detailed parameters for direct sparse solver in MATLAB, the function `spparms` can be called (see `https://www.mathworks.com/help/matlab/ref/spparms.html` for details).

cost analysis in Section 3.5 can be validated by comparing the computational performance of TDG methods with that of the implicit Newmark-β method.

Both time and storage costs and complexities are presented as follows. The total CPU time of obtaining solutions to the first space-time slab (or equivalently the first time-step for Newmark-β method) is reported as the performance metric for time usage since preconditioner is generated at this stage only. The memories used by system matrix factorization, i.e. the symbolic and numerical factorizations in the *Direct* solver and the incomplete factorizations in the *Iterative* and *Kronecker* solvers, are reported as the performance metric for storage cost.

The hardware platform for performance testing is a desktop workstation featured with the Intel Xeon E5-2623v3 CPU and 32 GB RAM. The testing was done using MATLAB of version R2015b. Note that it was discovered in numerical experiments that with default options MATLAB automatically accelerates some operations, e.g. those associated with the direct sparse solver, by using multiple CPU threads. To avoid the automatic parallelization and ensure the same computing environment, the single thread mode was activated by launching MATLAB with the following command line option: `-singleCompThread`.

3.7.2.1 2D spatial discretization

We first evaluate the computational performance for the 2D cases. A summary of the 2D mesh and the corresponding numbers of DOFs are presented in Table 3.7.

Computational performances of different solvers for the single-field formulation STFEM are plotted in Figure 3.9. It shows that the conventional preconditioned *Iterative* solver achieves a similar efficiency as the multi-frontal *Direct* solver. The time complexities of these two solvers are both $\mathcal{O}(N^{1.6})$. The *Kronecker* solver demonstrates significantly better performance. The time complexity is reduced to $\mathcal{O}(N^{1.3})$. For larger cases, e.g. $N > 10^4$, the *Kronecker* solver is at least one order of magnitude faster and requires at least one order of magnitude less memory than the other solvers.

Table 3.7. 2D meshes of the thin plate

No.	Element size (mm)	# Elements	# Nodes	# Spatial DOFs (Newmark-β)	# Space-time DOFs		
					TDG-1	TDG-2	TDG-e
1	3.0	20	33	66	198	264	396
2	1.5	80	105	210	630	840	1,260
3	0.75	320	369	738	2,214	2,952	4,428
4	0.375	1,280	1,377	2,754	8,262	11,016	16,524
5	0.1875	5,120	5,313	10,626	31,878	42,504	63,756
6	0.09375	20,480	20,865	41,730	125,190	166,920	250,380
7	0.046875	81,920	82,689	165,378	496,134	661,512	992,268

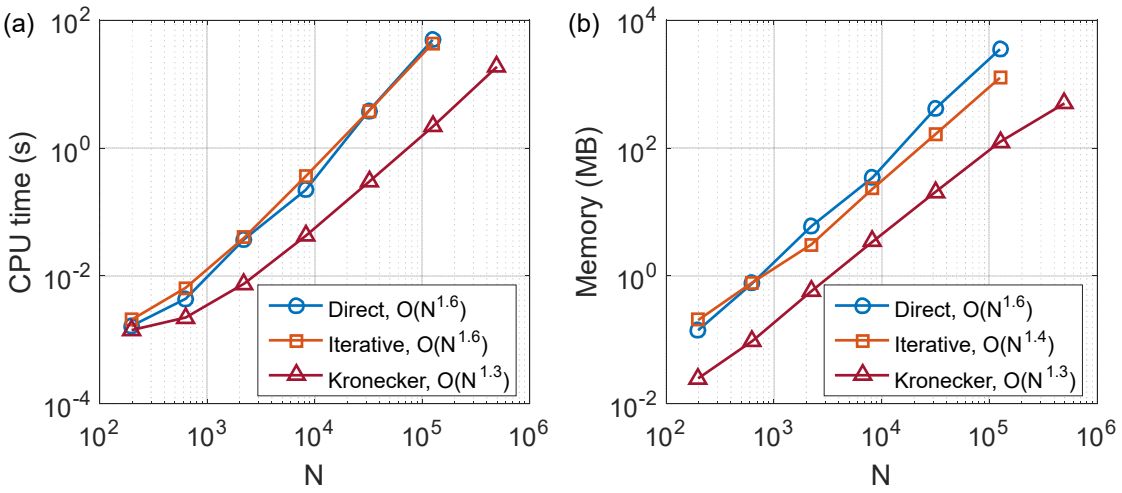


Figure 3.9. Comparison of computational performance between different solvers for *TDG-1* method: (a) CPU time and (b) memory usages

Figure 3.10 demonstrates the computational performances of different solvers for the two-field formulation STFEM. It also shows that the *Kronecker* solver works more efficiently when compared to the other solvers. The time complexity is further reduced from $\mathcal{O}(N^{1.7})$ to $\mathcal{O}(N^{1.2})$. Although storage complexities of the *Iterative* and *Kronecker* solvers are the same, the actual memory cost of the former is almost an order of magnitude higher than the latter, which indicates high memory usage efficiency. Therefore, it enables solution of much larger problems using the same hardware.

For the enriched formulation of XTFEM, the computational performances of different solvers are illustrated in Figure 3.11. The *Direct* solver shows time complexity of $\mathcal{O}(N^{1.6})$,

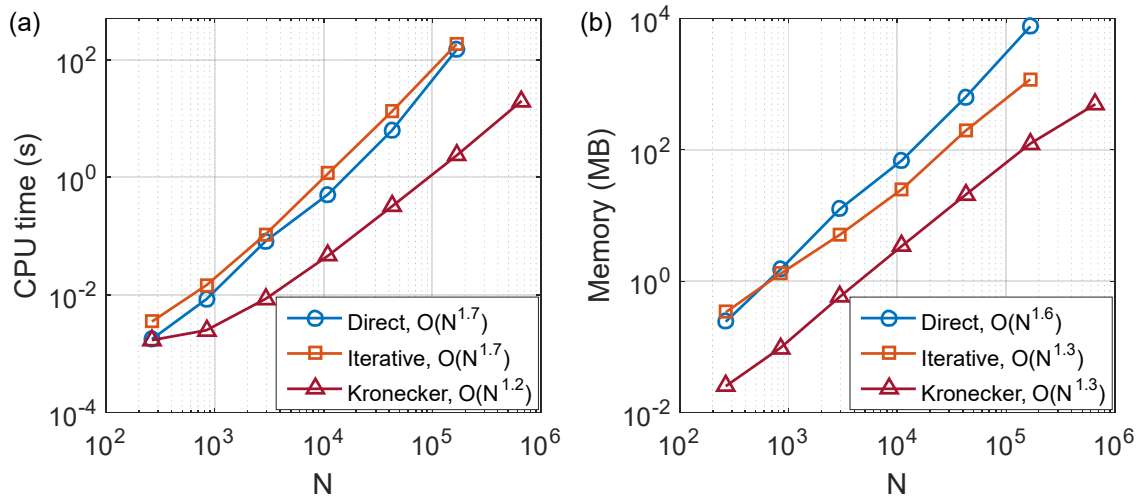


Figure 3.10. Comparison of computational performance between different solvers for *TDG-2* method: (a) CPU time and (b) memory usages

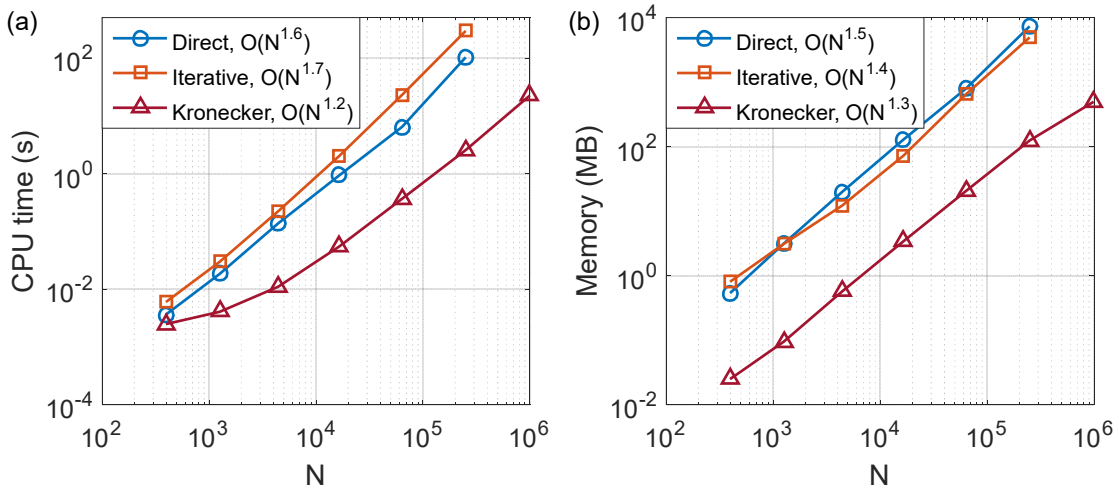


Figure 3.11. Comparison of computational performance between different solvers for *TDG-e* method: (a) CPU time and (b) memory usages

which is slightly better than $\mathcal{O}(N^{1.7})$ of the *Iterative* solver. The *Kronecker* solver further reduces the time complexity to $\mathcal{O}(N^{1.2})$. The memory cost of the Kronecker solver is close to two orders of magnitude lower than the others.

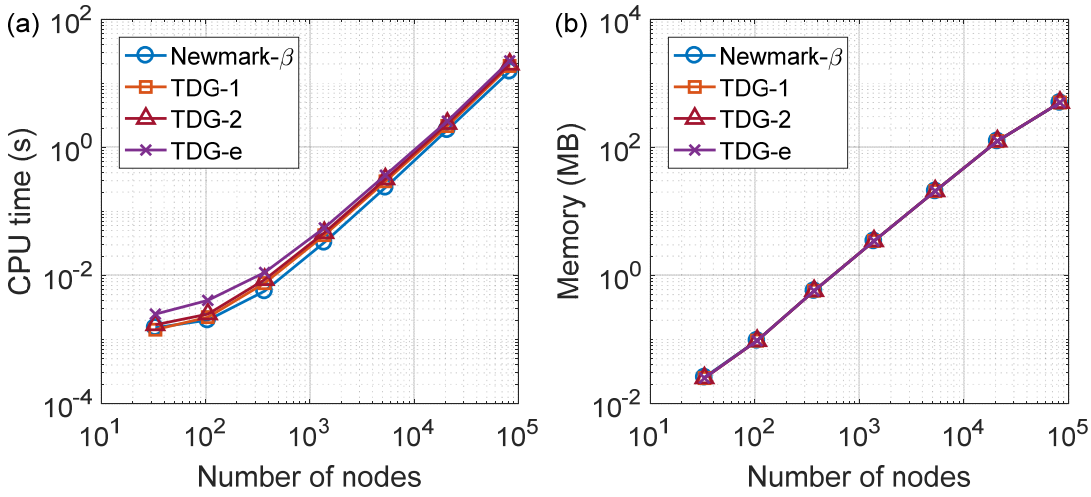


Figure 3.12. Comparison of computational performances between Newmark-β method and TDG-based methods: (a) CPU time and (b) memory usages

To further demonstrate efficiency of the *Kronecker* solver, we compare its computational cost with the Newmark-β method. Note that the *Iterative* solver is used for Newmark-β method. The results are shown in Figure 3.12. When accelerated by the *Kronecker* solver, all TDG formulations achieve the same computational complexity with Newmark-β method. The time costs of TDG methods are only slightly higher while the memory costs are almost the same.

3.7.2.2 3D spatial discretization

Next, we evaluate the computational performance of the proposed solution algorithm for 3D cases. A summary of the 3D meshes and corresponding numbers of DOFs are presented in Table 3.8.

Computational performances of different solvers for the single-field formulations are shown in Figure 3.13. The time efficiencies of the *Direct* solver and the *Iterative* solver are $\mathcal{O}(N^{2.5})$ and $\mathcal{O}(N^{1.9})$ respectively. Although the performance of the *Iterative* solver is better than the *Direct* solver, the numerical efficiencies of these two solvers both deterio-

Table 3.8. 3D meshes of the thin plate

No.	Element size (mm)	# Elements	# Nodes	# Spatial DOFs (Newmark-β)	# Space-time DOFs		
					TDG-1	TDG-2	TDG-e
1	2.0	45	128	384	1,152	1,536	2,304
2	1.5	80	210	630	1,890	2,520	3,780
3	1.0	360	651	1,953	5,859	7,812	11,718
4	0.75	960	1,476	4,428	13,284	17,712	26,568
5	0.5	2,880	3,965	11,895	35,685	47,580	71,370
6	0.4	5,625	7,296	21,888	65,664	87,552	131,328
7	0.3	14,000	16,968	50,904	152,712	203,616	305,424
8	0.25	23,040	27,225	81,675	245,025	326,700	490,050

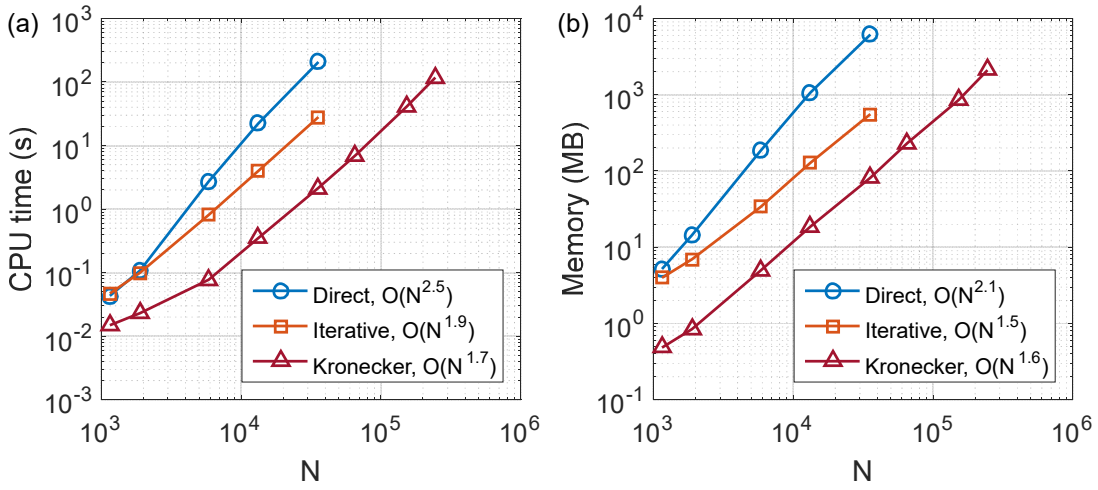


Figure 3.13. Comparison of computational performance between different solvers for *TDG-1* method: (a) CPU time and (b) memory usages

rated greatly when comparing with the 2D cases. However, the *Kronecker* solver remains a high efficiency with a time complexity of $\mathcal{O}(N^{1.7})$ and a storage complexity of $\mathcal{O}(N^{1.6})$. For $N > 10^4$, the performance of the *Kronecker* solver is an order of magnitude better than the *Iterative* solver and two orders of magnitude better than the *Direct* solver as shown in Figure 3.13.

For the two-field formulation in 3D cases, the *Kronecker* solver works significantly better than the other solvers as shown in Figure 3.14. It reduces the time complexity from

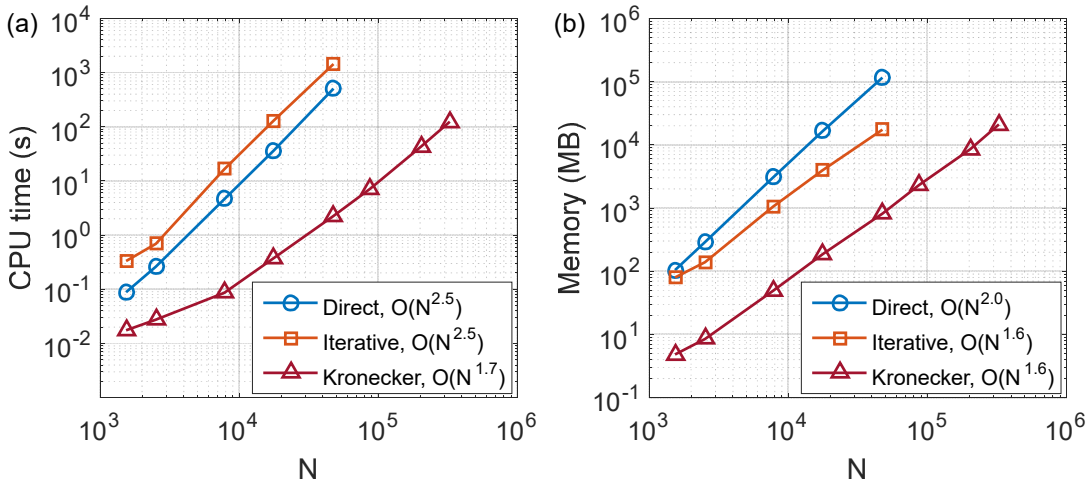


Figure 3.14. Comparison of computational performance between different solvers for *TDG-2* method: (a) CPU time and (b) memory usages

$\mathcal{O}(N^{2.5})$ to $\mathcal{O}(N^{1.7})$. The computational cost of the *Kronecker* solver is at least two orders of magnitude lower than the others for larger N.

Figure 3.15 demonstrates the computational performances of different solvers for the enriched formulation in 3D cases. The time complexity of the *Direct* solver is further increased to $\mathcal{O}(N^{2.7})$. The *Iterative* solver performs slightly better than the *Direct* solver. The *Kronecker* solver remains the same high efficiency of $\mathcal{O}(N^{1.7})$ and its cost is almost $1 \sim 2$ orders of magnitude lower than the other two for $N > 10^4$ cases.

The results presented so far showed that the *Kronecker* solver achieved significantly better performance for all TDG formulations in 3D cases. To further demonstrate its efficiency, we compare its performance with Newmark-β method. Figure 3.16 shows that the computational efficiency of all TDG formulations accelerated by the *Kronecker* solver are on the same level of Newmark-β method. While memory costs are very close, time costs of the TDG methods are constant times higher than Newmark-β method, which agrees well with the theoretical analysis in Section 3.5. As we already shown in Section 2.6 that the TDG methods generally employ a much larger time step than the semi-discrete schemes due

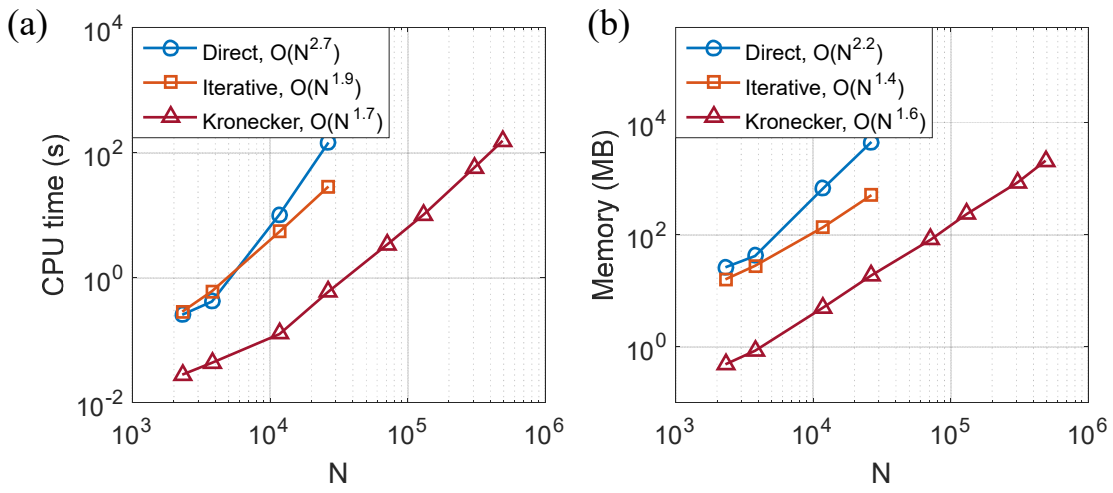


Figure 3.15. Comparison of computational performance between different solvers for *TDG-e* method: (a) CPU time and (b) memory usages

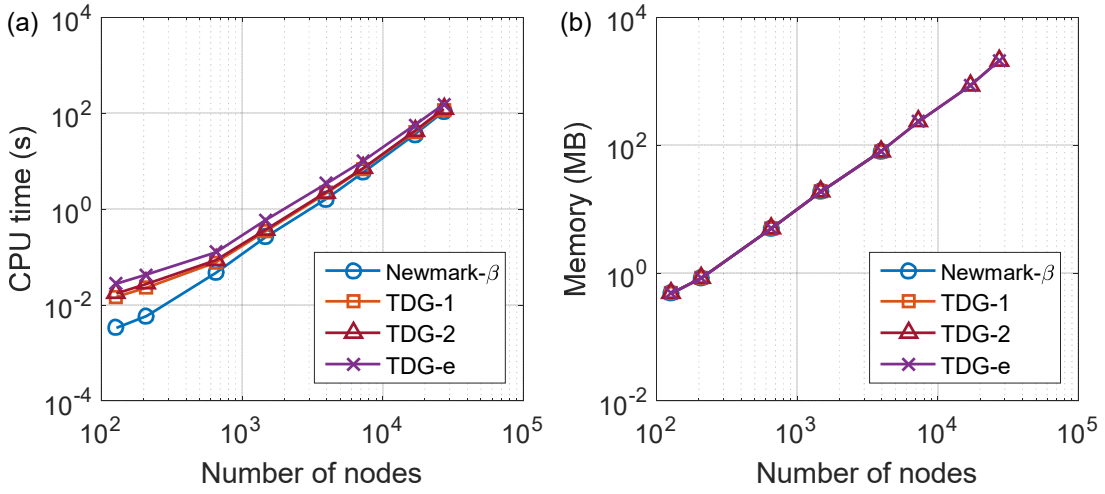


Figure 3.16. Comparison of computational performances between Newmark-β method and TDG-based methods: (a) CPU time and (b) memory usages

to the higher-order accuracy and unconditional stability. Thus, with the acceleration of the proposed *Kronecker* solver, the overall computational performance of TDG methods is much better than that of the traditional semi-discrete schemes such as the Newmark-β or central difference method.

With all the results shown above, it is concluded that the proposed *Kronecker* solver is highly efficient for numerous TDG formulations in both 2D and 3D cases.

3.7.3 Performance of the hybrid iterative/direct solver

To demonstrate the performance of the proposed *hybrid* iterative/direct solver, we study again the previous plate example with 3D discretization and evaluate the CPU time and memory usages for solving a single space-time slab. For simplicity, only XTFEM is considered here. The performance is assessed in two steps. First, we check the single-thread performance without using the parallel computing capability. Next, the computational complexity of the accelerated XTFEM is compared with standard FEM.

3.7.3.1 Serial version

To study the single-thread performance of the *hybrid* solver, we gradually refine the spatial discretization of the plate until it reaches about a half *million* DOFs. During the refinement process, the aspect ratio of the element remains the same to ensure consistent mesh quality. For comparison purposes, 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 is the conventional preconditioned iterative solver introduced in Section 3.4, which employs the ILU preconditioner with AMD reordering method and the GMRES iteration algorithm. A desktop workstation equipped with the Intel Xeon E5-2623v3 CPU and 32 GB RAM is used for the single-thread performance testing.

Figure 3.17 provides a comparison on the performances of those linear system solvers. According to Figure 3.17 (a), the *hybrid* solver achieves a time complexity of $\mathcal{O}(N^{1.4})$ while the time complexities of the *direct* and *iterative* solvers are respectively $\mathcal{O}(N^{2.6})$ and $\mathcal{O}(N^{1.8})$. For the cases of $N > 10^4$, the *hybrid* solver is at least $1 \sim 2$ orders of magnitude faster than

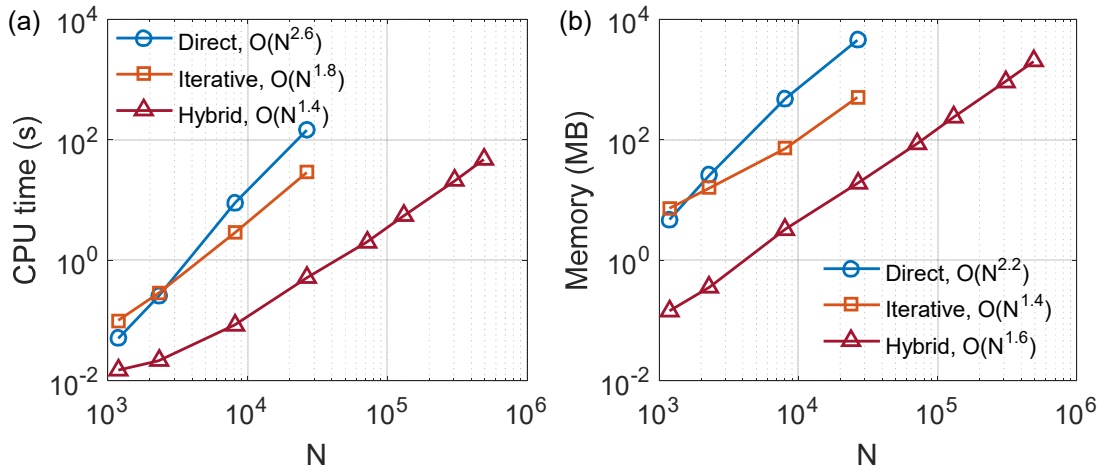


Figure 3.17. Comparison of computational performance among the *direct*, *iterative* and *hybrid* solvers, (a) CPU time usage and (b) memory cost

the others. The memory costs are shown in Figure 3.17 (b). In terms of storage complexity, the *iterative* solver achieves the best performance of $\mathcal{O}(N^{1.4})$. The *direct* solver shows the worst performance of $\mathcal{O}(N^{2.2})$. The *hybrid* solver demonstrates a complexity of $\mathcal{O}(N^{1.6})$, which is slightly higher than that of the *iterative* solver. However, in terms of the actual memory usage, the *hybrid* solver is $1 \sim 2$ 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 standard FEM that employs the implicit Newmark-β time integration method. 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.

Performance comparison between the XTFEM and Newmark-β method is shown in Figure 3.18. 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

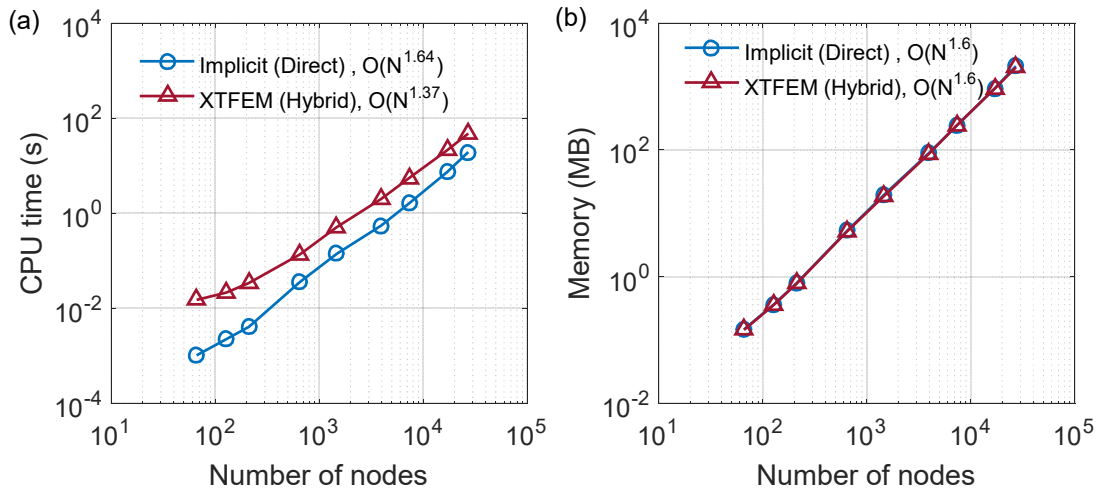


Figure 3.18. Comparison of performance between the implicit Newmark-β method and the XTFEM, (a) CPU time usage and (b) memory cost

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 $\mathcal{O}(N^{1.64})$ is higher than XTFEM, which is only $\mathcal{O}(N^{1.37})$. 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, which is already demonstrated in Section 2.6.

3.7.3.2 Parallel version

Similar to the single-thread test, a set of spatial meshes, which is summarized in Table 3.9, is created by refining the element size to test the parallel performance. The largest case leads to over 100 *million* DOFs. It is noted that all the parallel performance tests presented in this example are conducted on the Lonestar-5, a supercomputer from the Texas Advanced

Table 3.9. 3D plate meshes for the parallel solver testing

No.	Element size (mm)	# Elements	# Nodes	# DOFs
1	0.5	2,880	3,965	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

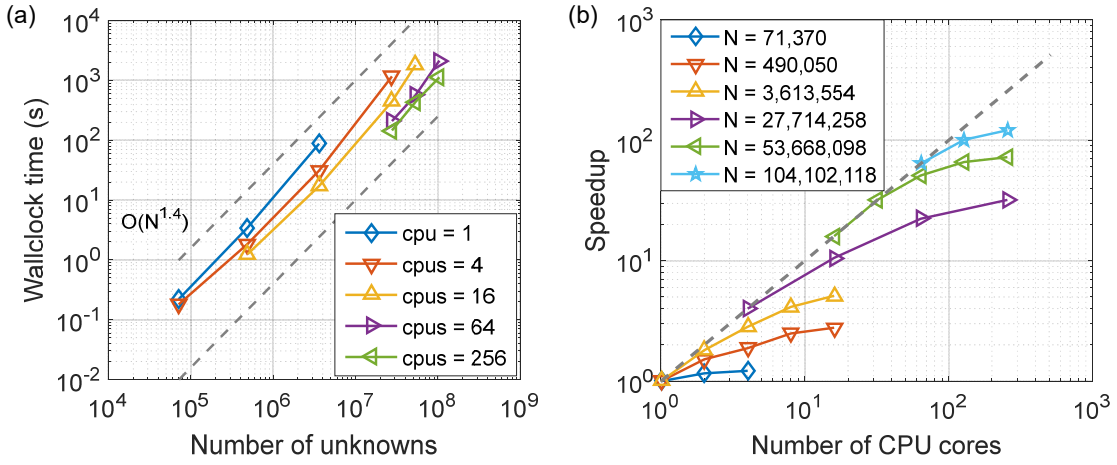


Figure 3.19. Performance of preconditioner evaluation by direct solver: (a) Wall-clock time usage vs. number of unknowns and (b) Speedup vs. number of CPU cores

Computing Center (TACC)[1]. Instead of using CPU time, the elapsed time or the wall-clock time usage is employed to measure the parallel performance of XTFEM.

As discussed in Section 3.6.1, 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

[1]See detailed hardware specifications at `https://portal.tacc.utexas.edu/user-guides/lonestar5`

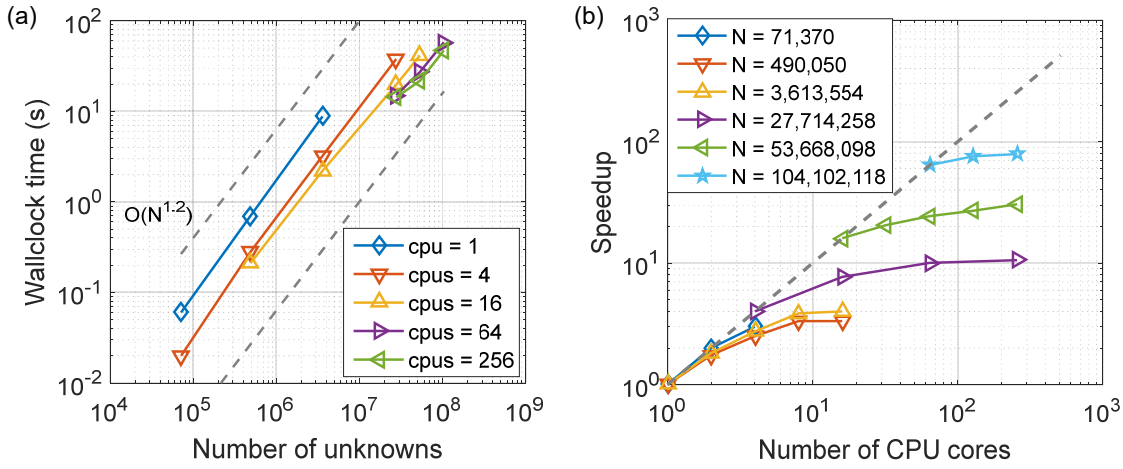


Figure 3.20. Performance of the GMRES solver: (a) Wall-clock time usage vs. number of unknowns and (b) Speedup vs. number of CPUs

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 Figure 3.19. It can be seen from Figure 3.19 (a) that the overall time complexity for different number of CPU cores is around $\mathcal{O}(N^{1.4})$. The speedup ratios are shown in Figure 3.19 (b) with a 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 gets large since the cost of communication becomes more significant and overwhelms that of computation.

Figure 3.20 shows the performance of the second part – the GMRES solver in terms of the computational cost of each time step. It can be seen from Figure 3.20 (a) that the time complexity of the GMRES solver is only $\mathcal{O}(N^{1.2})$. The corresponding speedup ratio is shown in Figure 3.20 (b), where a similar trend to that of the preconditioning part can be observed. However, the parallel efficiency of the GMRES part is lower than that of the preconditioning part.

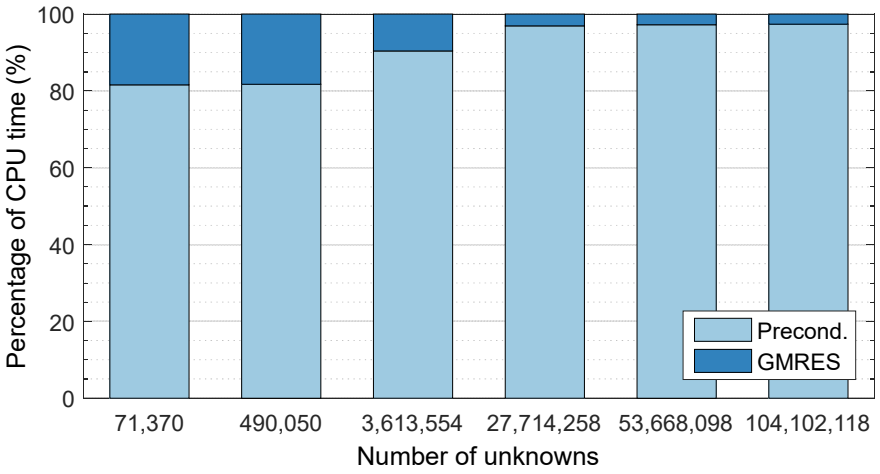


Figure 3.21. A breakdown of the time usage of the hybrid solver

A comparison of the time usages between those two parts for different number of unknowns is shown in Figure 3.21. 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 3.6.1, which minimizes the preconditioner evaluation.

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. The performance under such a testing configuration is summarized in Table 3.10. It shows that with the second-level parallelism additional speedup can be achieved for both parts of the hybrid solver. For this specific case, a total of $4 \sim 8$ OpenMP threads per MPI process yields the optimal efficiency.

Table 3.10. Hybrid MPI/OpenMP parallel performance of the hybrid solver

# DOFs	# MPI processes[a]	# OpenMP threads per process	Preconditioner (direct part)		GMRES (iterative part)	
			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

Note: [a] One MPI process per compute node

As a brief summary, we showed in this example that the serial version of the proposed hybrid solver is at least $1 \sim 2$ 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, the cost of solving XTFEM stiffness matrix equations is significantly reduced.

3.8 Summary

In this chapter, we studied several efficient solution algorithms to space-time FEM. We first introduced a conventional iterative solver features with sparse matrix storage/operation, ILU preconditioners, AMD/RCM reordering and GMRES iteration algorithms. Numerical examples show that its computational efficiency substantially outperforms the direct solver with a full matrix format and it effectively handles 2D problems with nearly half *million* unknowns on a single PC. However, further testing shows that efficiency of both the conventional iterative solver and the multi-frontal direct sparse solver significantly deteriorated in 3D cases.

By exploiting the unique block structure of space-time stiffness matrix, we proposed the Kronecker preconditioning technique to accelerate the conventional iterative solver. Extensive numerical examples demonstrate that significantly accelerates the solution to the

space-time linear systems of equations. Its numerical efficiency is at least $1 \sim 2$ orders of magnitude better than the direct sparse solver and the conventional iterative solver for relatively large numbers of DOFs (e.g., $N > 10^4$). It has been shown that the proposed Kronecker solver work well for single-field, two-field and enriched TDG formulations in both 2D and 3D cases. Unlike the iterative predictor/multi-corrector algorithms developed in (Li and Wiberg, 1996; Chien and Wu, 2000; Kunthong and Thompson, 2005), this algorithm does not require partially decoupling of the space-time stiffness matrix. Therefore, the proposed algorithm is generally applicable to all the TDG formulations introduced in Chapter 2.

Based on this novel Kronecker solver, we further proposed a hybrid iterative/direct solver by replacing the spatial component of Kronecker preconditioner based on incomplete factorization with the spatial stiffness matrix. The hybrid solver enables an HPC implementation that exploits both distributed- and shared-memory parallelisms in space-time FEM. Benchmark examples show that serial version of the hybrid solver is also at least $1 \sim 2$ orders of magnitude faster in computing time and cheaper in memory consumption than the conventional solvers. The parallel version efficiently handles XTFEM stiffness matrix equations with over 100 *million* unknowns using only 64 CPU cores.

With the advanced solution techniques presented so far, the computational cost of TDG methods for a single time-step is reduced to the same as the implicit algorithm implementation in the semi-discrete FEM, however, TDG method performs better since it is capable of using larger time step size than the implicit algorithm for accomplishing comparable accuracy. With such a substantial improvement in computational efficiency, space-time FEM is ideal and efficient for simulating 3D and large-scale problems that are featured by multi-temporal scales, sharp gradients and discontinuities in time. This implementation also paves the way for solving nonlinear problems as most of them can be treated by Newton's method that leads to linear system of equations, similar to the ones being treated here.

CHAPTER 4

GPU-ACCELERATED TWO-SCALE DAMAGE MODEL

4.1 Introduction

In Chapter 2 we have shown that, even with very large time steps, XTFEM remains highly accurate and stable in capturing high frequency oscillations in mechanical responses due to various fatigue loading conditions, which successfully addressed the multi-temporal scale challenge in direct numerical simulation of HCF problems. With the efficient acceleration techniques proposed in Chapter 3, the capability of XTFEM in handling practical HCF applications is substantially extended. However, the computational framework still lacks a nonlinear constitutive model to account for material degradation due to fatigue damage under cyclic loading.

Fatigue damage can be modeled using fracture mechanics, such as the famous Paris law (Paris and Erdogan, 1963). The porous metal plasticity approach, such as the Gurson, Tvergaard and Needleman (GTN) model (Gurson, 1977; Tvergaard and Needleman, 1984) can also be employed for modeling void growth under cyclic loading. Another broad class of fatigue damage models is based on the *Continuum Damage Mechanics* (Lemaitre, 1996; Lemaitre and Desmorat, 2005). Reviews on CDM-based fatigue damage models can be found in (Bhamare, 2012; Bhamare et al., 2014; Schlinkman, 2019).

In this chapter, we couple the XTFEM with a CDM-based two-scale fatigue damage model proposed by (Lemaitre and Doghri, 1994; Lemaitre et al., 1999) and (Desmorat et al.,

---

The following articles were reused in this chapter with permissions from the publishers:

1. Zhang, R., S. Naboulsi, T. Eason, and D. Qian (2019). A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing. *Finite Elements in Analysis and Design 166*, 103320. Reuse with permission from *Elsevier*.
2. Zhang, R., L.Wen, S. Naboulsi, T. Eason, V. K. Vasudevan, and D. Qian (2016). Accelerated multi-scale space-time finite element simulation and application to high cycle fatigue life prediction. *Computational Mechanics 58*(2), 329–349. Reuse with permission from *Springer Nature*.

2007) for HCF applications on metals. A unique feature of the two-scale 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 the number of loading cycles, which is critical for multiaxial and other complex loading conditions. Under complex loading conditions, the definition of cycle may become ambiguous and the cycle-dependent damage models require more or less arbitrary equivalent cycle count methods, such as rain-flow count. The two-scale damage model has already been successfully applied to HCF problems under uniaxial, biaxial, random and thermal cyclic loading histories (Lautrou et al., 2009; Zhang et al., 2016). However, it should be noted that many other microstructure-based material damage models (Liu et al., 2010; Tian et al., 2010; Anahid et al., 2011; Ghosh and Chakraborty, 2013) can also be integrated with XTFEM. For example, a CDM-based fatigue damage model was integrated with XTFEM for damage evolution in synthetic rubber under cyclic loading (Wada, 2017; Wada et al., 2018).

4.2 Two-scale high cycle fatigue damage model

In most HCF applications, magnitudes of fatigue loading are moderate such that the stress level is below the engineering yield stress. In other words, the mesoscale material behavior under HCF loading is essentially elastic. However, due to the existence of defects at microscale, such as voids/cracks, inclusions, etc., the mesoscale stress induces a micro-internal stress level greater than the local yield stress. The plastic strain accumulates under cyclic loading and leads to permanent deformation at microscale such as slip bands and decohesion, which further leads to microcracks and eventually mesoscale fracture.

To account for the micro-plasticity and micro-damage at the defects scale, Lemaitre and Doghri (Lemaitre and Doghri, 1994) and Lemaitre et al. (Lemaitre et al., 1999) developed a two-scale damage model that describes the micro-plasticity with classical plasticity equations and the micro-damage with an evolution law governed by the accumulated plastic strain

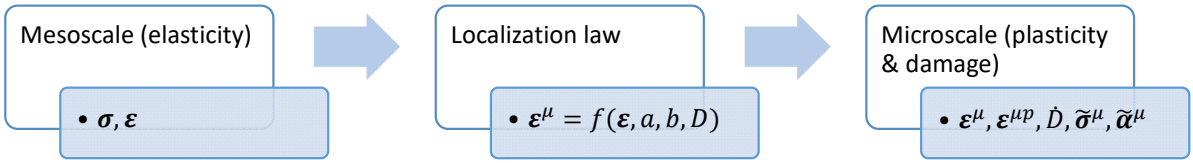


Figure 4.1. A sketch of the two-scale HCF damage model

rate. In this phenomenological model, the various microdefects are conceptually aggregated as a spherical weak inclusion embedded in the elastic material at mesoscale (Desmorat et al., 2007). The weakness of the inclusion is accounted for by a microscale yield stress that equals to the mesoscale endurance fatigue limit. To bridge these two length scales, the Eshelby-Kröner localization law (Eshelby, 1957; Kröner, 1961) is introduced. The above analysis provides a general picture of this two-scale damage model, which is further illustrated in Figure 4.1.

We briefly introduce formulations of the two-scale damage model proposed by Desmorat and coworkers (Desmorat et al., 2007) here. Note that the original model established in (Desmorat et al., 2007) accounts for mechanical loadings as well as temperature variations. It was integrated with XTFEM for thermo-mechanical HCF applications in a recent work (Schlinkman, 2019). However, in the current implementation, the thermally-induced stresses are not considered and the material behavior is assumed temperature-independent. Therefore, the related equations such as thermal expansion in the original formulation are excluded in this section.

4.2.1 Mesoscale modeling

As we already explained, the mesoscale material behavior under HCF loading conditions is elastic. Assuming an isotropic linear elastic material, we have the mesoscale constitutive law written as

$$\boldsymbol{\varepsilon} = \frac{1+\nu}{E}\boldsymbol{\sigma} - \frac{\nu}{E}\mathrm{tr}\boldsymbol{\sigma} \cdot \boldsymbol{I} \tag{4.1}$$

where E and ν are the Young's modulus and the Poisson's ratio, respectively. $\boldsymbol{I}$ is the second-order identity tensor.

4.2.2 Microscale modeling

A coupled elasto-plasticity/damage model is established to describe nonlinear microscale material behavior that accounts for micro-plasticity and micro-damage induced by mesoscale stresses. The total strain at the microscale is

$$\boldsymbol{\varepsilon}^{\mu} = \boldsymbol{\varepsilon}^{\mu e} + \boldsymbol{\varepsilon}^{\mu p} \tag{4.2}$$

where the superscript μ denotes a microscale variable, $\boldsymbol{\varepsilon}^{\mu e}$ and $\boldsymbol{\varepsilon}^{\mu p}$ are elastic and plastic strains, respectively.

The microscale elastic strain is governed by

$$\boldsymbol{\varepsilon}^{\mu e} = \frac{1+\nu}{E}\frac{\boldsymbol{\sigma}^{\mu}}{1-D} - \frac{\nu}{E}\frac{\mathrm{tr}\boldsymbol{\sigma}^{\mu}}{1-D}\cdot\boldsymbol{I} \tag{4.3}$$

where D is the scalar damage variable ranges between 0 (no damage) and 1 (completely damaged). A conceptual physical definition of damage is the ratio between damaged (unable to resist loading) and total cross-sectional areas of the Representative Volume Element (RVE) under consideration. The damage variable is a tensor for general cases. However, for isotropic damage, it is a scalar.

According to Lemaitre's damage law, we introduce the concept of effective stress, which is the stress acting on the undamaged resisting area and given by the force equilibrium

$$\tilde{\boldsymbol{\sigma}}^{\mu} = \frac{\boldsymbol{\sigma}^{\mu}}{1-D} \tag{4.4}$$

Substituting Eq. (4.4) into Eq.(4.3) yields a simplified microscale elasticity law that has the same form as in Eq. (4.1).

For isotropic material behavior, we employ the classical von Mises yield criterion given by

$$f = \left(\tilde{\boldsymbol{\sigma}}^{\mu} - \boldsymbol{X}^{\mu}\right)_{\text{eq}} - \sigma_f^{\infty} = 0 \tag{4.5}$$

where $\boldsymbol{X}^{\mu}$ is the backstress, σ_f^{∞} is the endurance (asymptotic) fatigue limit, which is a material property, and $(\cdot)_{\text{eq}}$ denotes the von Mises norm, which is given by

$$(\cdot)_{\text{eq}} = \sqrt{\frac{3}{2}\left(\cdot^{\text{dev}}\right) : \left(\cdot^{\text{dev}}\right)} \tag{4.6}$$

where superscript dev indicates the deviatoric part of a tensor. For example, the stress tensor can be decomposed by

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\text{dev}} + \boldsymbol{\sigma}^{\text{hyd}} \tag{4.7}$$

in which the hydrostatic part is

$$\boldsymbol{\sigma}^{\text{hyd}} = \sigma^{\text{hyd}} \cdot \boldsymbol{I} = \frac{1}{3}\text{tr}\boldsymbol{\sigma} \cdot \boldsymbol{I} \tag{4.8}$$

and the deviatoric part is

$$\boldsymbol{\sigma}^{\text{dev}} = \boldsymbol{\sigma} - \frac{1}{3}\text{tr}\boldsymbol{\sigma} \cdot \boldsymbol{I} \tag{4.9}$$

The microscale plastic strain rate is governed by the standard associated flow rule

$$\dot{\boldsymbol{\varepsilon}}^{\mu p} = \frac{3}{2}\frac{\tilde{\boldsymbol{\sigma}}^{\mu\text{dev}} - \boldsymbol{X}^{\mu}}{\left(\tilde{\boldsymbol{\sigma}}^{\mu} - \boldsymbol{X}^{\mu}\right)_{\text{eq}}}\dot{p}^{\mu} \tag{4.10}$$

in which $\dot{p}^{\mu}$ is the accumulated plastic strain rate, which is defined by

$$\dot{p}^{\mu} = \sqrt{\frac{3}{2}\dot{\boldsymbol{\varepsilon}}^{\mu p} : \dot{\boldsymbol{\varepsilon}}^{\mu p}} \tag{4.11}$$

Evolution of the backstress is described by the linear kinematic hardening rule

$$\dot{\boldsymbol{X}}^{\mu} = \frac{2}{3}C_y\left(1 - D\right)\dot{\boldsymbol{\varepsilon}}^{\mu p} \tag{4.12}$$

where C_y is the plastic modulus, which is also a material parameter.

The Lamaitre damage evolution law is given by

$$\dot{D} = \left(\frac{Y^{\mu}}{S}\right)^{s} \dot{p}^{\mu} \qquad \text{if} \quad w_s > w_D \tag{4.13}$$

which correlates the damage rate with the accumulated plastic strain rate. In Eq. (4.13), material parameters S and s are damage strength and damage exponent, respectively. Y^{μ} is the effective elastic energy density or the strain energy release rate. The criterion to update the damage rate is by comparing the stored energy density w_s with the energetic damage threshold w_D, which is another material parameter. The definitions of the above variables are given as follows.

The strain energy release rate is given by

$$\begin{aligned} Y^{\mu} =& \frac{1+\nu}{2E}\left[\langle\tilde{\boldsymbol{\sigma}}^{\mu}\rangle^{+} : \langle\tilde{\boldsymbol{\sigma}}^{\mu}\rangle^{+} + h\left(\frac{1-D}{1-hD}\right)^{2}\langle\tilde{\boldsymbol{\sigma}}^{\mu}\rangle^{-} : \langle\tilde{\boldsymbol{\sigma}}^{\mu}\rangle^{-}\right] \\ & - \frac{\nu}{2E}\left[\langle \mathrm{tr}\tilde{\boldsymbol{\sigma}}^{\mu}\rangle^{2} + h\left(\frac{1-D}{1-hD}\right)^{2}\langle -\mathrm{tr}\tilde{\boldsymbol{\sigma}}^{\mu}\rangle^{2}\right] \end{aligned} \tag{4.14}$$

where h is the micro-defects closure parameter that accounts for the smaller damage evolution in compression than in tension and typically $h \approx 0.2$ for metals (Lemaitre, 1996; Lemaitre and Desmorat, 2005), $\langle\tilde{\boldsymbol{\sigma}}^{\mu}\rangle^{+}$ and $\langle\tilde{\boldsymbol{\sigma}}^{\mu}\rangle^{-}$ are the positive and negative parts of the effective stress tensor in terms of principal values, respectively. Note that $\langle\cdot\rangle$ denotes the Macaulay bracket.

In the plasticity framework, the stored energy density is contributed by both isotropic and kinematic hardening. However, the kinematic hardening contribution is periodic under cyclic loading while the isotropic hardening contribution is monotonic. Since we are dealing with HCF applications, the periodic contribution is lost and the stored energy density is given by

$$w_s = \int_t \left(\sigma_{eq}^{\mu} - \sigma_y^{\infty}\right)\dot{p}^{\mu} dt \tag{4.15}$$

where the yield stress $\sigma_y^{\infty} \approx \sigma_f^{\infty}$.

The energetic damage threshold can be approximated from the monotonic loading test

$$w_D \approx \left(\sigma_u - \sigma_f^{\infty}\right) \varepsilon_{pD} \tag{4.16}$$

in which σ_u and ε_{pD} are respectively the nominal ultimate tensile strength and the corresponding plastic strain, respectively.

Finally, crack initiates when the damage variable reaches a critical value D_c.

4.2.3 Scale bridging

Localization from mesoscale to microscale in the two-scale damage model is governed by the modified Eshelby-Kröner law (Eshelby, 1957; Kröner, 1961).

First, we have the deviatoric and hydrostatic parts of the microscale strain given by

$$\boldsymbol{\varepsilon}^{\mu\mathrm{dev}} = \frac{1}{1-bD}\left[\boldsymbol{\varepsilon}^{\mathrm{dev}} + b\left(1-D\right)\boldsymbol{\varepsilon}^{\mu p}\right] \tag{4.17}$$

$$\varepsilon^{\mu\mathrm{hyd}} = \frac{\varepsilon^{\mathrm{hyd}}}{1-aD} \tag{4.18}$$

By combining Eqs. (4.17) and (4.18) we have the localization rule written as

$$\boldsymbol{\varepsilon}^{\mu} = \boldsymbol{\varepsilon}^{\mu\mathrm{dev}} + \varepsilon^{\mu\mathrm{hyd}} \cdot \boldsymbol{I} = \frac{1}{1-bD}\left[\boldsymbol{\varepsilon} + \frac{\left(a-b\right)D}{3\left(1-aD\right)}\mathrm{tr}\boldsymbol{\varepsilon} \cdot \boldsymbol{I} + b\left(1-D\right)\boldsymbol{\varepsilon}^{\mu p}\right] \tag{4.19}$$

Since the micro-defects are conceptually collected as a spherical inclusion in this model, the corresponding Eshelby parameters are given by

$$a = \frac{1}{3} \cdot \frac{1+\nu}{1-\nu} \tag{4.20}$$

$$b = \frac{2}{15} \cdot \frac{4-5\nu}{1-\nu} \tag{4.21}$$

in which ν is the Poisson's ratio.

Closed-form solution to the above set of ordinary differential equations under simple loading conditions is given by (Desmorat et al., 2007) and summarized in Appendix A.

4.3 Numerical implementation

For general loading conditions, given the mesoscale stress/strain history as an input, the two-scale damage model needs to be solved numerically. Desmorat et al. (Desmorat et al., 2007) proposed an efficient implicit *Backward Euler* solution algorithm to avoid the cumbersome *Newton-Raphson* iterations. The *Backward Euler* scheme solves the two-scale damage model in three steps shown in this section.

4.3.1 Step 1: elastic prediction

Assuming elastic strain increment at microscale, the total microscopic strain at time t_{n+1} is given by

$$\boldsymbol{\varepsilon}^{\mu}_{n+1} = \frac{1}{1-bD_n}\left[\boldsymbol{\varepsilon}_{n+1} + \frac{(a-b)\,D_n}{3\,(1-aD_n)}\mathrm{tr}\boldsymbol{\varepsilon}_{n+1}\cdot\boldsymbol{I} + b\,(1-D_n)\,\boldsymbol{\varepsilon}^{\mu p}_{n}\right] \tag{4.22}$$

where subscript n denotes the time increment.

The stress at microscale is

$$\tilde{\boldsymbol{\sigma}}^{\mu}_{n+1} = \boldsymbol{C} : \boldsymbol{\varepsilon}^{\mu e}_{n+1} = \boldsymbol{C} : \left(\boldsymbol{\varepsilon}^{\mu}_{n+1} - \boldsymbol{\varepsilon}^{\mu p}_{n}\right) \tag{4.23}$$

in which $\boldsymbol{C}$ is the fourth-order elasticity tensor.

Yield function is evaluated as

$$f_{n+1} = \left(\tilde{\boldsymbol{\sigma}}^{\mu}_{n+1} - \boldsymbol{X}^{\mu}_{n}\right)_{eq} - \sigma^{\infty}_{f} \tag{4.24}$$

4.3.2 Step 2: plastic correction

For HCF applications, the number of cycles to crack initiation is very large ($10^5 \sim 10^7$). Hence, it is reasonable to assume a constant damage over a time increment. If the yield function $f_{n+1} > 0$, then plasticity corrections are made by solving the following set of coupled plasticity-damage equations with the *Backward Euler* scheme

$$\boldsymbol{\varepsilon}^{\mu}_{n+1} = \boldsymbol{\varepsilon}^{\mu e}_{n+1} + \boldsymbol{\varepsilon}^{\mu p}_{n+1} \tag{4.25}$$

$$\varepsilon^{\mu e}_{n+1} = \frac{1+\nu}{E}\tilde{\boldsymbol{\sigma}}^{\mu}_{n+1} - \frac{\nu}{E}\mathrm{tr}\tilde{\boldsymbol{\sigma}}^{\mu}_{n+1}\cdot\boldsymbol{I} \tag{4.26}$$

$$\Delta\boldsymbol{\varepsilon}^{\mu p} = \frac{3}{2}\frac{\tilde{\boldsymbol{\sigma}}^{\mu\mathrm{dev}}_{n+1} - \boldsymbol{X}^{\mu}_{n+1}}{\left(\tilde{\boldsymbol{\sigma}}^{\mu}_{n+1} - \boldsymbol{X}^{\mu}_{n+1}\right)_{\mathrm{eq}}}\Delta p^{\mu} \tag{4.27}$$

$$\boldsymbol{X}^{\mu}_{n+1} - \boldsymbol{X}^{\mu}_{n} = \frac{2}{3}C_y\left(1-D_n\right)\Delta\boldsymbol{\varepsilon}^{\mu p} \tag{4.28}$$

First, the incremental form of the localization law is given by

$$\Delta\boldsymbol{\varepsilon}^{\mu} = \frac{1}{1-bD_n}\left[\Delta\boldsymbol{\varepsilon} + \frac{(a-b)\,D_n}{3\left(1-aD_n\right)}\mathrm{tr}\left(\Delta\boldsymbol{\varepsilon}\right)\cdot\boldsymbol{I} + b\left(1-D_n\right)\Delta\boldsymbol{\varepsilon}^{\mu p}\right] \tag{4.29}$$

which can be rewritten by using Eq. (4.25) and given by

$$\Delta\boldsymbol{\varepsilon}^{\mu e} + \frac{1-b}{1-bD_n}\Delta\boldsymbol{\varepsilon}^{\mu p} - \frac{1}{1-bD_n}\Delta\boldsymbol{\varepsilon} - \frac{(a-b)\,D_n}{3\left(1-aD_n\right)\left(1-bD_n\right)}\mathrm{tr}\left(\Delta\boldsymbol{\varepsilon}\right)\cdot\boldsymbol{I} = 0 \tag{4.30}$$

Substituting the linear elasticity constitutive law, we obtain

$$\Delta\tilde{\boldsymbol{\sigma}}^{\mu} + \frac{1-b}{1-bD_n}2G\Delta\boldsymbol{\varepsilon}^{\mu p} - \frac{1}{1-bD_n}\boldsymbol{C}:\Delta\boldsymbol{\varepsilon} - \frac{K\,(a-b)\,D_n}{\left(1-aD_n\right)\left(1-bD_n\right)}\mathrm{tr}\left(\Delta\boldsymbol{\varepsilon}\right)\cdot\boldsymbol{I} = 0 \tag{4.31}$$

where K is the bulk modulus.

Now we introduce a new variable defined by

$$\boldsymbol{S}^{\mu}_{n+1} = \tilde{\boldsymbol{\sigma}}^{\mu}_{n+1} - \boldsymbol{X}^{\mu}_{n+1} \tag{4.32}$$

and substitute it together with Eqs. (4.27) and (4.28) back into Eq. (4.32),

$$\begin{aligned}\boldsymbol{S}^{\mu}_{n+1} - \tilde{\boldsymbol{\sigma}}^{\mu}_{n} + \boldsymbol{X}^{\mu}_{n} + \left[C_y\left(1-D_n\right) + 3G\frac{1-b}{1-bD_n}\right]\frac{\boldsymbol{S}^{\mu\mathrm{dev}}_{n+1}}{\left(\boldsymbol{S}^{\mu}_{n+1}\right)_{\mathrm{eq}}}\Delta p^{\mu} \\ - \frac{1}{1-bD_n}\boldsymbol{C}:\Delta\boldsymbol{\varepsilon} - \frac{K\,(a-b)\,D_n}{\left(1-aD_n\right)\left(1-bD_n\right)}\mathrm{tr}\left(\Delta\boldsymbol{\varepsilon}\right)\cdot\boldsymbol{I} = 0\end{aligned} \tag{4.33}$$

which can be further simplified as

$$\boldsymbol{S}^{\mu}_{n+1} + \Gamma'\frac{\boldsymbol{S}^{\mu\mathrm{dev}}_{n+1}}{\left(\boldsymbol{S}^{\mu}_{n+1}\right)_{\mathrm{eq}}}\Delta p^{\mu} + \boldsymbol{Q}_s = 0 \tag{4.34}$$

where

$$\Gamma' = C_y\left(1-D_n\right) + 3G\frac{1-b}{1-bD_n} \tag{4.35}$$

and

$$\boldsymbol{Q}_s = \boldsymbol{X}_n^\mu - \tilde{\boldsymbol{\sigma}}_n^\mu - \frac{1}{1-bD_n}\boldsymbol{C} : \Delta\boldsymbol{\varepsilon} - \frac{K\,(a-b)\,D_n}{(1-aD_n)\,(1-bD_n)}\mathrm{tr}\,(\Delta\boldsymbol{\varepsilon})\cdot\boldsymbol{I} \tag{4.36}$$

are known from the states in the previous time increment.

Substituting Eq. (4.32) into the yield function Eq. (4.24) yields

$$\left(\boldsymbol{S}_{n+1}^\mu\right)_{eq} - \sigma_f^\infty = 0 \tag{4.37}$$

Note that $\boldsymbol{S}_{n+1}^\mu$ and Δp^μ are the only unknowns at time increment $n+1$ in Eqs. (4.34) and (4.37). Therefore, the exact solution for the plastic corrections is obtained as

$$\boldsymbol{S}_{n+1}^\mu = \frac{-\boldsymbol{Q}_s^{\mathrm{dev}}}{\left(1+\Gamma'\Delta p^\mu/\sigma_f^\infty\right)} - Q_s^H\cdot\boldsymbol{I} \tag{4.38}$$

where

$$\Delta p^\mu = \frac{\left(\boldsymbol{Q}_s\right)_{\mathrm{eq}} - \sigma_f^\infty}{\Gamma'} \tag{4.39}$$

4.3.3 Step 3: states update and damage evolution

With the solutions from the previous step, now we update the internal state variables and evaluate the damage.

The normal to the yield surface is

$$\boldsymbol{m}^\mu = \frac{\boldsymbol{S}_{n+1}^{\mu D}}{\sigma_f^\infty} \tag{4.40}$$

and the internal states at t_{n+1} are

$$\boldsymbol{\varepsilon}_{n+1}^{\mu p} = \boldsymbol{\varepsilon}_n^{\mu p} + \frac{3}{2}\boldsymbol{m}^\mu\Delta p^\mu \tag{4.41}$$

$$\boldsymbol{X}_{n+1}^\mu = \boldsymbol{X}_n^\mu + C_y\,(1-D_n)\,\boldsymbol{m}^\mu\Delta p^\mu \tag{4.42}$$

$$\tilde{\boldsymbol{\sigma}}_{n+1}^\mu = \boldsymbol{S}_{n+1}^\mu + \boldsymbol{X}_{n+1}^\mu \tag{4.43}$$

$$\boldsymbol{\varepsilon}_{n+1}^{\mu e} = \boldsymbol{C}^{-1} : \tilde{\boldsymbol{\sigma}}_{n+1}^\mu \tag{4.44}$$

When the energetic damage threshold satisfied, i.e. $w_s > w_D$ in which w_s can be integrated by the trapezoidal rule, the damage at t_{n+1} is updated by

$$D_{n+1} = D_n + \left(Y^{\mu}_{n+1}/S\right)^s \Delta p^{\mu} \tag{4.45}$$

where

$$\begin{aligned} Y^{\mu}_{n+1} =& \frac{1+\nu}{2E}\left[\left\langle \tilde{\boldsymbol{\sigma}}^{\mu}_{n+1}\right\rangle^{+} : \left\langle \tilde{\boldsymbol{\sigma}}^{\mu}_{n+1}\right\rangle^{+} + h\left(\frac{1-D_n}{1-hD_n}\right)^2 \left\langle \tilde{\boldsymbol{\sigma}}^{\mu}_{n+1}\right\rangle^{-} : \left\langle \tilde{\boldsymbol{\sigma}}^{\mu}_{n+1}\right\rangle^{-}\right] \\ & - \frac{\nu}{2E}\left[\left\langle \tilde{\boldsymbol{\sigma}}^{\mu}_{n+1} : \boldsymbol{I}\right\rangle^2 + h\left(\frac{1-D_n}{1-hD_n}\right)^2 \left\langle -\tilde{\boldsymbol{\sigma}}^{\mu}_{n+1} : \boldsymbol{I}\right\rangle^2\right] \end{aligned} \tag{4.46}$$

Finally, the stress tensor at t_{n+1} is updated by

$$\boldsymbol{\sigma}^{\mu}_{n+1} = (1 - D_{n+1})\, \tilde{\boldsymbol{\sigma}}^{\mu}_{n+1} \tag{4.47}$$

Material parameters involved with the two-scale damage model are calibrated with experiment data using the closed-form solution (Appendix A) and the above numerical procedure. See Appendix B for details.

4.4 Coupling with XTFEM

The numerical solution algorithm to the two-scale damage model was implemented first as a standalone program for the purpose of material parameter calibration as demonstrated in Appendix B. It was implemented as a post-processor for damage and fatigue for FEM packages (Desmorat et al., 2007). Similarly, it was also integrated with finite element code ABAQUS as a user's material subroutine (UMAT) to be evaluated on Gauss quadrature points (Schlinkman, 2019). In the current work, we integrate the two-scale damage model with the XTFEM as a nonlinear material constitutive solver. The element deletion/erosion technique is employed to remove failed elements with which the associated damage value reaches the critical damage D_c at crack initiation.

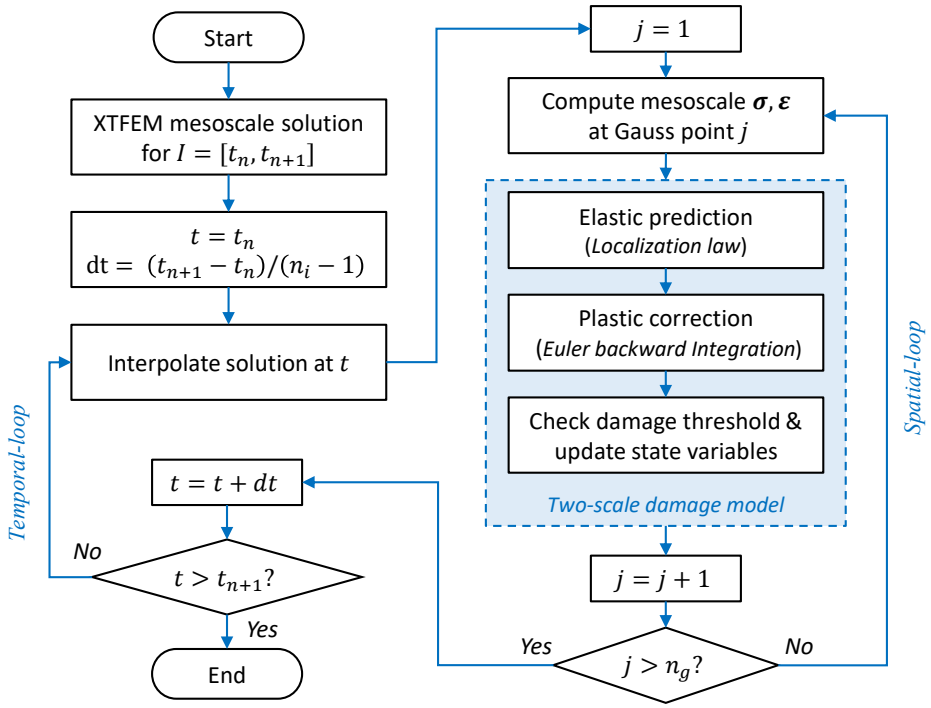


Figure 4.2. Implementation of the two-scale damage model and its coupling with XTFEM

Figure 4.2 shows the flowchart for the implementation of the two-scale damage model and its coupling with XTFEM. As it can be seen, we first interpolate XTFEM mesoscale solutions at the refined time increments by using the enriched space-time shape function. Then the mesoscale stress/strain histories are evaluated for each spatial Gauss point, which are provided as inputs for the two-scale damage model. Once the damage value at a Gauss point reaches the critical value D_c, the corresponding element will be deleted by removing its contribution to the stiffness matrix and fatigue crack initiates or propagates. In Figure 4.2, n_i is the number of temporal interpolation points and n_g is the number of spatial Gauss quadrature points. Therefore, the evaluation of two-scale damage model is embedded in a nested space-time loop and solved $n_i n_g$ times.

Additional temporal interpolation points per load cycle are required to ensure the accuracy of the single-step implicit *Backward Euler* scheme of the two-scale damage model.

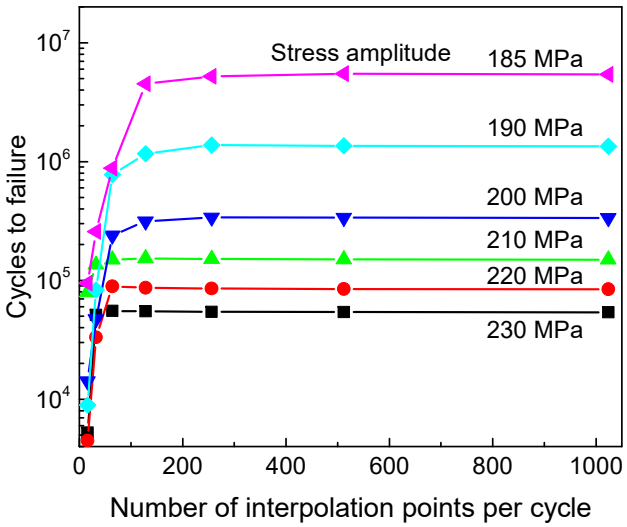


Figure 4.3. Convergence study of two-scale damage model under various stress amplitudes: cycles to failure vs. number of interpolation points per cycle

Thus, n_{ipc}, the number of temporal interpolation points per cycle, is first determined by a convergence study. This is implemented in the standalone code that solves the two-scale damage constitutive model using the same single-step implicit scheme. In the standalone code, the stress state is uniaxial and fully-reversed sinusoidal loading with constant amplitude is prescribed. The material parameters employed for this study are listed in Section 4.7. For various stress amplitudes, Figure 4.3 shows that the number of cycles to failure converged after $n_{\text{ipc}} > 200$. Based on this study, we choose the n_{ipc} to be 256. The number of interpolation points per XTFEM step n_i equals 25,600 if we employ a space-time slab size of 100 loading cycles.

In terms of computational cost, solving the nonlinear constitutive model is a major component in standard FEM implementation due to the complex constitutive update algorithms and repeated evaluations on all the material points. In the current computational framework, this implementation is more expensive 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, as shown in Figure 4.3, a good temporal resolution typically requires more than 200 increments per loading cycle to ensure convergence of solution to the two-scale damage model. To satisfy those requirements, the two-scale damage model is embedded in the nested temporal and spatial loops as illustrated in Figure 4.2 with a computational cost of $\mathcal{O}(n_i n_g)$, which could be very large for practical HCF problems. Consequently, solving the two-scale damage model adds to significant computational cost and becomes a bottleneck, which motivates the development of high-performance parallel computing algorithm.

4.5 GPGPU parallel computing acceleration

As illustrated in Figure 4.2, implementation of the two-scale damage model is involved in nested loops. Although the loop over temporal interpolation points is sequential by nature, the loop over the spatial Gauss points is not since the constitutive updates at different spatial quadrature points are independent of each other. As such, calculations on the spatial quadrature points can be carried out simultaneously and the computing task is well-suited for the *single instruction, multiple data* (SIMD) parallelization-based platform. With this intrinsic instruction-level parallelism, parallel computing techniques can be employed to accelerate the two-scale damage model. The many-core structured *Graphic Processing Unit* (GPU) is an ideal candidate for data-parallel computation tasks such as the constitutive updates outlined above.

4.5.1 Hardware and software architectures

GPUs are specialized processors that were originally designed for high-volume graphics data processing. Although the operating frequencies of the GPUs are generally lower than the CPUs, they devote many times more transistors to arithmetic operations (see Figure 4.4). A single GPU nowadays has up to several thousands of computing cores and this number is

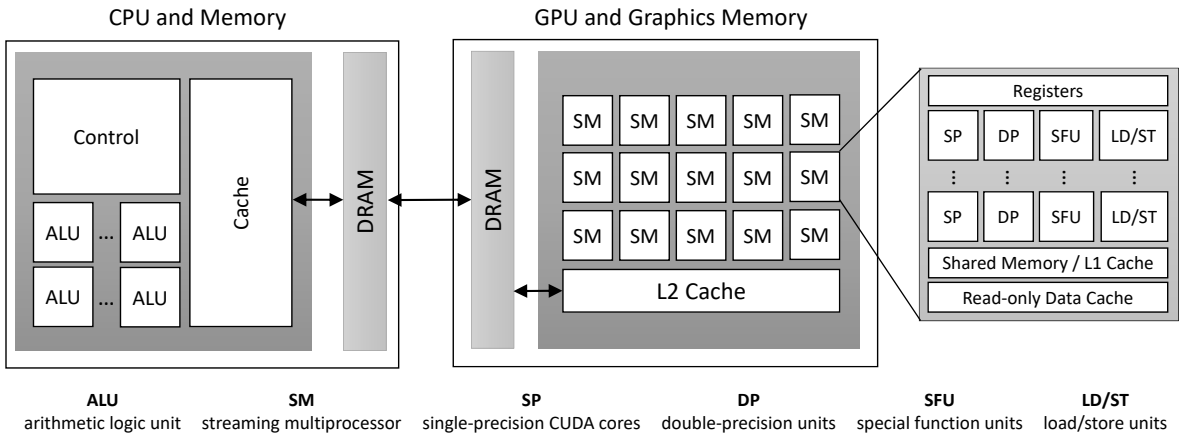


Figure 4.4. Hardware architectures of the CPU and GPU

still increasing rapidly. Based on this key feature, the GPUs are more suitable for computing tasks that are arithmetic-intensive and highly parallel. For this work, we employed the GPUs (TESLA K20c and later K-series models) by NVIDIA for parallel computing purpose. These GPUs are based on the Kepler architecture, which is schematically illustrated in Figure 4.4.

A full Kepler GPU is implemented with 15 streaming multiprocessors, which are commonly termed as SM or SMX. Furthermore, each of these SMs contains 192 single-precision cores (SP), 64 double-precision units (DP), 32 special function units (SFU) and 32 load/store units (LD/ST). These lead to a total of 2880 single-precision cores and 960 double-precision cores in a full Kepler GPU. In terms of memory hierarchy, the off-chip DRAM (Dynamic Random-Access Memory) is the largest memory for GPU (up to several Gigabytes), which also has a high theoretical bandwidth up to several hundred Gigabytes per second (see Table 3.4). All the parallel threads can access the data stored in DRAM through the on-chip level-2 (L2) high-speed cache. Within each SMs, there are on-chip shared memory/L1 cache and read-only data cache that can be accessed by threads running on the same SM, and registers are dedicated to each thread.

The idea of general-purpose computing on GPUs (GPGPU) has been introduced with a goal to streamline the programming on GPUs for high performance scientific computing

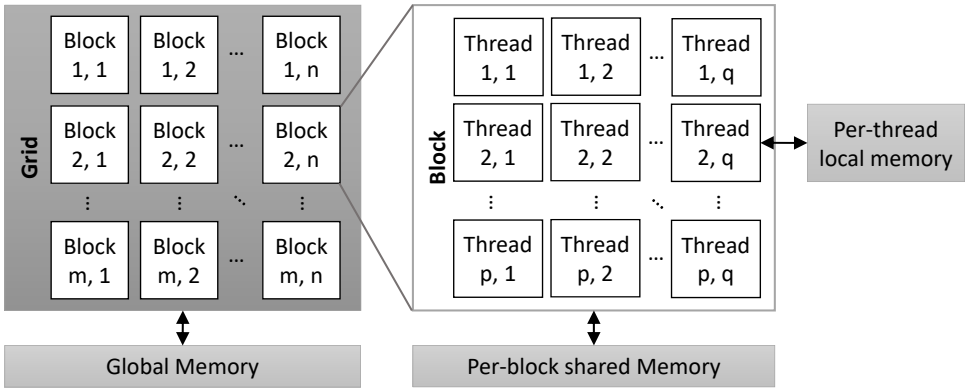


Figure 4.5. Thread-blocks and memory hierarchy in CUDA

purpose. In GPGPU, one first picks out the part of the computing tasks that are suitable for GPU and originally handled by CPU. By reprogramming the associated part, these computing tasks are then assigned to GPU, which works as co-processor attached to CPU. The above procedure can be realized by a specific implementation of GPGPU computing platform, such as OpenCL, CUDA, Stream, etc.

CUDA, which stands for *Compute Unified Device Architecture*, is a widely used GPGPU computing platform and programming model developed by NVIDIA since 2006. In the current work, the two-scale damage model is implemented in CUDA programming language, which is an extension to many popular languages, such as C/C++. In terms of CUDA programming model, CPU and memory are called host while GPU and graphics memory are referred to as device. Two sets of codes are developed for host and device respectively. The host codes running on CPU are responsible for transferring data between host and device as well as launching device codes termed as kernels. The kernels are executed by GPU while the parallel threads are managed hierarchically by a thread-block-grid structure, i.e., a certain number of threads form a block and so forth, which is illustrated in Figure 4.5. A unique index to each thread can be calculated by the built-in `blockID` and `threadID` variables. Figure 4.5 also shows the memory hierarchy in CUDA, which is similar to its underlying hardware implementation illustrated in Figure 4.4.

4.5.2 CUDA implementation of the two-scale model

Table 4.1 summarized the pseudocode for the host program that running on CPU. It first allocates memory on device to store the displacement solution and the internal state variables that will be copied from the host memory. Since GPU has a quite high memory bandwidth and the size of data is only of $\mathcal{O}(n_g)$, time cost due to data transferring could be neglected. The host code then invokes the GPU kernel with two launching parameters `blocksPerGrid` and `threadsPerBlock` respectively. Once the GPU kernel is completed, the host code copies the updated state variables from the device memory and then cleans up. Line 3 in Table 4.1 launches the GPU kernel. Since both the blocks and threads are organized in a one-dimensional fashion in the current work, multiplication of `blocksPerGrid` and `threadsPerBlock` gives the total number of threads that are launched in GPU. This number will be slightly larger than number of threads that is needed if the number of Gauss points is not a multiple of `threadsPerBlock`. There are many factors affecting the choice of `threadsPerBlock`. Presently this choice is made based on the following considerations: (1) It should be a multiple of 64 from the viewpoint of performance; (2) Small values will result in an insufficient GPU occupancy; (3) Resources available within a block is limited, such as registers and other on-chip memory; (4) The maximum value is either 512 or 1024 based on different architectures of GPU and (5) Values between 128 and 256 are suggested by CUDA's official optimization manual and have no significant impact on the performance (Takahashi and Hamada, 2009). In the current work, we chose `threadsPerBlock` = 128 by considering the above factors.

The pseudocode for the device program that running on GPU is summarized in Table 4.2. This pseudocode is a prototype implementation of the two-scale damage model, which is referred to as the prototype GPU kernel. Here are some explanations on the pseudocode for prototype GPU kernel:

Table 4.1. Pseudocode for the host program

Line number	Operation
1	Allocate memory on device
2	Copy mesoscale solution and internal state variables from host to device
3	Launch kernel<<<`blocksPerGrid`, `threadsPerBlock`>>>
4	Copy updated internal state variables from device to host
5	Release allocated device memory

Table 4.2. Pseudocode for the device program (prototype kernel)

Line number	Operation
1	Calculate index by `blockID` and `threadID`
2	If index (0-based) < number of Gauss points
3	Loop over temporal interpolation points
4	Interpolate displacement solution
5	Calculate strain and stress at mesoscale
6	Elastic prediction at microscale
7	If yield function is satisfied
8	Plastic correction at microscale
9	Update internal state variables at microscale
10	If energetic damage threshold satisfied
11	Update damage at microscale
12	End If
13	End If
14	End Loop
15	End If

- Line 1 determines the unique index of each thread by the built-in block and `threadID` variables;
- Line 2 prevents the execution of kernel for excessively issued threads by comparing the thread index with the number of Gauss points;
- Lines 4 and 5 perform the temporal interpolation on mesoscale displacement solution and calculation of stress/strain, and
- Lines 6 $\sim$ 13 evaluate the two-scale damage model.

4.5.3 Optimizations

Two optimization techniques are introduced in current work to improve the computational performance of the GPU-accelerated two scale damage model. The first one is data caching technique for improving the efficiency of prototype GPU kernel. The second optimization is adapting the GPU code to computing platforms equipped with multiple GPUs.

4.5.3.1 Data caching

The data transferred to the device by the host program is stored on graphics memory, which also known as off-chip global memory. It takes about $400 \sim 600$ clock cycles when the threads access data from the global memory. Although memory latency can be hidden by higher occupancy of GPU to some extent and the GPUs nowadays have some built-in implicit data caching techniques, the frequent global memory access within a loop body slows down the computation. In the present case, the prototype GPU kernel requires a total of $\mathcal{O}(n_i)$ times access to the global memory for each thread. To resolve this issue, we proposed an explicit data caching technique to minimize performance degradation due to memory latency.

Figure 4.6 presents a comparison between the prototype and the optimized GPU kernels, while the pseudocode of the optimized GPU kernel is given in Table 4.3. At the beginning of each GPU thread, as shown in line 3, data corresponding to the current Gauss point is copied from global memory to on-chip registers. Then the variables stored in registers can be accessed rapidly from the loop body (line 4) for evaluating the two-scale damage model. Finally, the results are copied from registers to global memory (line 5) and transferred back to host memory. With this data caching technique, the time spent on accessing global memory by each thread is reduced from $\mathcal{O}(n_i)$ to $\mathcal{O}(1)$, and the impact of memory access latency is minimized.

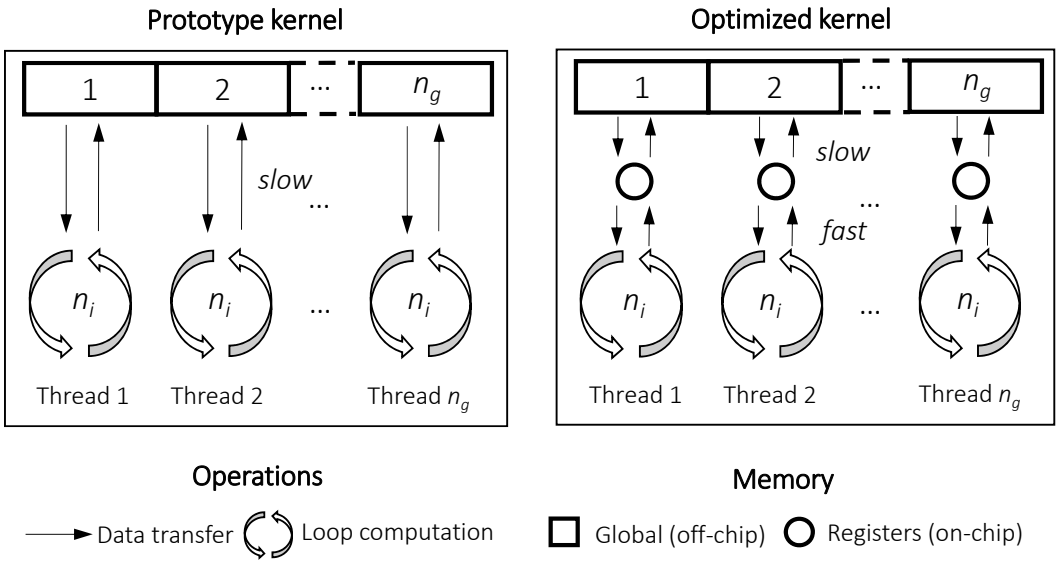


Figure 4.6. Comparison between the prototype and optimized GPU kernels

Table 4.3. Pseudocode for the device program (optimized kernel)

Line number	Operation
1	Calculate index by `blockID` and `threadID`
2	If index (0-based) $<$ number of Gauss points
3	Copy data from global memory to registers
4	Same with lines $3 \sim 14$ in the prototype kernel
5	Copy data from registers to global memory
6	End If

4.5.3.2 Multiple GPUs

CUDA provides several features to facilitate multi-GPUs programming, thus we can further extend the GPU-accelerated two scale model to adapt multi-GPUs platforms. In practice, this extension is implemented by modifying the host program in a multi-CPU-processes/multi-GPUs fashion.

Table 4.4 shows the pseudocode for the extended host program. Since the size of total computing task is related to the number of Gauss points, we first divide the Gauss points

Table 4.4. Pseudocode for the host program on a multi-GPUs platform

Line number	Operation
1	Divide Gauss points into N_{GPU} groups
2	Parallel loop over the groups using MPI or OpenMP
3	`cudaSetDevice(groupID)`
4	Allocate memory on Device(`groupID`)
5	Copy data[`groupID`] from host to Device(`groupID`)
6	Launch kernel<<<`blocksPerGrid`, `threadsPerBlock`>>>
7	Copy results from Device(`groupID`) to host
8	Release allocated memory on Device(`groupID`)
9	End Parallel

into N_{GPU} groups (line 1), where N_{GPU} is the number of available GPUs. In terms of load balancing, the number of Gauss points in each group is determined by the computational capability of the corresponding GPU. Then, as shown in line 2, N_{GPU} CPU processes are invoked in parallel by using either MPI or OpenMP to distribute the computing tasks to each GPU. For each CPU threads, we assign a dedicated GPU by using the CUDA function `cudaSetDevice` (line 3), which is an API function provided by CUDA. The rest of the extended host program shown in lines $4 \sim 8$ are the same as the original code, except that only the computing task related to the assigned group of Gauss points will be performed by each CPU thread and the dedicated GPU.

4.6 Hybrid CPUs/GPUs parallel acceleration

Following the previously developed parallel computing framework for XTFEM (see Section 3.6.2), a hierarchy of parallelisms is also established for the two-scale damage model. Similarly, the first-level of parallelism arises from domain partitioning. However, communications among MPI processes are not required since it is a SIMD-type computing task. At the second-level of parallelism, an element-wise multithreading is employed. In addition to the CUDA (GPU) version in the previous section, we also developed an OpenMP (CPUs)

Table 4.5. Pseudocode for the OpenMP version program

Line number	Operation
1	!$OMP PARALLEL DO
2	Loop over local spatial Gauss points
3	Loop over temporal interpolation points
4	Interpolate displacement solution
5	Calculate strain and stress at mesoscale
6	Elastic prediction at microscale
7	If yield function is satisfied
8	Plastic correction at microscale
9	Update internal state variables at microscale
10	If energetic damage threshold satisfied
11	Update damage at microscale
12	End If
13	End If
14	End Loop
15	End Loop
16	!$OMP END PARALLEL DO

version of the parallel implementation of the two-scale damage model, which is summarized in Table 4.5. The first and last lines are the OpenMP directives (in FORTRAN format). Without those directives, it reduces to a serial code. Note that only the outer loop over the Gauss points on subdomain is expanded by OpenMP directive.

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

4.7 Numerical example

Now we consider an HCF life prediction problem of a single edge notched plate made of 304L stainless steel. Material parameters are given as density $\rho = 7860$ kg/m^3, Young's

Table 4.6. Two-scale damage model parameters

C_y (MPa)	σ_u (MPa)	ε_{pD}	σ_f^∞ (MPa)	h	D_c	S	s
1,740	577	0.08	180	0.2	0.3	0.5	0.5

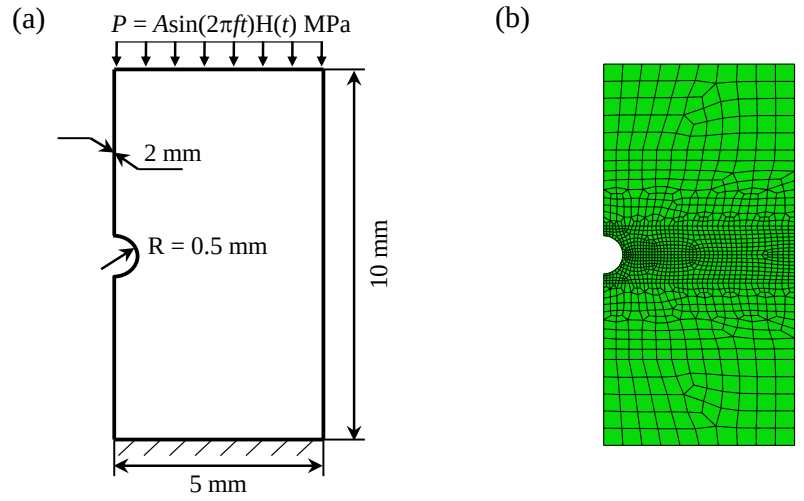


Figure 4.7. HCF of single edge notched plate: (a) geometric model and boundary conditions; (b) 2D spatial mesh

modulus $E = 197$ GPa and Poisson's ratio $\nu = 0.3$. Table 4.6 shows the parameters of the two-scale damage model given by (Desmorat et al., 2007; Bhamare et al., 2014), which have been verified with experimental results (Vincent et al., 2012).

Figure 4.7 (a) shows the geometric model and boundary conditions of the single edge notched specimen. A constant amplitude, fully-reversed cyclic fatigue load is uniformly applied on top side of the specimen, which is $P = A\sin(2\pi ft)H(t)$ MPa with $f = 20$ Hz. The bottom side of the plate is fixed. As shown in Figure 4.7 (b), 1,208 bilinear quadrilateral (Q4) 2D elements with plane stress formulation are used for spatial domain discretization. The element size at the notch root is 0.05 mm, which was determined by a mesh convergence study (Bhamare, 2012). The number of DOFs for this problem is 15,240. Before crack initiates, the XTFEM time step, or equivalently the size of the space-time slab, is $100\,T$,

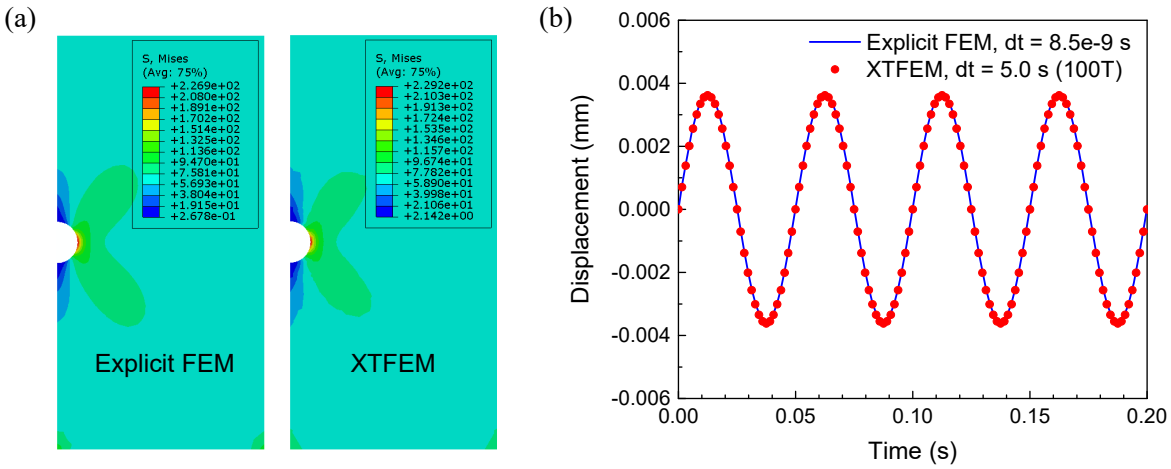


Figure 4.8. Comparison of explicit FEM and XTFEM solutions for single edge notched plate ($A = 70$ MPa): (a) von-Mises stress; (b) displacement history

where T is the period of the applied loading cycle. Once fatigue crack initiated, the time step is reduced to $10\,T$ to capture the propagating crack.

For the purpose of comparison, we employed ABAQUS/Explicit solver for the first few cycles without the damage model. Figure 4.8 (a) shows the contour plot of maximum mesoscale von-Mises stress under load amplitude $A = 70$ MPa. Displacement history on the loading edge is shown in Figure 4.8 (b). It is clearly illustrated that the results obtained by XTFEM agree well with the explicit FEM results.

Figure 4.9 shows the results of XTFEM simulations on HCF of the single edge notched sample. Fatigue crack propagation is shown in Figure 4.9 (a). The crack initiated at the notch root and propagated to the critical crack length, which is half width of the specimen. Exponential crack growth shown in Figure 4.9 (b) is consistent with the trends that are observed in the experiments on fatigue crack growth. The typical HCF behavior that most of the fatigue life is consumed by crack initiation is also effectively captured and clearly demonstrated in Figure 4.9 (c).

For parallel acceleration, both dual CPUs (Intel Xeon E5-2667, 3.5 GHz, 6 cores per CPU) and dual GPUs (NVIDIA TESLA K20c, 2496 CUDA cores per GPU, peak double-precision

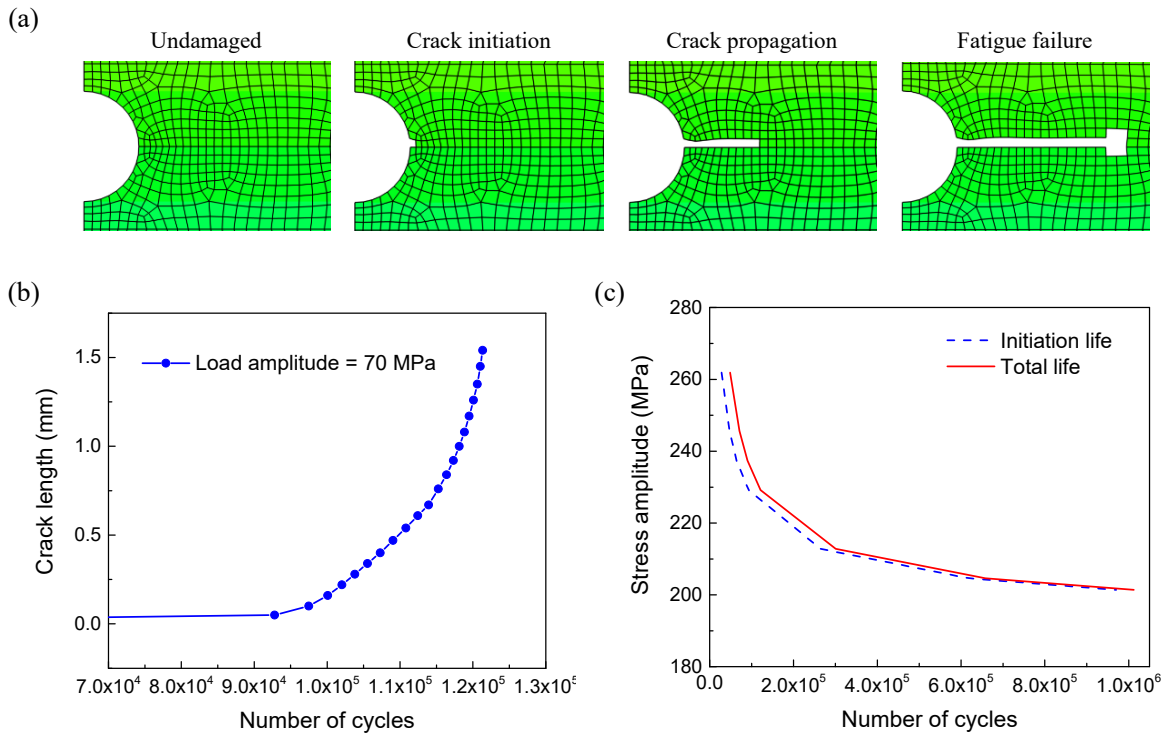


Figure 4.9. Results of XTFEM simulations on HCF of the single edge notched plate: (a) fatigue crack path; (b) crack length vs. cycles; (c) S-N curve

performance of 1.17 TFLOPS) are employed in this numerical example. The computational time by the OpenMP code that runs on a single CPU core is taken as a reference to calculate the speedup ratio. Figure 4.10 (a) shows that the speedup of the OpenMP code scales linearly with the number of CPU cores. By using 12 CPU cores, a speedup ratio of 11 is obtained. The speedup ratio of the prototype GPU code is 57, while the optimized code achieved a much higher speedup ratio of 126. By using two GPUs, the speedup ratio is further increased to 214. In addition, by varying the number of Gauss points, the performance of the optimized GPU code is illustrated in Figure 4.10 (b). It shows that the speedup ratio is low for small number of Gauss points. The poor performance is due to the insufficient GPU occupancy. However, the performance improves rapidly with the increasing of the number of Gauss points and finally reaches a plateau. Thus, we conclude that the proposed HPC

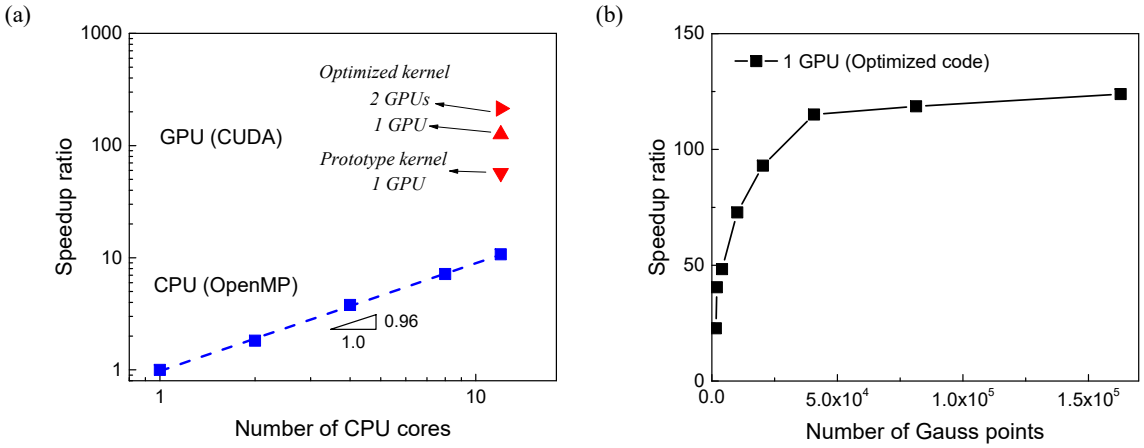


Figure 4.10. Speedup ratio of the two-scale damage model by parallel computing

implementation has a good parallel scalability and efficiently accelerates the evaluation of the two-scale damage model.

4.8 Summary

In this chapter, we introduced the formulation and numerical implementation of a CDM-based two-scale HCF damage model proposed by Desmorat et al. (Desmorat et al., 2007). This model is integrated with the XTFEM as a nonlinear constitutive solver to account for material degradation under cyclic loading conditions, which leads to a complete multiscale computational framework for HCF life prediction applications. Computational cost analysis shows that the evaluation of the damage model is a bottleneck in the framework due to the high-resolution requirement in terms of spatial-temporal quadrature points. HPC parallel implementations based on CPUs (MPI/OpenMP) and GPUs (CUDA) are developed to accelerate the two-scale model. Finally, a 2D example shows that the accelerated computational framework effectively captures the physics of HCF material behavior and efficiently handles the direct numerical simulation with a good accuracy.

CHAPTER 5

DATA-DRIVEN MICROSTRUCTURE-BASED CONCURRENT MULTISCALE MATERIAL MODELING

5.1 Introduction

In Chapter 4, a two-scale damage model developed by (Desmorat et al., 2007) is integrated with XTFEM for HCF life prediction applications. In this model, an RVE is established by conceptually aggregating micro-defects such as cracks and voids as a spherical weak inclusion embedded in the elastic material at mesoscale. Plasticity and damage are accounted for on the weak inclusion where the microscale stress is derived from the mesoscale stress using the localization law. The two-scale damage model is a phenomenological constitutive model and it does not directly account for the material microstructure, which plays a key role in HCF applications (McDowell, 2007; Chan, 2010; Przybyla et al., 2010).

In this chapter, we introduce microstructure-based RVEs into HCF damage modeling. Multiscale models such as the FE^2 method (Feyel, 1999; Feyel and Chaboche, 2000; Feyel, 2003) are computationally expensive and makes HCF simulation infeasible with the current computing technology. In addition to the challenge associated with computational costs, reasonably accurate microstructure models are necessary to capture HCF failure mechanisms. In this context, it is desirable to establish a reduced-order multiscale material model without compromising the accuracy. Herein, a data-driven reduced-order multiscale model, i.e. the *Self-consistent Clustering Analysis* (SCA) proposed in (Liu et al., 2016), is employed in this work to for microstructure-based HCF material constitutive modeling.[1]

[1]We gratefully acknowledge Professor Wing Kam Liu and his group at Northwestern University for providing us access to the SCA software package.

5.2 Self-consistent Clustering Analysis

In this section, mathematical formulations of the SCA under the small deformation assumption is briefly reviewed. For more details please refer to (Liu et al., 2016).

5.2.1 RVE: the Lippmann-Schwinger equation

Considering a material microstructural RVE in domain Ω subjected to far-field macroscopic loading, the equilibrium condition in the absence of body force can be equivalently represented by the integral Lippmann-Schwinger equation in terms of local strain under the assumption of periodical boundary conditions. For any point $\boldsymbol{x} \in \Omega$, the Lippmann-Schwinger equation is given by

$$\boldsymbol{\varepsilon}(\boldsymbol{x}) + \int_{\Omega} \boldsymbol{\Phi}^0(\boldsymbol{x}, \boldsymbol{x}') : \left[\boldsymbol{\sigma}(\boldsymbol{x}') - \boldsymbol{C}^0 : \boldsymbol{\varepsilon}(\boldsymbol{x}')\right] d\boldsymbol{x}' - \boldsymbol{\varepsilon}^0 = \boldsymbol{0} \tag{5.1}$$

where $\boldsymbol{\Phi}^0(\boldsymbol{x}, \boldsymbol{x}')$ is the Green's function, $\boldsymbol{C}^0$ is the isotropic linear elastic stiffness tensor of the homogenized reference material, and $\boldsymbol{\varepsilon}^0$ is the far-field macroscopic strain.

In order to solve for local strain $\boldsymbol{\varepsilon}(\boldsymbol{x})$, macroscopic constraint is added to Eq. (5.1),

$$\frac{1}{|\Omega|} \int_{\Omega} \boldsymbol{\varepsilon}(\boldsymbol{x}) d\boldsymbol{x} = \bar{\boldsymbol{\varepsilon}} \qquad \text{or} \qquad \frac{1}{|\Omega|} \int_{\Omega} \boldsymbol{\sigma}(\boldsymbol{x}) d\boldsymbol{x} = \bar{\boldsymbol{\sigma}} \tag{5.2}$$

where $|\Omega|$ is the volume of the RVE, and the superimposed bar indicates the effective macroscale variable.

For convenience, Eq. (5.1) is rewritten in an incremental form:

$$\Delta\boldsymbol{\varepsilon}(\boldsymbol{x}) + \int_{\Omega} \boldsymbol{\Phi}^0(\boldsymbol{x}, \boldsymbol{x}') : \left[\Delta\boldsymbol{\sigma}(\boldsymbol{x}') - \boldsymbol{C}^0 : \Delta\boldsymbol{\varepsilon}(\boldsymbol{x}')\right] d\boldsymbol{x}' - \Delta\boldsymbol{\varepsilon}^0 = \boldsymbol{0} \tag{5.3}$$

To numerically solve Eq. (5.3), the RVE is first discretized with a very fine voxel mesh, which is called the high-fidelity RVE. An example of the high-fidelity RVE is illustrated in Figure 5.1, where several spherical inclusions are embedded in the matrix material. This cubical RVE is discretized by a total of 1 *million* voxels ($100 \times 100 \times 100$), where each

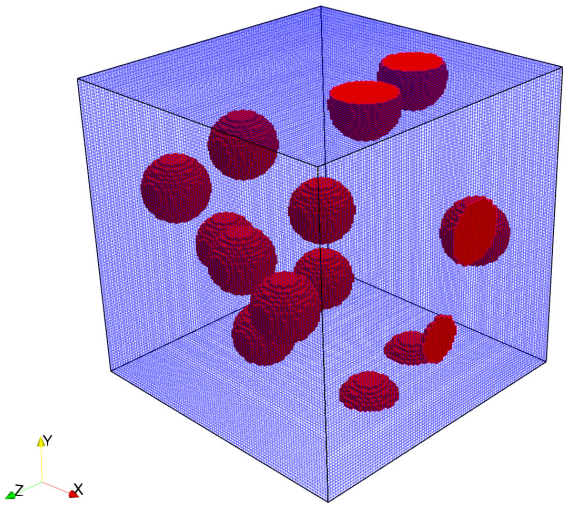


Figure 5.1. An example of high-fidelity RVE with 1 *million* voxels

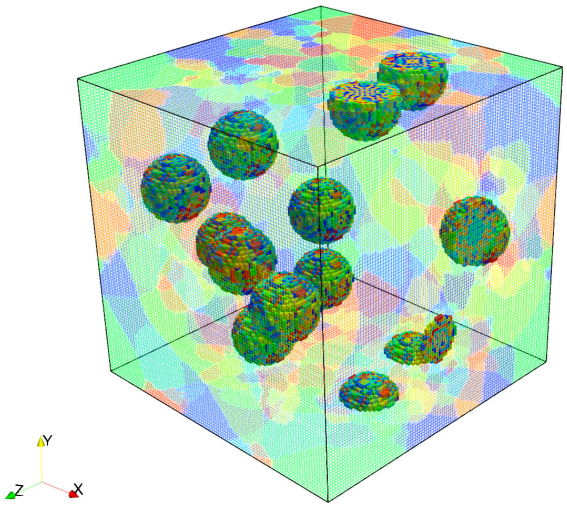


Figure 5.2. An example of reduced-order RVE with 128 clusters

voxel represents a material point in the RVE. The computational cost would be very high to directly solve Eq. (5.3) for every voxel in the high-fidelity RVE.

In order to reduce the computational cost, SCA partitions the large number of voxels in the high-fidelity RVE into a small number of voxel groups termed as *material clusters*. The partitioned RVE is called the reduced-order RVE. Figure 5.2 shows an example of the reduced-order RVE generated from Figure 5.1. The inclusion and matrix phases are both decomposed into 64 material clusters as indicated by different colors. SCA further assumes that all local variables such as $\boldsymbol{\varepsilon}(\boldsymbol{x})$ are uniform in a cluster. With that in mind, the Lippmann-Schwinger equation (5.3) is averaged for each cluster. For the I-th cluster, the reduced-order cluster-based Lippmann-Schwinger equation is given by

$$\Delta\boldsymbol{\varepsilon}^I + \sum_{J=1}^{k} \boldsymbol{D}^{IJ} : \left[\Delta\boldsymbol{\sigma}^J - \boldsymbol{C}^0 : \Delta\boldsymbol{\varepsilon}^J\right] - \Delta\boldsymbol{\varepsilon}^0 = \boldsymbol{0} \tag{5.4}$$

where k is the number of clusters, $\boldsymbol{D}^{IJ}$ is a well-defined quantity in micromechanics termed as the *interaction tensor* between the I-th and J-th cluster and is given by

$$\boldsymbol{D}^{IJ} = \frac{1}{c^I\,|\Omega|}\int_\Omega\int_\Omega \chi^I(\boldsymbol{x})\chi^J(\boldsymbol{x}')\boldsymbol{\Phi}^0(\boldsymbol{x},\boldsymbol{x}')d\boldsymbol{x}'d\boldsymbol{x} \tag{5.5}$$

in which c^I is the volume fraction of the I-th cluster, $\chi^I(\boldsymbol{x})$ is called the characteristic function and defined by

$$\chi^I(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{x} \in \Omega^I \\ 0 & \boldsymbol{x} \notin \Omega^I \end{cases} \tag{5.6}$$

Similarly, the constraints are rewritten as

$$\sum_{I=1}^{k} c^I \Delta\boldsymbol{\varepsilon}^I = \Delta\bar{\boldsymbol{\varepsilon}} \qquad \text{and} \qquad \sum_{I=1}^{k} c^I \Delta\boldsymbol{\sigma}^I = \Delta\bar{\boldsymbol{\sigma}} \tag{5.7}$$

The discretized Lippmann-Schwinger equation (5.4) and the macroscopic constraints (5.7) constitute the governing equation of the reduced-order RVE to be solved for macroscopic material points. In SCA, the solution to these equations is determined in two stages. The

first one is an a priori analysis called the offline stage, which is relatively slow but only needs to be performed once. The second stage is called the online stage and rapidly solves the equations on the reduced-order RVE under given macroscopic loading conditions.

5.2.2 Offline stage: a data-driven clustering analysis

The offline stage of SCA comprises three steps. In the first step, the high-fidelity RVE is subjected to a series of orthogonal loading conditions and linear elastic direct numerical simulations (DNS) are performed, which is similar to the RVE analysis in micromechanics. From each of these DNS, we obtain a database of mechanical response at every material point in the high-fidelity RVE. The mechanical response is characterized by the following strain concentration tensor,

$$\boldsymbol{A}(\boldsymbol{x}) = \frac{\partial \boldsymbol{\varepsilon}^{\text{micro}}(\boldsymbol{x})}{\partial \boldsymbol{\varepsilon}^{\text{macro}}} \tag{5.8}$$

After the database generation step, a domain decomposition step is carried out. In this step, material points in the high-fidelity RVE are clustered based on similarity in their mechanical responses. SCA further assumes that material points in the same cluster share exactly the same constitutive behavior within the elastic regime and always response the same way under inelastic loading conditions. It is worth noting that the cluster analysis is widely used in other fields such as data mining, data compression, and machine learning. There are many clustering algorithms readily available. In SCA, the k-means clustering algorithm is often employed (MacQueen, 1967). In this algorithm, material points are clustered solely based on their mechanical responses instead of the spatial adjacency. Hence, material points in the same cluster might be in disconnected parts of the RVE (see Figure 5.2). Also the clusters usually do not have an equivalent number of material points, which is substantially different with the conventional location-based domain decomposition techniques. A key benefit from this algorithm is that even with a relatively small number of material clusters the extreme microscopic mechanical responses in the high-fidelity RVE, which is critical in many

applications, as well as the overall macroscopic material behavior can be well captured by the reduced-order RVE. It has been extensively shown that SCA usually achieves a high accuracy with a small number of clusters (usually at most 256 clusters are sufficient) compared to DNS (Liu et al., 2016, 2018; Yu et al., 2019; Han et al., 2020). Therefore, SCA significantly reduces the total number of unknowns in RVE to achieve a satisfactory accuracy.

The last step in the offline stage is to compute the interaction tensor $\boldsymbol{D}^{IJ}$ in the cluster-based Lippmann-Schwinger equation (5.4). With periodic RVE and isotropic linear elastic reference material, Green's function in Eq. (5.5) can be expressed in a simple form in the Fourier space,

$$\hat{\boldsymbol{\Phi}}^0(\boldsymbol{\xi}) = \frac{1}{2\mu^0}\hat{\boldsymbol{\Phi}}^1(\boldsymbol{\xi}) - \frac{\lambda^0}{2\mu^0(\lambda^0+2\mu^0)}\hat{\boldsymbol{\Phi}}^2(\boldsymbol{\xi}) \tag{5.9}$$

where $\boldsymbol{\xi}$ is the Fourier coordinate corresponding to $\boldsymbol{x}$, λ^0 and μ^0 are Lamé constants of the reference material, and

$$\hat{\Phi}^1_{ijkl}(\boldsymbol{\xi}) = \frac{\delta_{ik}\xi_j\xi_l}{|\boldsymbol{\xi}|^2} \qquad \text{and} \qquad \hat{\Phi}^2_{ijkl}(\boldsymbol{\xi}) = \frac{\xi_i\xi_j\xi_k\xi_l}{|\boldsymbol{\xi}|^4} \tag{5.10}$$

where $|\boldsymbol{\xi}| = \sqrt{\xi_i\xi_i}$.

Based on Eq. (5.9), the interaction tensor can be expressed as

$$\boldsymbol{D}^{IJ} = c_1\boldsymbol{D}^{IJ}_1 + c_2\boldsymbol{D}^{IJ}_2 \tag{5.11}$$

where

$$c_1 = \frac{1}{2\mu^0} \qquad \text{and} \qquad c_2 = -\frac{\lambda^0}{2\mu^0(\lambda^0+2\mu^0)} \tag{5.12}$$

and

$$\boldsymbol{D}^{IJ}_\alpha = \frac{1}{c^I\,|\Omega|}\int_\Omega\int_\Omega \chi^I(\boldsymbol{x})\chi^J(\boldsymbol{x}')\boldsymbol{\Phi}^\alpha(\boldsymbol{x},\boldsymbol{x}')d\boldsymbol{x}'d\boldsymbol{x} \quad \alpha = 1,2 \tag{5.13}$$

In the offline stage, only the material independent parts of the interaction tensor, i.e. Eq. (5.13), are evaluated by using the *Fast Fourier Transform* (FFT) algorithm. The full interaction tensor is calculated using Eq. (5.11) in the online stage with the updated reference material constants.

The above offline stage of SCA is analogous to the model training or learning stage in the data-driven *Machine Learning* method. In that sense, DNS of the high-fidelity RVE provides the training data set. The cluster analysis performs the model learning from the training data set. The cluster-based interaction tensor $\boldsymbol{D}_{\alpha}^{IJ}$ represents the trained reduced-order model to be employed in the rapid prediction stage.

5.2.3 Online stage: a self-consistent scheme

The online stage of SCA is integrated into macroscopic analysis and serves as a material model, e.g. UMAT or VUMAT user subroutine in ABAQUS, to be evaluated at macroscopic material points for every load increment.

The online stage solves the discrete Lippmann-Schwinger equation in an iterative self-consistent scheme. The algorithm is summarized as follows:

1. Compute the interaction tensor $\boldsymbol{D}^{IJ}$ using Eq. (5.11) with updated λ^0 and μ^0;
2. Solve the cluster-based Lippmann-Schwinger equation (5.4) using Newton's method, see Appendix C for details;
3. Compute the effective tangent stiffness tensor (usually anisotropic) of the reduced-order RVE by
$$\bar{\boldsymbol{C}} = \sum_{I=1}^{k} c^I \boldsymbol{C}_{\text{alg}}^I : \boldsymbol{A}^I \tag{5.14}$$
where
$$\boldsymbol{C}_{\text{alg}}^I = \frac{\partial \Delta \boldsymbol{\sigma}^I}{\partial \Delta \boldsymbol{\varepsilon}^I} \qquad \text{and} \qquad \boldsymbol{A}^I = \frac{\partial \Delta \boldsymbol{\varepsilon}^I}{\partial \Delta \boldsymbol{\varepsilon}^0} \tag{5.15}$$
are tangent stiffness tensor and strain concentration tensor of the I-th cluster, respectively. Both of them can be computed from the converged Newton iteration in Step 2, see Appendix C for details;
4. Compute the approximate isotropic linear elastic Lamé constants of the reference material by
$$\lambda^0 = \frac{1}{3}\left(\bar{\boldsymbol{C}} :: \boldsymbol{J} - \frac{1}{8}\bar{\boldsymbol{C}} :: \boldsymbol{K}\right) \qquad \text{and} \qquad \mu^0 = \frac{1}{16}\bar{\boldsymbol{C}} :: \boldsymbol{K} \tag{5.16}$$

where $\boldsymbol{J} = \frac{1}{3}\boldsymbol{I}_2 \otimes \boldsymbol{I}_2$ and $\boldsymbol{K} = \boldsymbol{I}_4 - \boldsymbol{J}$, in which $\boldsymbol{I}_2$ and $\boldsymbol{I}_4$ are the identity tensors of rank 2 and 4, respectively;

5. Repeat the previous steps until the reference material constants are converged;
6. Update the output variables (such as the stress tensor and tangent stiffness matrix) and continue to the next load increment.

In the above algorithm, the second step is the most critical and computationally expensive one. In this step, material constitutive laws are evaluated for each cluster to obtain the incremental stress tensor $\Delta\boldsymbol{\sigma}$, the local tangent stiffness tensor $\boldsymbol{C}_{\text{alg}}$, and other material-dependent state variables. Detailed algorithm for this step can be found in Appendix C.

5.3 Data-driven fatigue damage modeling

For HCF or ultra HCF (UHCF) applications, material microstructures play a critical role and multiscale modeling is often a necessity to study fatigue failure mechanisms (McDowell, 2007; Chan, 2010; Przybyla et al., 2010; Przybyla and McDowell, 2010, 2011; Gillner and Münstermann, 2017). However, concurrent multiscale modeling with detailed microstructural features is usually computationally too expensive for HCF simulations. The reduced-order SCA introduced in the previous section provides an ideal tool for efficient concurrent multiscale fatigue damage modeling. In this section, we establish a data-driven microstructure-based fatigue damage modeling procedure based on SCA. Figure 5.3 illustrates a conceptual sketch of the proposed multiscale HCF damage model with SCA. Similar to the two-scale damage model, macroscale or mesoscale analysis is assumed to be elastic due to the relatively low load magnitude in HCF applications. The microscale analysis is performed by the reduced-order RVE based on SCA. To model the microscale plastic deformation and damage under cyclic fatigue loading, material clusters in the RVE are governed by coupled plasticity-damage constitutive laws.

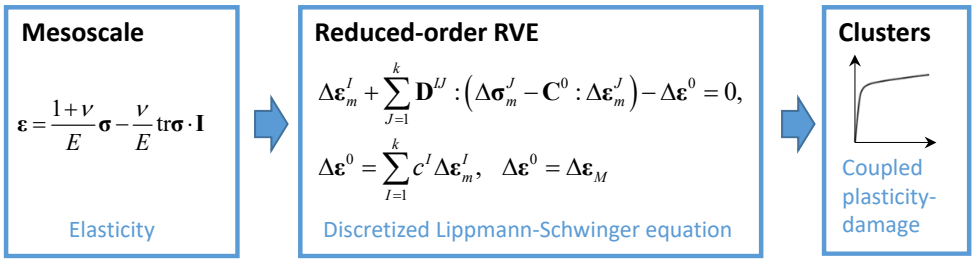


Figure 5.3. A sketch of the HCF material model using SCA

5.3.1 From microstructure to RVE

The first step in the multiscale fatigue damage modeling is to establish RVEs based on material microstructures, which has been extensively studied in the past (Drugan and Willis, 1996; Kanit et al., 2003; Ostoja-Starzewski, 2006; Gitman et al., 2007; Amirmaleki et al., 2016).

There are several ways to numerically generate the RVEs. They can be directly reconstructed from 2D or 3D scanned images of material microstructures, which is straightforward and accurate but could be expensive due to the sheer volume of data to be generated and processed. They can also be generated numerically based on the statistics from experiment characterizations to mimic the microscale morphology. For composite materials, RVEs can be established based on designed microstructures. In some cases, RVEs are simply created with assumed microstructures. See (Bargmann et al., 2018) for a recent review on RVE generation techniques.

For HCF applications, typical microstructure features to be considered are defects such as voids and cracks, inclusions, grain boundaries, etc. See (Moore et al., 2016; Kafka et al., 2018) for some microstrcutrue-based RVE examples in HCF life prediction. For a particular material type, the RVE needs to be established based on the observed material microstructures and the identified microstructural features critical to HCF life. For example, when microdefects present in a material microstructure, they lead to high stress concentration and tend to be the weakest spots from where fatigue cracks initiate.

In this work, we focus on establishing a generic concurrent multiscale HCF modeling method. Hence, simplified RVEs are employed for convenience, see Figure 5.1 for an example. Once the RVE is generated, the offline stage of SCA is conducted to train the reduced-order database for the online prediction stage.

5.3.2 Material laws for clusters

In the online prediction stage of SCA, the cluster-based Lippmann-Schwinger equation is solved at the microscale, which captures interactions between microstructural features as well as local mechanical responses in every material clusters. Note that arbitrary local material constitutive models can be employed for each cluster. For HCF applications, it is important to capture the permanent plastic deformation and the resulting damage in material accumulated under large numbers of loading cycles. As an illustration, here we employ a simple plasticity model with linear kinematic hardening and coupled with a damage model based on the equivalent plastic strain. Note that other material models can also be incorporated when appropriate, such as the crystal plasticity model (Anahid et al., 2011; Ghosh and Chakraborty, 2013; Moore et al., 2016; Liu et al., 2018).

A simple coupled plasticity-damage model Here we consider a simple linear kinematic hardening plasticity model for local clusters. First the isotropic linear elastic equation is given by

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon}^e + \lambda \mathrm{tr}(\boldsymbol{\varepsilon}^e)\boldsymbol{I} \tag{5.17}$$

where λ and μ are Lamé parameters, $\boldsymbol{\varepsilon}^e$ is the elastic strain tensor.

For the plasticity part, the von Mises yield function, the equivalent plastic strain rate, the associated plastic flow law, and the linear (Prager-Ziegler) kinematic hardening rule are respectively given by

$$\sqrt{\frac{3}{2}(\boldsymbol{s}-\boldsymbol{\alpha}) : (\boldsymbol{s}-\boldsymbol{\alpha})} - \sigma_y = 0 \tag{5.18}$$

$$\dot{\bar{\varepsilon}}^p = \sqrt{\frac{2}{3}\dot{\boldsymbol{\varepsilon}}^p : \dot{\boldsymbol{\varepsilon}}^p}, \quad \dot{\boldsymbol{\varepsilon}}^p = \frac{3}{2}\frac{\dot{\bar{\varepsilon}}^p}{\sigma_y}(\boldsymbol{s} - \boldsymbol{\alpha}), \quad \dot{\boldsymbol{\alpha}} = \frac{2}{3}C_y\dot{\boldsymbol{\varepsilon}}^p \tag{5.19}$$

where $\boldsymbol{s} := \boldsymbol{\sigma}^{\text{dev}}$ is the stress deviator tensor, $\boldsymbol{\alpha}$ is the backstress tensor, σ_y is the yield stress, $\boldsymbol{\varepsilon}^p$ is the plastic strain tensor, and C_y is the plastic modulus.

For the damage part, we employ the Lamaitre damage evolution law in the previous two-scale damage model. See Eq. (4.13) in Section 4.2.

Numerical integration The above equations are integrated numerically in an implicit Backward Euler scheme. First, the elastic predictor is computed by

$$\bar{\sigma}^{pr} = \sqrt{\frac{3}{2}(\boldsymbol{s}^{pr} - \boldsymbol{\alpha}^o) : (\boldsymbol{s}^{pr} - \boldsymbol{\alpha}^o)}, \quad \boldsymbol{s}^{pr} = \boldsymbol{s}^o + 2\mu\Delta\boldsymbol{\varepsilon}^e \tag{5.20}$$

where superscript "pr" indicates the elastic prediction and superscript "o" indicates the initial state or the previous load increment.

When the elastic predictor is greater than the yield stress, plastic flow occurs and the incremental plastic strain tensor is obtained by

$$\Delta\boldsymbol{\varepsilon}^p = \frac{3}{2}\Delta\bar{\varepsilon}^p\boldsymbol{\eta} \tag{5.21}$$

where $\boldsymbol{\eta} = (\boldsymbol{s}^{pr} - \boldsymbol{\alpha}^o)/\bar{\sigma}^{pr}$ is the plastic flow direction and the equivalent plastic strain increment is

$$\Delta\bar{\varepsilon}^p = \frac{\bar{\sigma}^{pr} - \sigma_y}{C_y + 3\mu} \tag{5.22}$$

Subsequently the state variables are updated by

$$\boldsymbol{\alpha} = \boldsymbol{\alpha}^o + C_y\Delta\bar{\varepsilon}^p\boldsymbol{\eta}, \quad \boldsymbol{\sigma} = \boldsymbol{\alpha} + \sigma_y\boldsymbol{\eta} + (\boldsymbol{\sigma}^{pr})^{\text{hyd}} \tag{5.23}$$

and the consistent tangent stiffness tensor can be extracted from

$$\Delta\dot{\boldsymbol{\sigma}} = 2\mu'\Delta\dot{\boldsymbol{\varepsilon}} + \lambda'\text{tr}(\Delta\dot{\boldsymbol{\varepsilon}})\boldsymbol{I} + \left(\frac{C_y}{1 + C_y/3\mu} - 3\mu'\right)(\boldsymbol{\eta} : \Delta\dot{\boldsymbol{\varepsilon}})\,\boldsymbol{\eta} \tag{5.24}$$

where $\mu' = \mu\left(\sigma_y + C_y\Delta\bar{\varepsilon}^p\right)/\bar{\sigma}^{pr}$ and $\lambda' = K - \frac{2}{3}\mu'$, in which K is the bulk modulus.

Finally, the damage variable is updated as a function of the equivalent plastic strain increment $\Delta\bar{\varepsilon}^p$, see Section 4.3.3 for details.

5.3.3 An efficient solution scheme for HCF

Computational performance is always a critical consideration in HCF simulations. Although SCA greatly reduces the computational expense of RVE simulations, it could be expensive or even infeasible for direct numerical HCF simulations given the current computer technology. In fact, the interaction tensor in SCA is typically a dense matrix and the computational complexity of SCA online stage is at $\mathcal{O}(k^3)$, where k is the number of clusters (Yu et al., 2019). In comparison to the previous two-scale damage model, in which only one coupled plasticity-damage constitutive law is resolved for each macroscopic material point, SCA needs to evaluate k local material laws, formulate the Jacobian matrix (which is also dense) and solve the matrix equation for every Newton iteration, and this whole Newton process is embedded in an outer loop of the self-consistent iteration scheme. Direct implementation of SCA would be orders of magnitude more expensive than the two-scale damage model. In this section we present an efficient solution scheme to accelerate the SCA online stage for HCF applications.

First, as we already explained before, load magnitudes in HCF are usually moderate such that the material is loaded at a stress level below the yield stress. In other words, macroscopic material mechanical responses are within the elastic regime while the plasticity and damage occur at smaller scales. In addition, HCF life is typically dominated by the crack initiation stage. Hence, we assume that the macroscale material parameters remain constant during the HCF simulation and the macroscopic material point is eroded once the corresponding RVE fails due to damage. Subsequently, we no longer need to update the reference material in the SCA online stage. Only the second step in SCA online stage (see Section 5.2.3), i.e. solving the cluster-based Lippmann-Schwinger equation, is performed to evolve the micro-plasticity and micro-damage for each load increment at macroscale. Therefore, computational expense is saved by avoiding the iterations for updating the reference material constants.

To further accelerate the SCA online stage, we propose a quasi-Newton method featured with a Jacobian recycling scheme to replace the full Newton method for solving the cluster-based Lippmann-Schwinger equation (Appendix C). Such an acceleration technique is rooted in the cyclic nature in material mechanical responses under fatigue loading and the similarity in responses shared by adjacent loading cycles. It has been employed in our previous work on HCF life prediction of rubber materials (Wada, 2017; Wada et al., 2018). This Jacobian-recycling quasi-Newton algorithm is summarized as follows:

1. Assume there are n time increments per each loading cycle;
2. For a given loading cycle, solve the corresponding n cluster-based Lippmann-Schwinger equations with the full Newton method and save all the Jacobian matrices $\boldsymbol{\mathcal{M}} = \{\boldsymbol{M}_1, \boldsymbol{M}_2, ..., \boldsymbol{M}_n\}$;
3. For subsequent loading cycles, solving the i-th Lippmann-Schwinger equation by quasi-Newton method using the corresponding recycled Jacobian matrix $\boldsymbol{M}_i$;
4. If the quasi-Newton method in Step 3 failed to converge, then switch back to the full Newton method and update & recycle the corresponding Jacobian matrix.

In Step 3 of the above algorithm, the quasi-Newton method saves the computational costs associated with computing the local tangent stiffness tensors, assembling the Jacobian matrix, and solving the matrix equation (assuming that $\boldsymbol{M}_i^{-1}$ is also recycled). This algorithm would be much faster compared to the full Newton method in the expense of more memory usage and degraded convergence rate.

5.4 Microstructure-based HCF material modeling: an example

In this example, we model a high-carbon chromium bearing steel using the proposed concurrent multiscale method and study its inclusion induced HCF life. The material microstructure and elastic properties are extracted from (Gu et al., 2019).

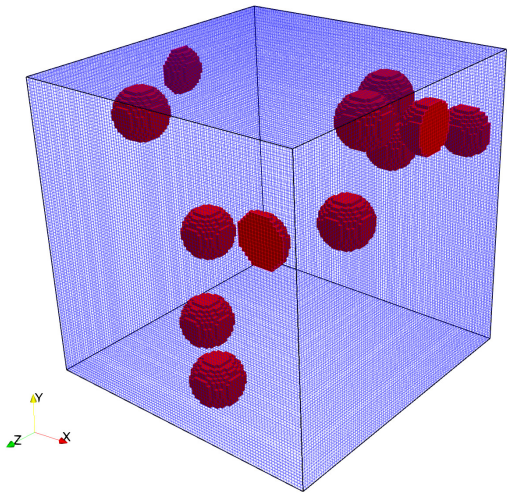


Figure 5.4. High-fidelity RVE of the steel matrix with spherical Calcium aluminate inclusions

5.4.1 RVE generation

In (Gu et al., 2019), the microstructural RVE is generated in two steps. First, the grain structure of the steel matrix is created based on the electron backscatter diffraction (EBSD) data and a statistical approach. Then, Calcium aluminate inclusions based on scanning electron microscope (SEM) images are inserted into the matrix. They found that the diameters of inclusions, which are observed in fatigue crack initiation sites, are ranged from 12.5 μm to 33.2 μm.

In this work, we generate a simplified RVE with the size of $100 \times 100 \times 100$ μm^3. The steel matrix is assumed to be homogeneous. Spherical Calcium aluminate inclusions with a diameter of 16 μm are randomly inserted into the RVE in a periodic fashion. The total volume fraction of the inclusions is assumed to be 2%. The high-fidelity RVE is discretized by 1 *million* voxels, which is illustrated in Figure 5.4.

Table 5.1. Elastic properties of the steel matrix and the Calcium aluminate inclusion

Material	**Young's modulus (GPa)**	**Poisson's ratio**
Steel	206	0.30
Calcium aluminate	113	0.23

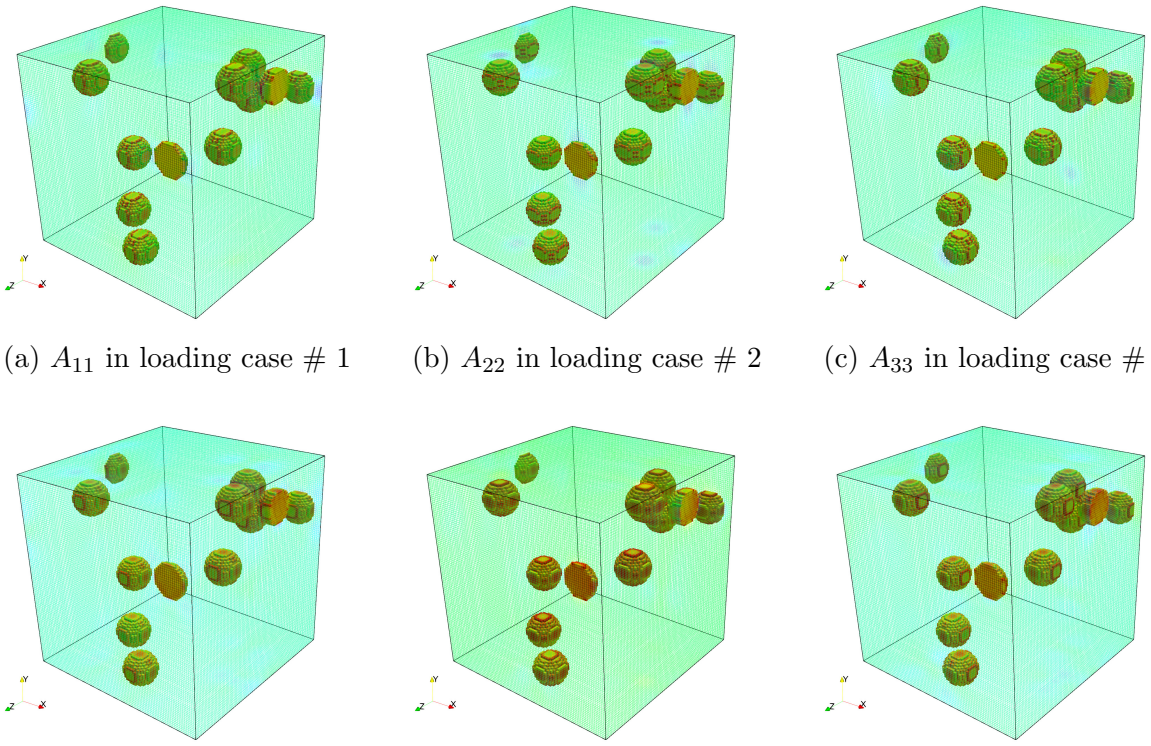
Table 5.2. Macroscopic strain constraints in RVE mechanical analysis

Loading case #	**Direction**	**Strain tensor**
1	11	$\{0.001, 0, 0, 0, 0, 0\}$
2	22	$\{0, 0.001, 0, 0, 0, 0\}$
3	33	$\{0, 0, 0.001, 0, 0, 0\}$
4	23	$\{0, 0, 0, 0.001, 0, 0\}$
5	13	$\{0, 0, 0, 0, 0.001, 0\}$
6	12	$\{0, 0, 0, 0, 0, 0.001\}$

5.4.2 Offline training stage

First, linear elastic mechanical analysis is conducted based on the high-fidelity RVE. The material parameters are given in Table 5.1. The RVE is subjected to 6 orthogonal macroscopic strain loading conditions summarized in Table 5.2 with periodical boundary conditions. The RVE analysis is performed by using FFT-based micromechanics. The resulting strain concentration tensors are illustrated in Figure 5.5. For each loading case, the component of strain concentration tensor corresponding to the loading direction is presented. In Figure 5.5, the blue to red rainbow colormap is employed.

Next, reduced-order RVEs are generated by using k-means clustering algorithm and the database of strain concentration tensors. Here, we created 3 reduced-order RVEs with different number of material clusters. In the coarsest case, there are 4 clusters in each material phase of the RVE and a total of 8 clusters. Similarly, there are in total 32 and 128 material clusters in the medium and the finest cases. These reduced-order RVEs are illustrated in Figure 5.6 and colored by cluster indices. Figure 5.7 illustrates the histograms of the mate-



(a) A_{11} in loading case # 1 (b) A_{22} in loading case # 2 (c) A_{33} in loading case # 3

(d) A_{23} in loading case # 4 (e) A_{13} in loading case # 5 (f) A_{12} in loading case # 6

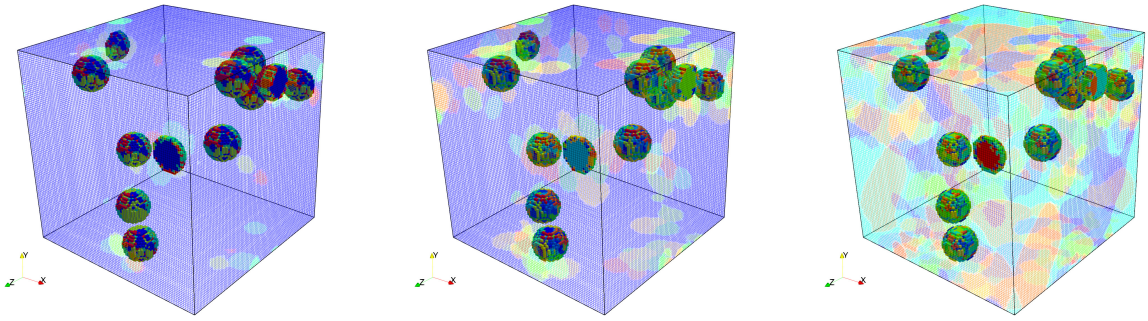
Figure 5.5. Strain concentration tensors under different loading conditions

rial volume fractions in each reduced-order RVE. A bi-modal distribution is observed for all three cases, which clearly shows that SCA employs a response-based domain decomposition strategy instead of a location-based one.

Finally, the material-independent interaction tensors (see Eq (5.13) in Section 5.2.2) are computed using FFT for the above reduced-order RVEs.

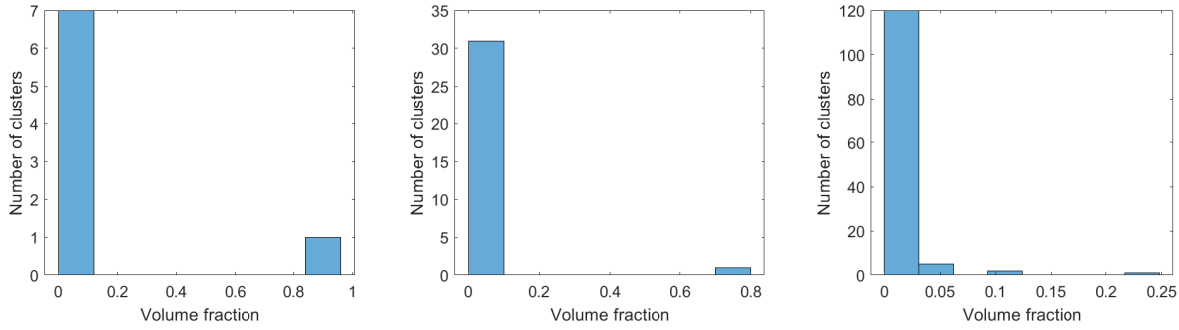
5.4.3 Online prediction stage

First, we apply a strain-controlled uniaxial tensile loading condition in the x direction to the reduced-order RVEs. We assume the inclusions are elastic. The linear kinematic hardening plasticity law is employed for the matrix clusters. The yield strength and hardening modulus are assumed to be 2.4 GPa and 14.7 GPa, respectively. The resulting effective macroscopic



(a) $k = 8$ (b) $k = 32$ (c) $k = 128$

Figure 5.6. Reduced-order RVEs with different number of material clusters



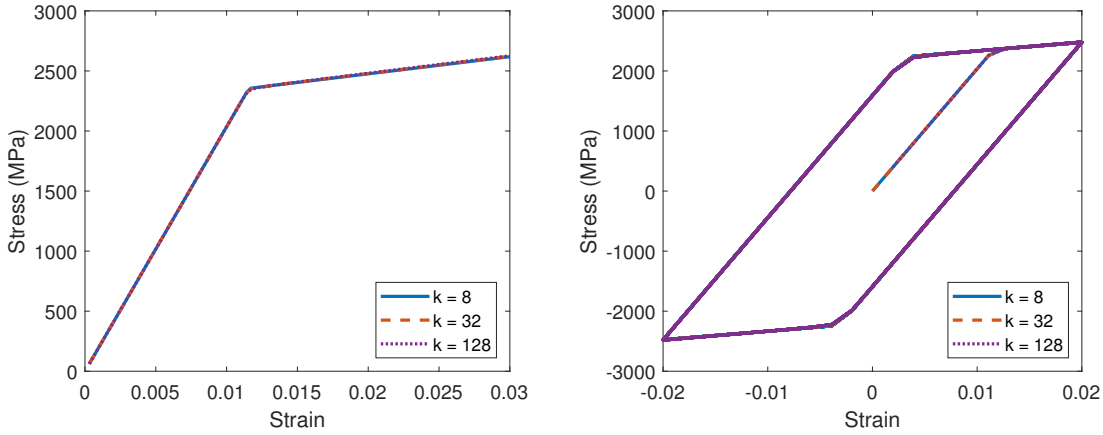
(a) $k = 8$ (b) $k = 32$ (c) $k = 128$

Figure 5.7. Histograms of cluster volume fractions in the reduced-order RVEs

stress-strains curves are plotted in Figure 5.8 (a). It can be seen that all three reduced-order RVEs lead to the same elasto-plastic macroscopic material response under uniaxial tensile loading condition.

Similarly, a strain-controlled, constant amplitude ($\varepsilon = 0.02$), and fully reversed ($R = -1$) cyclic loading condition in the x direction is applied to the RVEs for low cycle fatigue simulation. Here, the first 10 loading cycles are simulated with 64 increments per cycle. The cyclic stress-strain curves are shown in Figure 5.8 (b). Again, all three cases lead to the same inelastic material behavior under cyclic loading.

Next, we move on to HCF simulations. The cyclic loading condition is similar to the LCF case with an exception of the strain amplitude, which is reduced to 0.01 such that the



(a) monotonic loading (b) cyclic loading

Figure 5.8. Stress strain curves under different loading conditions

macroscopic stress level is lower than the yield strength. For damage evolution in the matrix material, the following parameters are assumed: $\sigma_u = 2.6$ GPa, $\varepsilon_{pD} = 0.03$, $S = 28$, and $s = 9$. These parameters are explained in Table B.1.

Unlike the previous cases, the coarsest reduced-order RVE with 8 clusters failed in predicting any plasticity and damage under the HCF loading condition. The mechanical responses at both scales are within the elastic regime. On the other hand, fatigue crack initiation cycles predicted by RVEs with 32 and 128 clusters are 16,619 and 8,420, respectively. The microscopic damage evolution in all reduced-order RVEs under HCF loading condition are plotted in Figure 5.9. It can be seen that exponential damage growth is captured by both RVEs with $k = 32$ and 128.

Figure 5.10 shows contour plots of field variables at the final configuration obtained by the HCF simulation with $k = 32$ RVE. The von Mises stress distribution is illustrated by Figure 5.10 (a), where stress concentrations near the inclusions are clearly observed. Figure 5.10 (b) and (c) present the equivalent plastic strain and damage distributions, respectively. The fatigue crack initiation sites are closely matched with the high plastic strain spots, which are all located surrounding the inclusions.

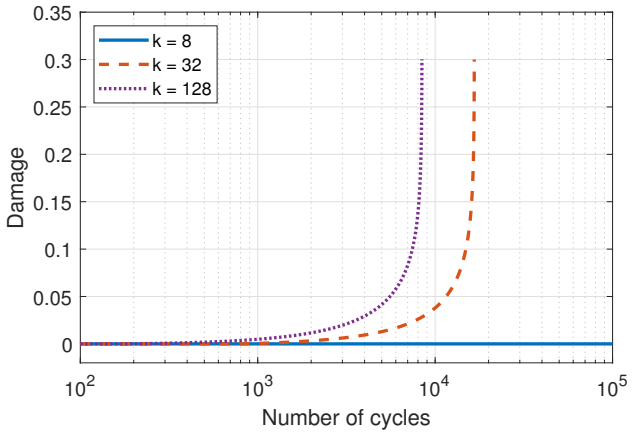
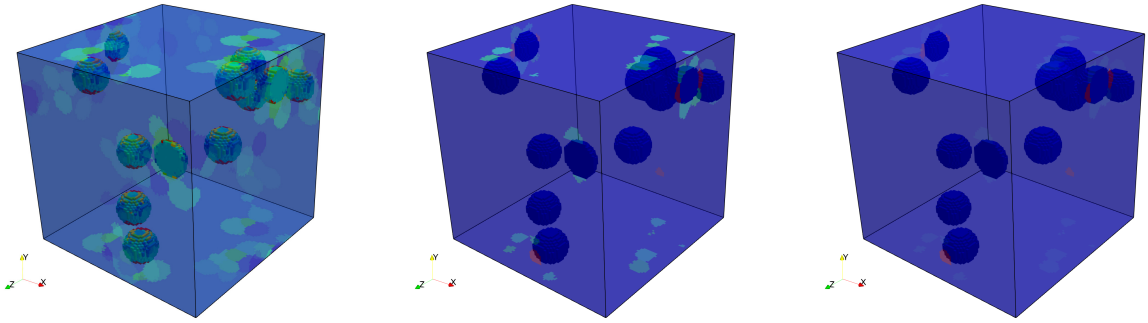


Figure 5.9. Damage evolution in the reduced-order RVEs under HCF loading condition

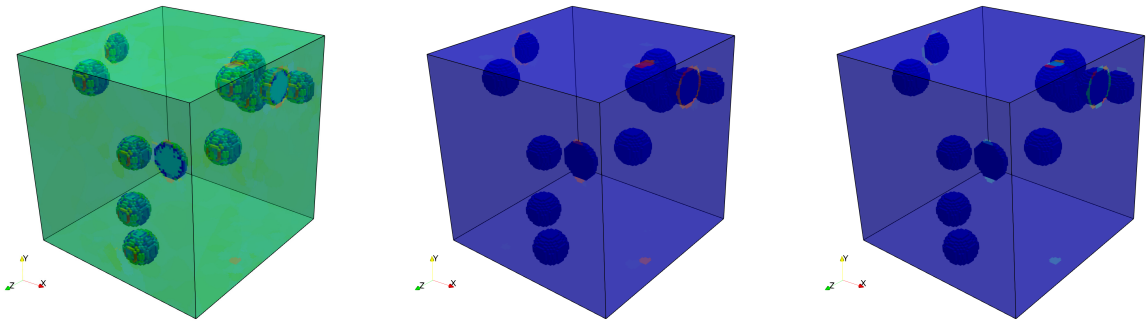


(a) von Mises stress (b) Equivalent plastic strain (c) Damage

Figure 5.10. HCF simulation results obtained by RVE $k = 32$

Results from the finest RVE are demonstrated in Figure 5.11. Since there are more material clusters in this RVE, higher resolution in contour plots and more detailed features can be observed in Figure 5.11 (a) $\sim$ (c).

Finally, we demonstrate the computational performance by measuring the time usage for the first 100 loading cycles. The SCA online prediction stage is written in C++ as a standalone program. The code is compiled by the Intel C++ compiler and linked to the Math Kernel Library (MKL) for solving matrix equations. The testing is done with a single core of Intel Xeon CPU E5-2698 v4. Simulations are performed using the reduced-order RVE with 32 clusters. The original solution algorithm took about 17.1 s to complete the



(a) von Mises stress (b) Equivalent plastic strain (c) Damage

Figure 5.11. HCF simulation results obtained by RVE $k = 128$

simulation of 100 loading cycles. By employing the efficient solution algorithm proposed in this work, the computing time is reduced to only 0.9 s, which is 19 times faster.

5.5 Summary

In this chapter, we briefly reviewed the Self-consistent Clustering Analysis first introduced by (Liu et al., 2016), which is a data-driven reduced-order concurrent multiscale material modeling method. Based on SCA, we established a microstructure-based material modeling method for fatigue applications and developed an efficient solution scheme. Numerical example demonstrated the key features and the unique capability of the proposed method in capturing material microstructural effects, which is critical for HCF applications.

CHAPTER 6

APPLICATIONS ON HIGH CYCLE FATIGUE LIFE PREDICTION

6.1 Introduction

In the previous chapters, we established an efficient multiscale computational framework based on XTFEM and CDM for HCF applications. The numerical implementation features with a hybrid parallel CPUs/GPUs computing on HPC platforms. In this chapter, we will demonstrate 3D HCF applications by using the developed computer program. The program is mainly written in FORTRAN and C/C++. Third partly libraries such as MUMPS, METIS, Intel MKL, CUDA, etc. are integrated with the program. We also developed Python codes for pre- and post-processing by using ABAQUS's user scripting interface. For the examples to be presented in this chapter, the program was compiled by the Intel compilers for FORTRAN and C++ with Intel MPI and the NVIDIA NVCC compiler. The computing platform is TACC's Lonestar 5 supercomputer, which features with 1252 Cray XC40 computing nodes, each with dual 12-core Intel Xeon E5-2690v3 CPUs and 64 GB RAM. It also has 16 computing nodes equipped with NVIDIA K40 GPU and 4 of these GPU nodes can be accessed at a time.

6.2 Single edge notched plate

6.2.1 Problem statement

The geometric dimensions and boundary conditions of the notched plate are illustrated in Figure 6.1. The geometry is designed according to the widely employed single edge notched

---

The following article was reused in this chapter with permission from the publisher:

1. Zhang, R., S. Naboulsi, T. Eason, and D. Qian (2019). A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing. *Finite Elements in Analysis and Design 166*, 103320. Reuse with permission from *Elsevier*.

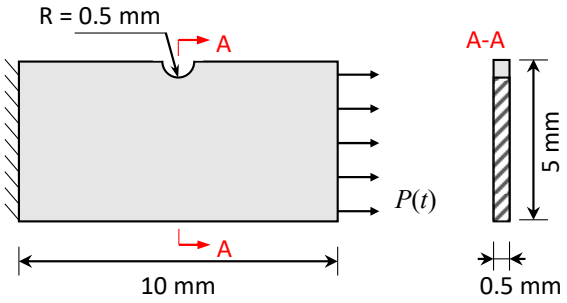


Figure 6.1. Geometric dimensions and boundary conditions of the single edge notched plate tension (SENT) experiment for fatigue studies of metals and alloys (Anderson, 2017). The specimen is fixed at its left end. A cyclic load $P(t)$ is applied to the right end. The material properties are given as Young's modulus $E = 197$ GPa, Poisson's ratio $\nu = 0.3$ and mass density $\rho = 7860$ kg/m^3. Other parameters associated with the two-scale damage model are given in Table 4.6.

6.2.2 Results of HCF simulations

A mesh convergence study is first conducted to determine the spatial discretization. The notched plate is discretized by C3D8 elements using ABAQUS. To reduce the 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 Figure 6.2 (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. Figure 6.2 (b) shows that the maximum von-Mises stress converges to 330 MPa in the case of element size is 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.

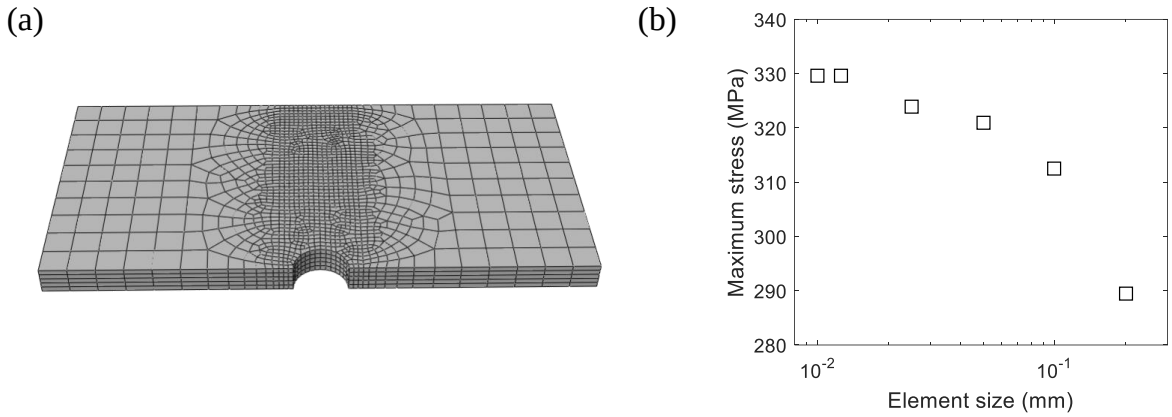


Figure 6.2. Mesh convergence study: (a) a sample spatial mesh with element size = 1 mm and (b) maximum stress versus element size at the notch root

Results of HCF simulation under cyclic load $P(t) = 62.5\sin(40\pi t)$ MPa are shown in Figure 6.3. Fatigue crack initiates at the notch root and propagates to the half width of the specimen. Figure 6.3 (b) shows the microscale nonlinear damage accumulation at the notch root. An exponential crack growth is captured and shown in Figure 6.3 (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 Figure 6.4 in the form of the S-N curve. It is shown that more than 1 *million* cycles are simulated by the proposed framework. To our best knowledge, such 3D direct 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

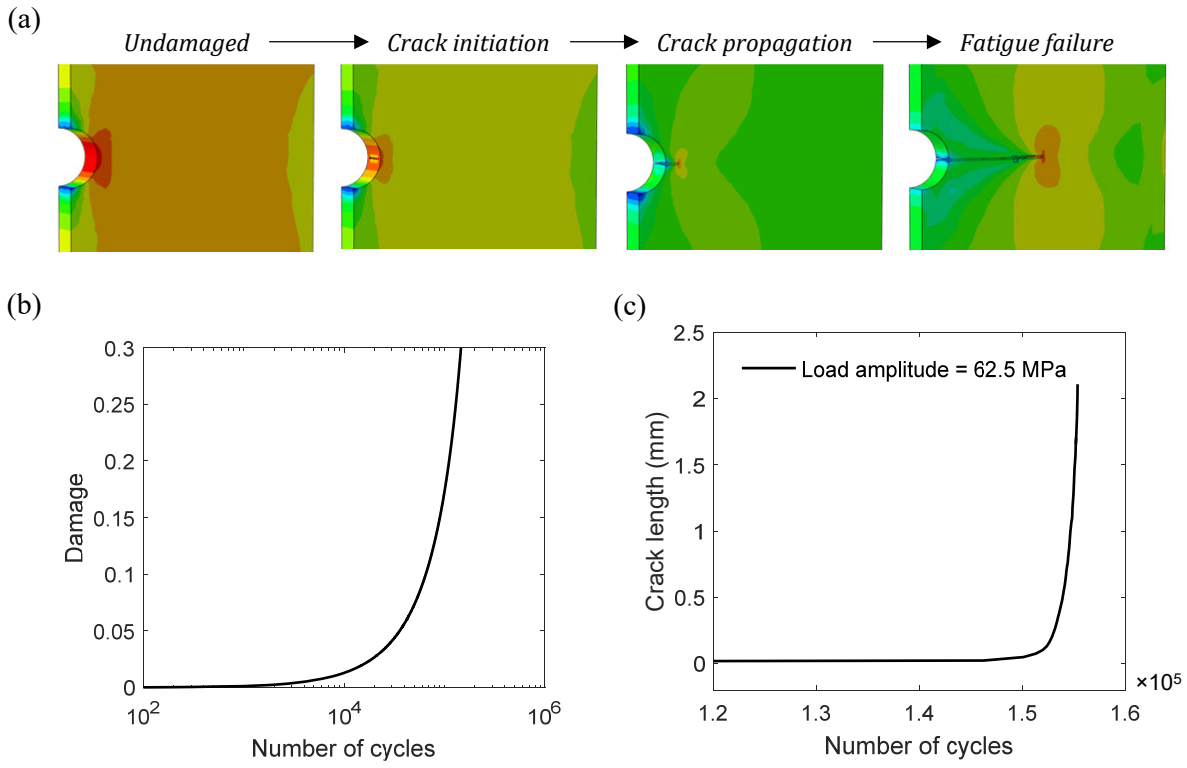


Figure 6.3. Results of HCF simulation: (a) the processes of crack initiation and propagation (colored by von-Mises stress in logarithmic scale), (b) damage accumulation at the notch root, and (c) crack length vs. number of cycles

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 Figure 6.4 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 (Bhamare et al., 2013; Karim et al., 2018).

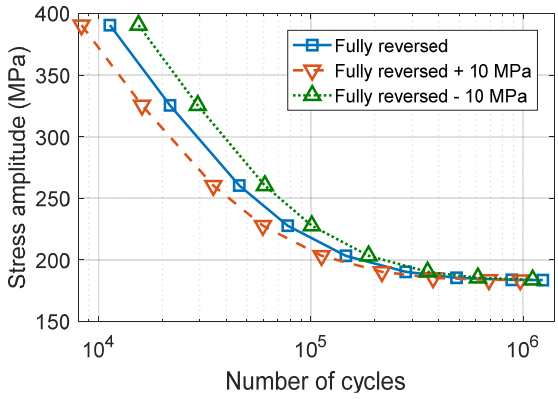


Figure 6.4. S-N curves obtained from series of HCF simulations on the notched specimen

6.2.3 Parallel performance of the two-scale damage model

Figure 6.5 (a) shows the wall-clock time usage by the damage model for 100 loading cycles versus the 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 $\mathcal{O}(N)$, where N is the number of Gauss quadrature points. Parallel efficiency of the OpenMP version damage code is illustrated in Figure 6.5 (b) and shows an optimal speedup.

Similarly, performance of the CUDA version damage code is illustrated in Figure 6.6. The model of GPUs employed is NVIDIA TESLA K40. The maximum number of GPUs employed for testing is 4. In Figure 6.6 (b), the speedup of the CUDA version damage code shows a low performance for the coarse mesh (N = 14,592) when the 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, thereby enabling the large-scale applications.

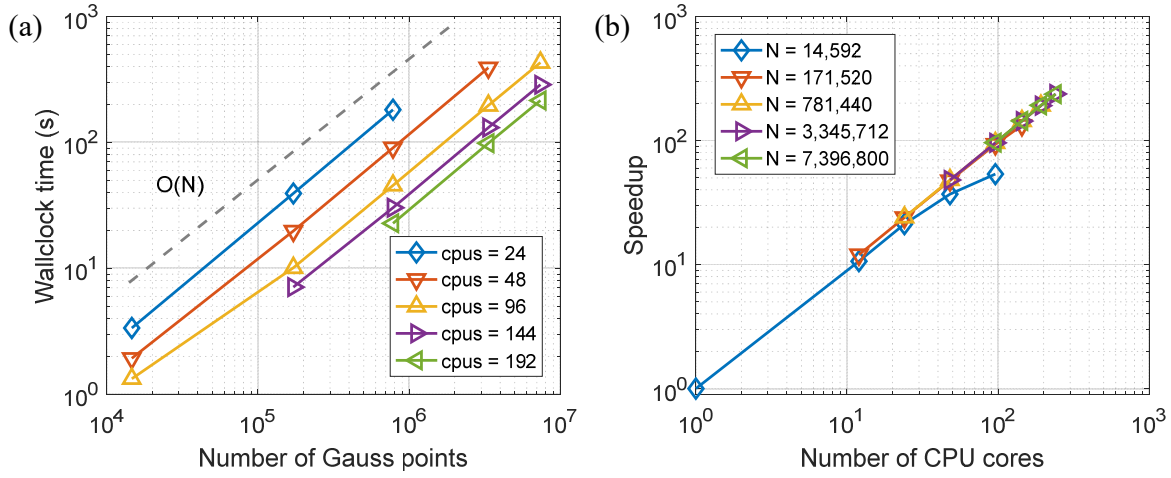


Figure 6.5. Parallel performance of the two-scale damage model on CPUs: (a) wall-clock time vs. number of Gauss points for different number of CPU cores (shown in the inset box), (b) speedup vs. number of CPU cores for different number of Gauss points (shown in the inset box)

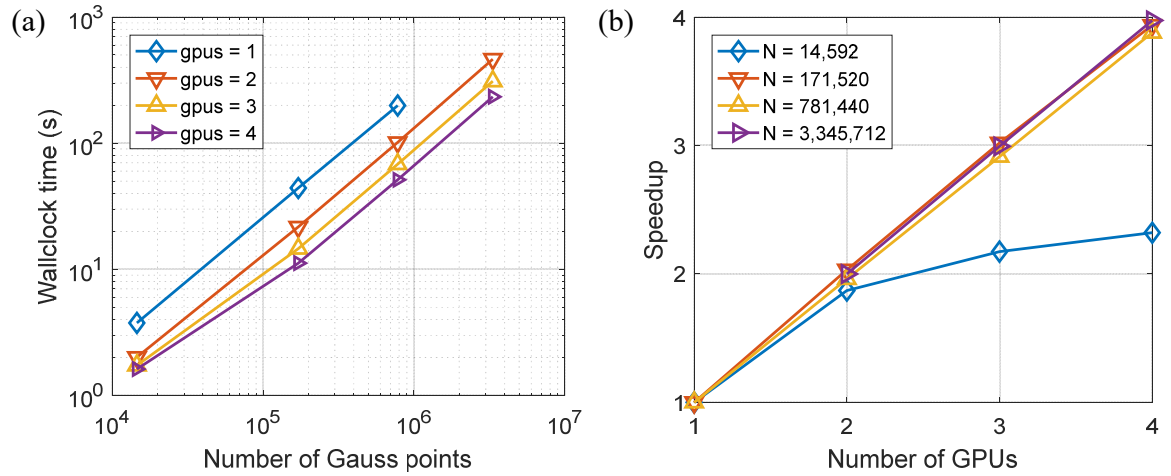


Figure 6.6. Parallel performance of the two-scale damage model on GPUs: (a) wall-clock time vs. number of Gauss points for different number of CPUs (shown in the inset box), (b) speedup vs. number of GPUs for different number of Gauss points (shown in the inset box)

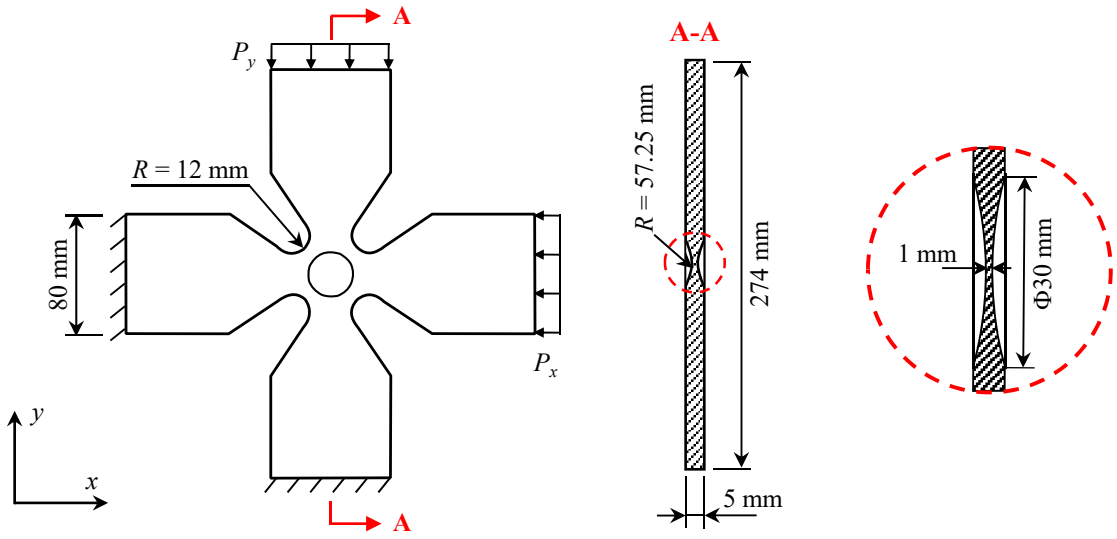


Figure 6.7. Geometry and dimensions of the cross-shaped biaxial HCF specimen

6.3 Biaxially loaded cruciform specimen

6.3.1 Problem statement

Based on the series of biaxial HCF experiments conducted by Poncelet et al. (Poncelet et al., 2010) and Cláudio et al. (Cláudio et al., 2014), biaxial HCF simulations are performed to further demonstrate the capability of the developed framework on handling complex multi-axial loading conditions. The biaxial specimen considered here is adopted from (Poncelet et al., 2010). Geometry and dimensions of the specimen are provided in Figure 6.7. 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 the x and the 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,

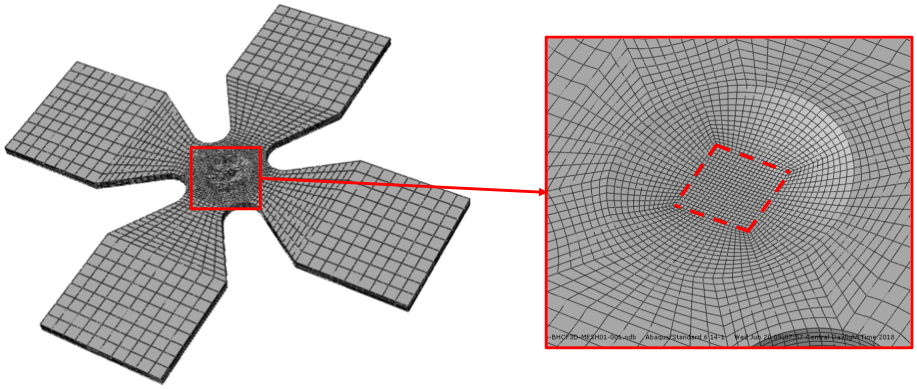


Figure 6.8. Spatial discretization of the biaxial specimen, dashed box indicates the gauge zone

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 Figure 6.8. 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.

Figure 6.9 shows that the maximum von-Mises stress converges when the element size is less than 0.5 mm in the gauge zone. The relative error of stresses between the coarsest and the finest discretizations is only 0.3% due to the higher-order element formulation.

Furthermore, a comparison between the stress distributions obtained from ABAQUS and XTFEM is shown in Figure 6.10. Those two solutions are almost identical. The relative error of maximum von-Mises stress is less than 1.0×10^{-5}. Therefore, the element size of 0.5 mm at the gauge zone is 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 crack.

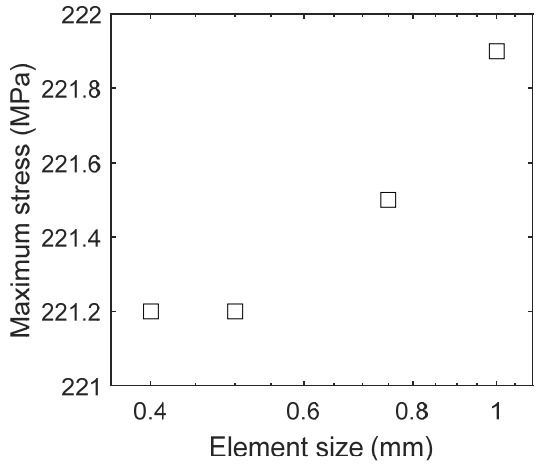


Figure 6.9. Mesh convergence for the biaxial HCF specimen

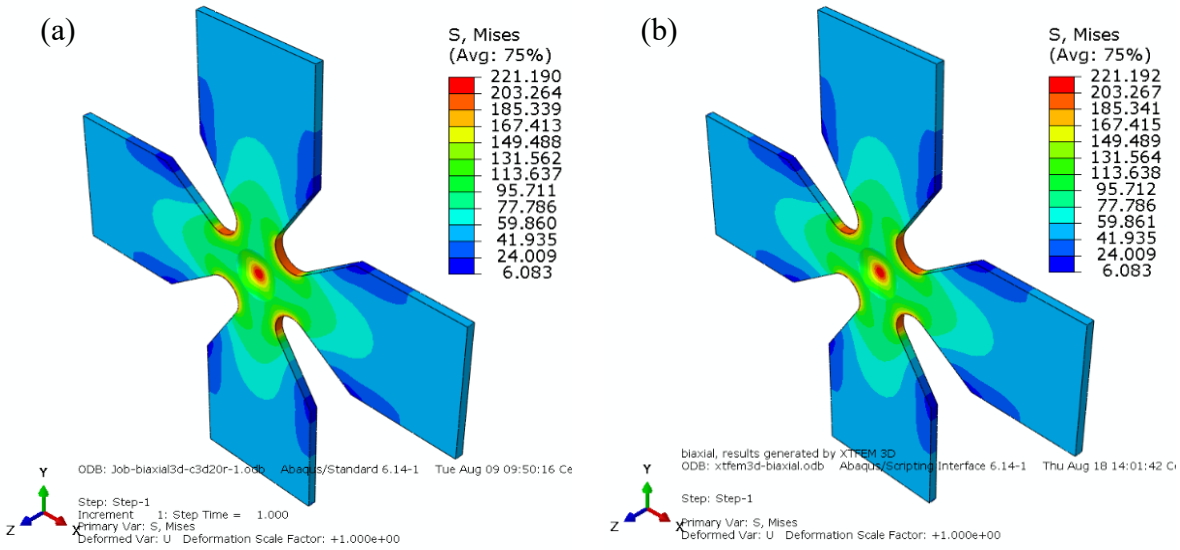


Figure 6.10. Comparison of stress distributions obtained from (a) ABAQUS and (b) XTFEM

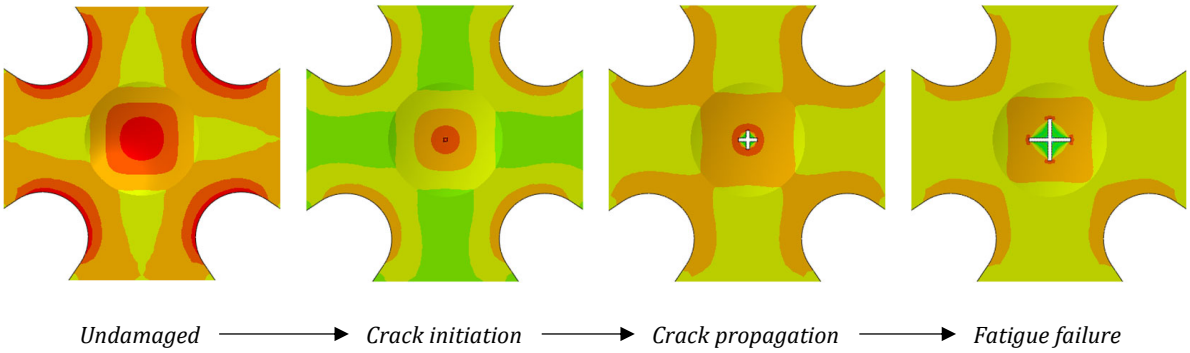


Figure 6.11. Crack initiation and propagation of the biaxial HCF specimen (colored by von-Mises stress in logarithmic scale)

6.3.2 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 propagate along both the x and the y directions to the critical crack length (the size of the gauge zone) at 98,420 cycles. The process of crack initiation and propagation is illustrated in Figure 6.11. Crack growth data obtained from the simulation is plotted in Figure 6.12 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 Figure 6.11.

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 the x and the 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

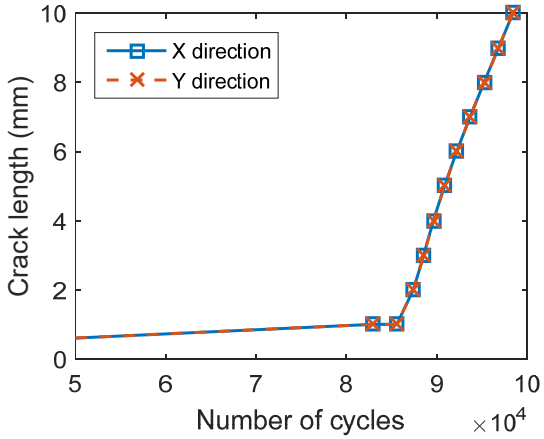


Figure 6.12. Cracks growth of the biaxial specimen

to symmetry in x-y plane. Results of this group of biaxial HCF simulations are presented in Figure 6.13, 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 Figure 6.13 (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 Figure 6.13 (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 Figure 6.13 (c) clearly demonstrates such complex interactions between P_x and P_y. A similar result is reported in Fig.6.12 in (Lemaitre and Desmorat, 2005).

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 is applied in the y direction with amplitude varying from 0 MPa to 30 MPa with an interval of 5 MPa. Figure 6.14 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.

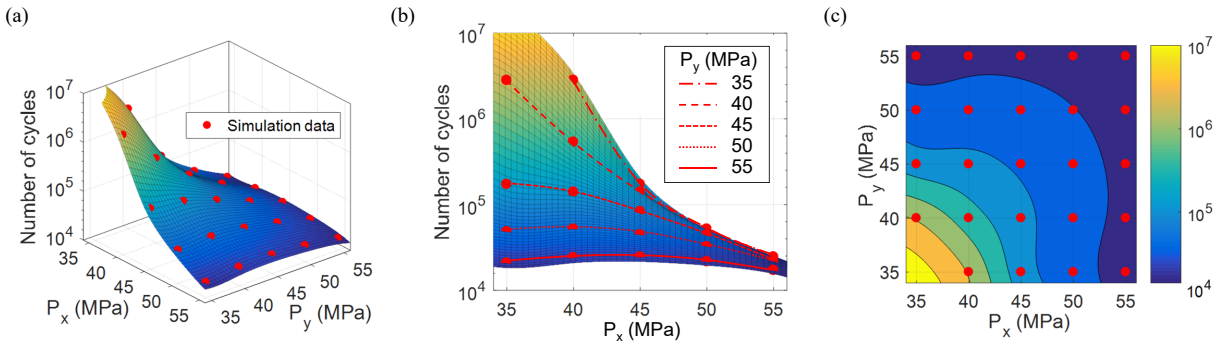


Figure 6.13. Biaxial fatigue life as function of stress amplitudes in both the x and the y directions: (a) the 3D plot, (b) the projected 2D view of the 3D plot, and (c) the 2D contour plot (dots are simulation results; the trend surface is obtained from a curve fitting)

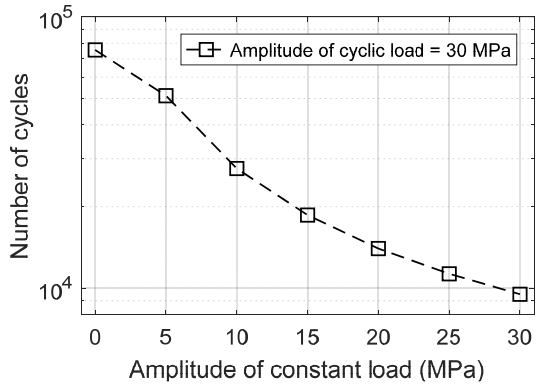


Figure 6.14. Results of biaxial HCF simulations conducted under cyclic loading along the x direction and constant loading along the y direction

6.3.3 Computational performance

For different number of unknowns, the wall-clock time usages for three of the most computationally intensive parts are summarized in Table 6.1. Note that the time usage of the damage model is based on the OpenMP version with 24 CPU cores per compute node. Table 5.1 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 $7 \sim 35$ times of the hybrid solver depending on problem size.

Table 6.1. 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

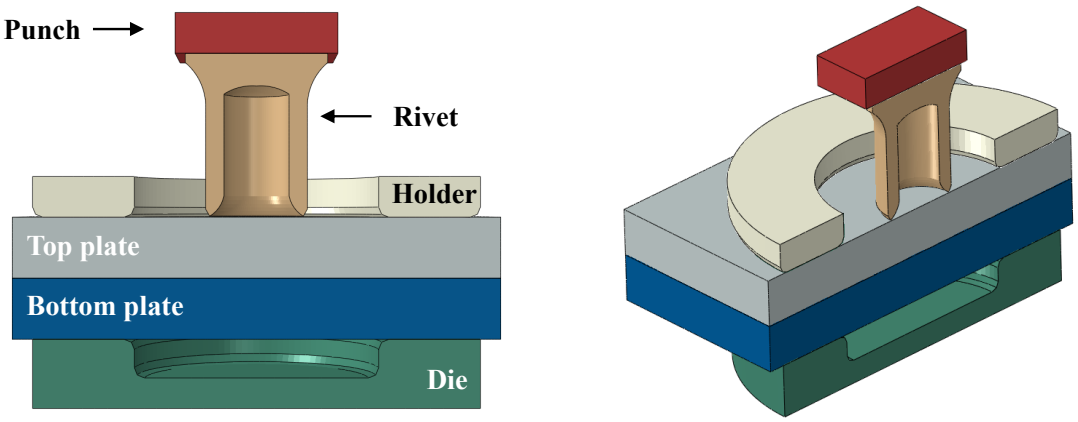
Note: [a] With 4 MPI processes per node and 6 OpenMP threads per process

6.4 Self-piercing riveted joint

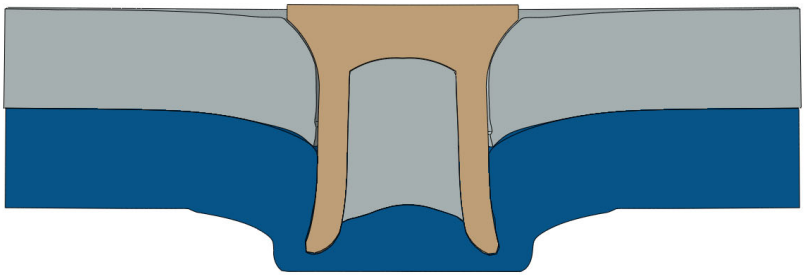
6.4.1 Problem statement

Self-piercing riveting (SPR) has been widely employed in the automotive industry as a cold mechanical joining process to reduce the weight of vehicle structures and achieve a higher fuel efficiency (Li et al., 2017). Unlike traditional sheet metal joining processes, SPR does not require pre-drilled/punched holes and is environmentally friendly, i.e. no spark, no fume, and low noise. The SPR process is fast (typically $1 \sim 4$ s) and easy for automation. In addition to the many advantages of SPR, it is also capable of producing joints with high fatigue strength (Iyer et al., 2005; Sun et al., 2007; Chung and Kim, 2016). Robust numerical simulations are helpful for automotive designers to better understand the mechanical behaviors of SPR joints, especially with a multiscale approach (Gao, 2020). In this example, we present a preliminary study on HCF life prediction for a SPR joint based on the proposed multiscale method[1]. The long-term goal is to establish a systematic mechanistically meaningful concurrent multiscale numerical framework to provide a comprehensive understanding of the fatigue mechanism and the role of microstructure in fatigue life of SPR joints as well as other engineering structures and components.

[1]This work is based on a collaborative project with Professor Wing Kam Liu's group at Northwestern University. The SPR process simulation is done by Jiaying Gao, Derick Suarez, and Sourav Saha (Northwestern). Yingjian Liu (UT Dallas) calibrated material parameters and prepared finite element meshes for HCF simulation. Their contributions are gratefully acknowledged.



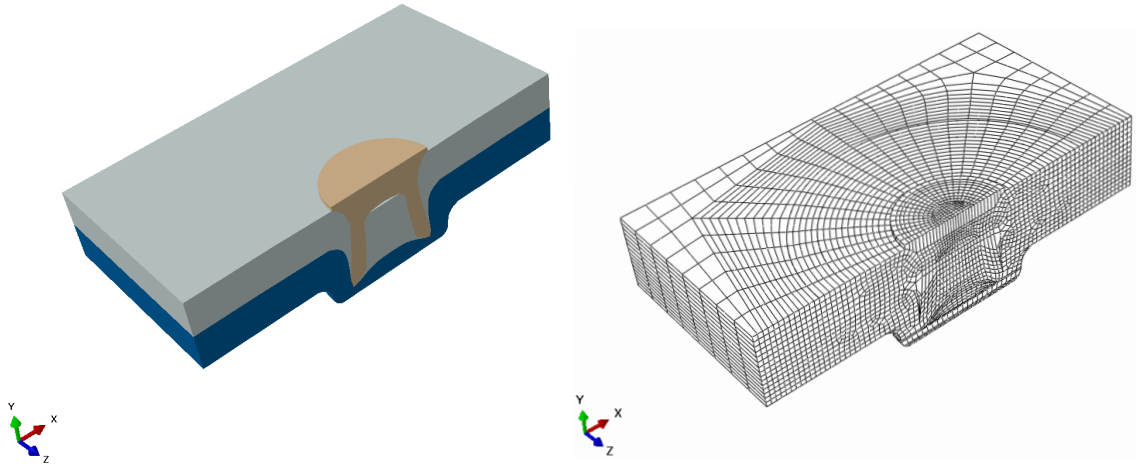
(a) Initial configuration



(b) Final configuration

Figure 6.15. SPR process simulation

In this example, the riveted joint is obtained by a SPR process simulation using the concurrent multiscale self-consistent clustering analysis (Gao, 2020). Figure 6.15 (a) illustrates the initial configuration of the SPR process simulation. In this model, two aluminum (AA6060-T4) sheets are held together by a rigid holder on the top and a rigid die in the bottom. The steel rivet and the rigid punch are initially placed at the positions shown in Figure 6.15 (a). The SPR process is simulated in three steps, i.e. punching, holding, and relaxing. The final configuration is shown in Figure 6.15 (b), which serves as the starting point of the multiscale HCF simulation. More details of the SPR process simulation can be found in (Gao, 2020).



(a) Geometry (b) Mesh

Figure 6.16. Reconstructed SPR joint model for HCF simulation

6.4.2 HCF simulation model

The geometry model employed in HCF simulation is reconstructed based on the final configuration of the SPR process simulation. The main reason for not directly using the deformed mesh from the SPR process simulation is that many elements are distorted in the final configuration and the mesh quality is not good enough for HCF simulation. The reconstructed geometry model is shown in Figure 6.16 (a). Note that a half model is shown here to visualize the cross section. The HCF model has three parts: two pierced aluminum sheets and a flared steel rivet, which in together formed an interlock in the mechanical joint. Since the current XTFEM code lacks the capability of handling contacts, the aluminum plates and the steel rivet are merged together before meshing. In other words, perfect bonding is assumed between these parts to approximate the small sliding contact formulation under HCF loading conditions (Simulia, 2014a). Similar to previous HCF simulations, we created a high-quality gradient mesh with the quadratic 20-node brick elements with reduced integration (C3D20R), which is illustrated in Figure 6.16 (b). In this mesh, there are 36,864 elements and 156,485 nodes, which lead to 294,912 material points and 2,816,730 space-time DOFs, respectively.

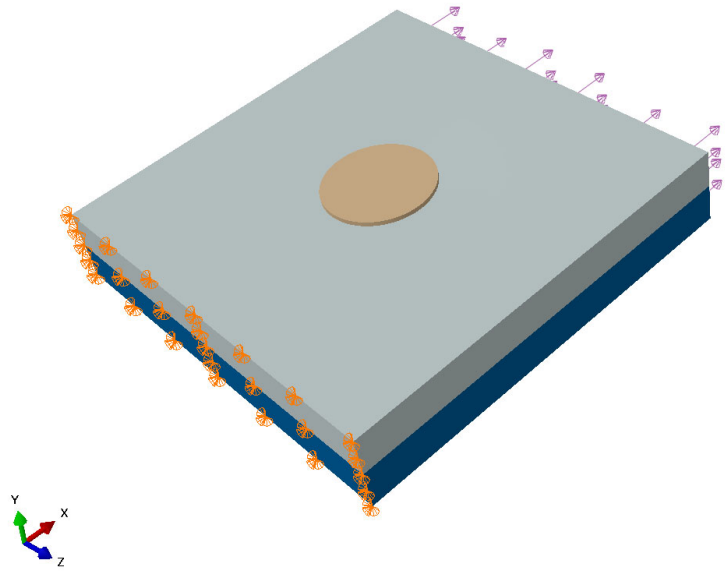
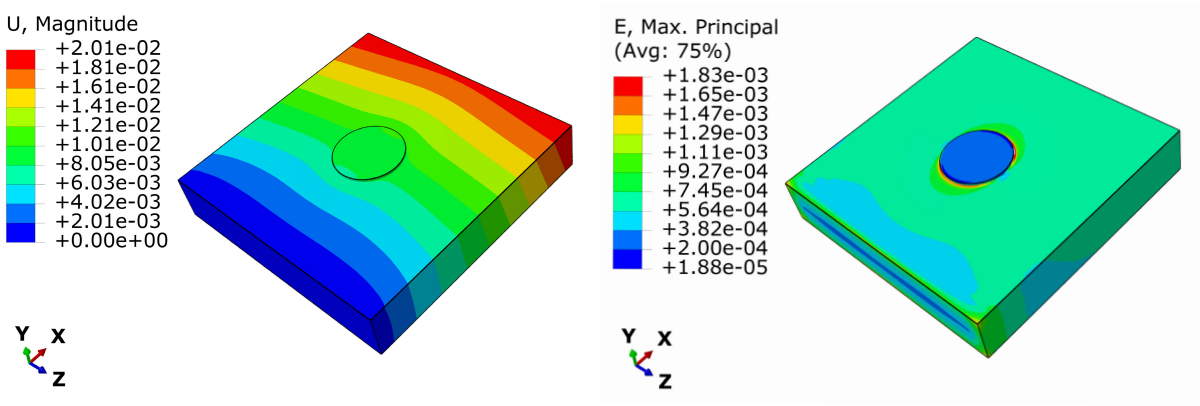


Figure 6.17. SPR joint HCF simulation: boundary conditions

In the HCF simulation, the steel rivet is modeled by the isotropic linear elastic material model without fatigue damage. The volumetic mass density, Young's modulus and Poisson's ratio of the steel are 7860 kg/m^3, 210 GPa and 0.3, respectively. The aluminum plates are modeled by the two-scale damage model with the following parameters: $\rho = 2700$ kg/m^3, $E = 65.5$ GPa, $\nu = 0.33$, $C_y = 1090$ MPa, $\sigma_u = 175$ MPa, $\varepsilon_{pD} = 0.13$, $\sigma_f^\infty = 60$ MPa, $S = 0.05$, and $s = 29.4$. These material parameters are calibrated by using experimental data in (Timmermann et al., 2014; Wagener and Melz, 2018).

HCF boundary conditions of the riveted joint are illustrated in Figure 6.17. Both plates are fixed at the surfaces where $X = X_{\min}$, while the opposite surfaces at $X = X_{\max}$ are subjected to a fully-reversed cyclic traction in the x direction with a constant amplitude of 40 MPa and a loading frequency of 20 Hz. Similar to the previous examples, initial temporal size of space-time slab is $100T$, where $T = 0.05$ s is the period of the loading cycle. The time step size is reduced to $10T$ after crack initiation. XTFEM solution for each space-time slab is further interpolated into 200 temporal increments per cycle for solving the two-scale damage model on material points.



(a) Displacement magnitude (b) Maximum principal strain

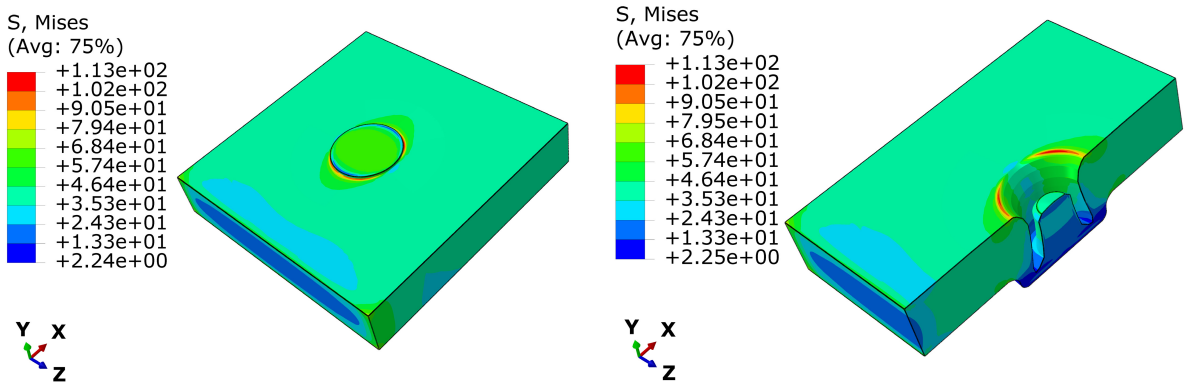
Figure 6.18. SPR joint HCF simulation: displacement and strain at macroscale

6.4.3 HCF simulation results

The macroscopic displacement magnitude and the maximum principal strain distributions in the mechanical joint obtained from the HCF simulation are illustrated in Figure 6.18 (a) and (b), respectively. Note that due to the cyclic nature of fatigue loading, these results are obtained by interpolating the XTFEM solution such that dynamic mechanical responses under the maximum surface traction are captured. Figure 6.18 (b) shows that a high elastic strain gradient is observed at the top aluminum sheet near the hole punched by the steel rivet.

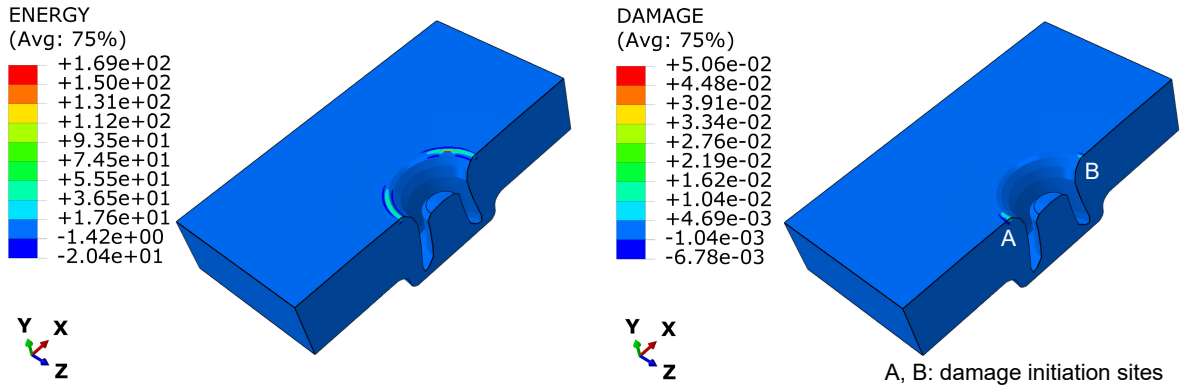
Next, the macroscopic von Mises equivalent stress distribution under the HCF loading condition is shown in Figure 6.19. The stress distribution is captured in the same manner as the displacement/strain distributions. Figure 6.19 (b), i.e. the half model without the rivet, also shows that the maximum stress occurs at the top plate surrounding the rivet. The stress concentration due to the riveting process creates potential fatigue crack initiation sites under HCF loading condition.

At the microscale, Figure 6.20 shows the stored energy density (defined in Eq. (4.15)) and damage distributions after 45,000 loading cycles. Similarly, only a half model without the rivet is shown here for a better illustration. Recall that in the two scale damage model,



(a) Full model (b) Half model without the rivet

Figure 6.19. SPR joint HCF simulation: macroscopic stress distribution



(a) Stored energy density (b) Damage

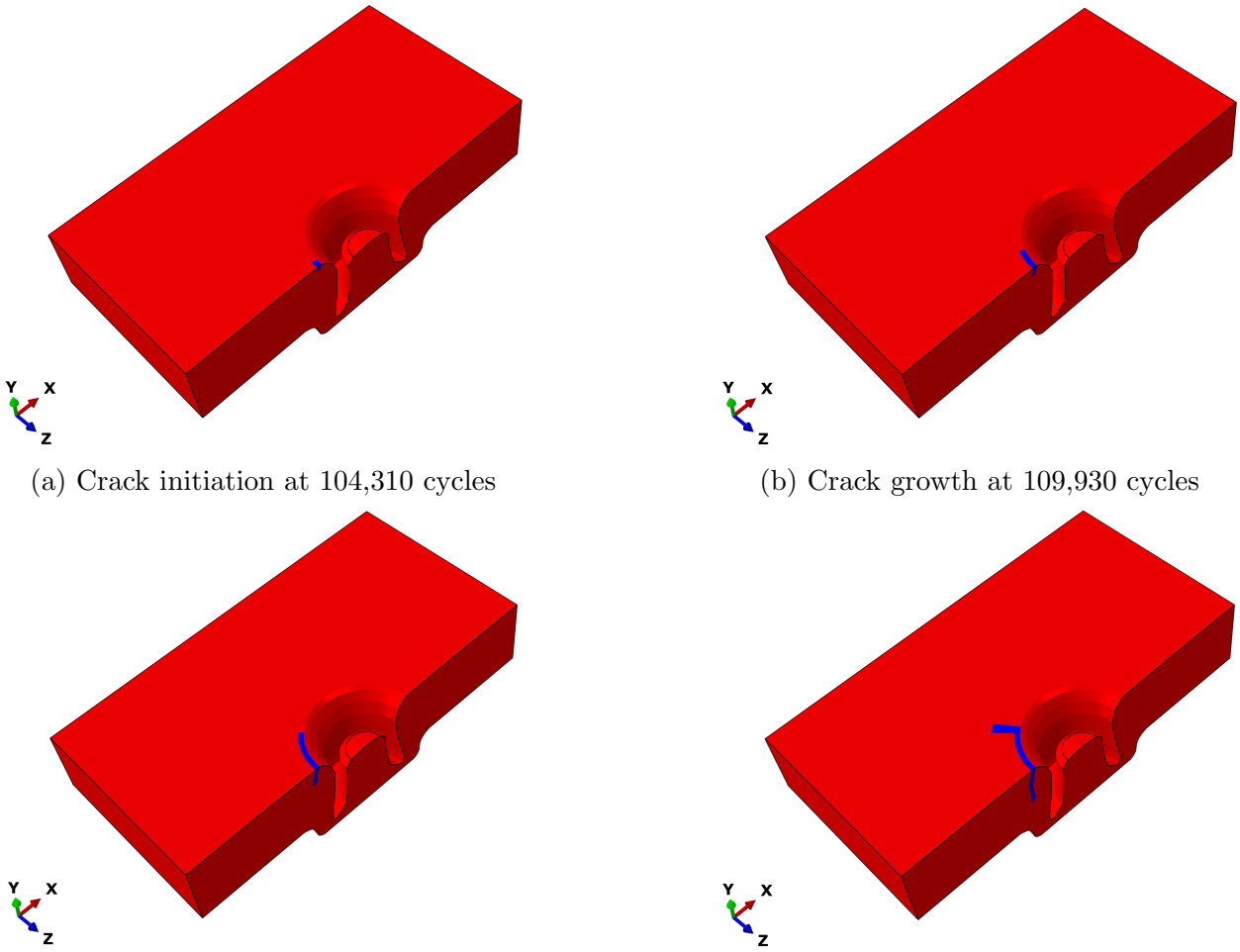
Figure 6.20. SPR joint HCF simulation: stored energy density and damage distributions at microscale after 45,000 loading cycles

damage initiates when the stored energy density is greater than the energetic damage threshold value (defined in Eq. (4.16)). In Figure 6.20 (b), two damage initiation sites are observed and marked by points **A** and **B**, respectively. It shows that the damage is highly localized due to the stress concentration around the punched hole in the top plate. Note that the fatigue crack initiates when the damage reaches the critical value $D_c = 0.3$.

Finally, the fatigue crack initiation and propagation are illustrated in Figure 6.21 (a) $\sim$ (d). As shown in Figure 6.21 (a), the crack initiates at 104,310 loading cycles at the previously

observed damage initiation site **A**. Figure 6.21 (b) and (c) show that the crack propagates along two directions. The first one is approximately along the z direction, which is perpendicular to the loading direction (x). The crack also propagates toward the thickness direction of the plates, i.e. the $-y$ direction. As illustrated in Figure 6.21 (d), the final fatigue crack path is closer to the fully fixed side of the model, which shows the influence of boundary conditions. It is worth noting that the lap-shear loading condition, which is widely employed in experiments (Iyer et al., 2005; Sun et al., 2007; Chung and Kim, 2016), can be simply simulated by extending the top and bottom plates at opposite directions in the modeling stage and applying proper boundary conditions. For this HCF simulation, the fatigue fracture happens at 124,120 cycles when the crack propagates through the thickness direction of the top aluminum plate, which is shown in Figure 6.21 (d). It shows that most fatigue life is consumed by the crack initiation stage under HCF loading condition. Although a quantitative comparison with experiment results is not available for this particular model, the same failure mode is reported in experiment observations for similar sample geometries and testing configurations (Iyer et al., 2005; Sun et al., 2007).

As a brief summary of this example, we have further demonstrated the capability of the proposed multiscale framework in direct simulating HCF failures in real engineering structures/components. We would like to note that this example presents only preliminary results for the purpose of demonstration. For future work, there are many aspects to explore. For example, we can incorporate the residual stress and/or plastic strain fields from the SPR process simulation to the HCF simulations. Also, SCA fatigue damage model proposed in Chapter 5 can be integrated with XTFEM to account for microstructural effects on fatigue life. Additionally, contact models would be helpful to further improve the accuracy in SPR HCF simulation. Finally, more complex loading conditions and histories can be directly simulated by the proposed method.



(a) Crack initiation at 104,310 cycles

(b) Crack growth at 109,930 cycles

(c) Crack growth at 115,150 cycles

(d) Fatigue fracture at 124,120 cycles

Figure 6.21. Fatigue crack initiation and propagation in SPR joint under HCF loading (red color indicates intact elements while blue color represents failed elements)

6.5 Summary

In this chapter, we have carried out series of HCF simulations on the single edge notched specimen, the cruciform biaxial specimen, and the self-piercing riveted mechanical joint. Mean stress effects, multiaxial loading interactions, and complex geometries with multiple mateirals are directly simulated by using the proposed multiscale space-time approach. These examples clearly demonstrated the capabilities of the proposed framework on handling large 3D problems and complex fatigue loading conditions. To our best knowledge, this is the first

time that such direct numerical HCF simulations are performed for large 3D applications with complex geometries and loading conditions in literature as opposed to the traditional fatigue modeling approaches and the *jump-in-cycle* simulation approach.

PART II

A CONCURRENT MULTISCALE METHOD TO DYNAMIC FRACTURE

CHAPTER 7

PERIDYNAMICS

7.1 Introduction

Peridynamics (PD) is first introduced by Silling (Silling, 2000) to model spatial discontinuities, e.g. cracks, in a consistent continuum solid mechanics framework without special treatments. It is a nonlocal generalization of *classical continuum mechanics* (CCM) in which material points interact directly with each other by long-range forces over finite distances. In this chapter, we start by introducing both the bond-based and state-based PD formulations. PD numerical implementation is developed with parallel computing acceleration and algorithm optimizations. Finally, numerical examples are presented to demonstrate the unique capability of PD in terms of handling discontinuities, especially for spontaneous crack initiations and propagations.

7.2 Theory of Peridynamics

In PD, the force of a material point $\boldsymbol{x}'$ in a body $\mathcal{B}$ exerted on another material point $\boldsymbol{x}$ is characterized by a force density function $\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t)$, of which the unit is force/volume^2. To ensure conservation of linear momentum, the force density $\boldsymbol{f}$ must satisfy

$$\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t) = -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{x}', t) \tag{7.1}$$

The relative position vector between these points in the reference configuration, $\boldsymbol{\xi}$, is termed as a *bond* and given by

$$\boldsymbol{\xi} = \boldsymbol{x}' - \boldsymbol{x} \tag{7.2}$$

and the relative displacement $\boldsymbol{\eta}$ is given by

$$\boldsymbol{\eta} = \boldsymbol{u}(\boldsymbol{x}', t) - \boldsymbol{u}(\boldsymbol{x}, t) \tag{7.3}$$

so that $\boldsymbol{\xi} + \boldsymbol{\eta}$ gives the relative position in the current configuration.

Conservation of angular momentum requires that

$$\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t) \times (\boldsymbol{\xi} + \boldsymbol{\eta}) = \boldsymbol{0} \tag{7.4}$$

Based on the above definitions, the PD equation of motion at a material point $\boldsymbol{x}$ in the reference configuration at time t is given by

$$\rho(\boldsymbol{x})\ddot{\boldsymbol{u}}(\boldsymbol{x}, t) = \int_{\mathcal{B}} \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t) dV_{\boldsymbol{x}'} + \boldsymbol{b}(\boldsymbol{x}, t) \tag{7.5}$$

where $\rho(\boldsymbol{x})$ is the volumetric mass density, $\boldsymbol{u}(\boldsymbol{x}, t)$ is the displacement, $\boldsymbol{b}(\boldsymbol{x}, t)$ is the prescribed body force density, superimposed dot indicates partial derivative with respect to time.

It is further assumed in PD that material points interact with each other over a finite distance δ such that

$$\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t) = \boldsymbol{0} \qquad \forall\, |\boldsymbol{\xi}| > \delta \tag{7.6}$$

The distance δ is termed as *horizon radius* or simply *horizon*, which is considered as a material parameter, i.e. an intrinsic length of the particular material, in PD. A discussion on PD horizon can be found in (Bobaru and Hu, 2012). The *family* of a material point $\boldsymbol{x}$ is the collection of all other material points within its horizon, which is denoted by $\mathcal{H}_{\boldsymbol{x}}$. The PD equation of motion can be rewritten as follows:

$$\rho(\boldsymbol{x})\ddot{\boldsymbol{u}}(\boldsymbol{x}, t) = \int_{\mathcal{H}_{\boldsymbol{x}}} \boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t) dV_{\boldsymbol{x}'} + \boldsymbol{b}(\boldsymbol{x}, t) \tag{7.7}$$

A substantial distinction between PD and CCM can be clearly seen from Eq. (7.7). The PD internal force term is evaluated by integration of the force density over a finite subdomain, i.e.

$$\boldsymbol{f}_{\text{PD}}^{\text{int}} = \int_{\mathcal{H}} \boldsymbol{f} dV \tag{7.8}$$

which is free of spatial derivatives thus valid regardless of the presence of spatial discontinuities.

In contrast, in CCM, the internal force term is given by

$$\boldsymbol{f}_{\mathrm{CCM}}^{\mathrm{int}} = \nabla \cdot \boldsymbol{\sigma}\left(\nabla \boldsymbol{u}\right) \tag{7.9}$$

in which partial derivatives on displacement field with respect to spatial coordinates are not well defined mathematically on spatial discontinuities.

The force density function $\boldsymbol{f}$ characterizes the interactions between material points and contains all material behavior information. Based on choice of the force density function, PD formulations can be classified into two main categories: (1) bond-based and (2) state-based, which will be introduced as follows.

7.2.1 Bond-based materials

In bond-based materials, the force density function $\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t)$ only directly depends on the bond between these two material points, i.e.

$$\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{\xi}, \boldsymbol{\eta}, t) \tag{7.10}$$

Conservation laws of linear and angular momentums are rewritten as

$$\boldsymbol{f}(\boldsymbol{\xi}, \boldsymbol{\eta}) = -\boldsymbol{f}(-\boldsymbol{\xi}, -\boldsymbol{\eta}) \tag{7.11}$$

and

$$\boldsymbol{f}(\boldsymbol{\xi}, \boldsymbol{\eta}) \times (\boldsymbol{\xi} + \boldsymbol{\eta}) = \boldsymbol{0} \tag{7.12}$$

Note that we omitted the dependency on time for simplicity.

Followed by these conservation laws, the force density function can be expressed as

$$\boldsymbol{f}(\boldsymbol{\xi}, \boldsymbol{\eta}) = f(\boldsymbol{\xi}, \boldsymbol{\eta}) \boldsymbol{M}(\boldsymbol{\xi}, \boldsymbol{\eta}) \tag{7.13}$$

where $\boldsymbol{M}(\boldsymbol{\xi}, \boldsymbol{\eta})$ is a unit vector given by

$$\boldsymbol{M} = \frac{\boldsymbol{\xi} + \boldsymbol{\eta}}{|\boldsymbol{\xi} + \boldsymbol{\eta}|} \tag{7.14}$$

and $f(\boldsymbol{\xi}, \boldsymbol{\eta})$ is a scalar-valued function which satisfies

$$f(\boldsymbol{\xi}, \boldsymbol{\eta}) = -f(-\boldsymbol{\xi}, -\boldsymbol{\eta}) \tag{7.15}$$

The interactions between bond-based material can be conceptually understood as springs that connect the material points. Mechanical behaviors of these springs, i.e. $f(\boldsymbol{\xi}, \boldsymbol{\eta})$, can be either linear or nonlinear, and material failure can be simply represented by irreversible breakage of these springs. One of the simplest bond-based materials is called *Prototype Micro-elastic* (PM) material. The spring function in PM is given by

$$f(\boldsymbol{\xi}, \boldsymbol{\eta}) = c\frac{e}{|\boldsymbol{\xi}|} \tag{7.16}$$

where c is called *spring constant* and e is the *bond extension* defined by

$$e = |\boldsymbol{\xi} + \boldsymbol{\eta}| + |\boldsymbol{\xi}| \tag{7.17}$$

We further define *bond strain* by

$$s = \frac{e}{|\boldsymbol{\xi}|} \tag{7.18}$$

such that the spring function is simplified as

$$f(\boldsymbol{\xi}, \boldsymbol{\eta}) = cs \tag{7.19}$$

For isotropic linear elastic materials, the spring constant c can be calibrated by matching the spring potential energy with the CCM strain energy at an arbitrary interior material point $x \in \mathcal{B}$ that is far from any boundaries. For the detailed calibration procedures please refer to (Madenci and Oterkus, 2014; Bobaru et al., 2016). After the calibration, the spring

constant c is expressed in terms of the bulk elastic properties of a material, which is given by

$$c = \begin{cases} \frac{2E}{A\delta^2} & \text{for 1D} \\ \frac{12K'}{\pi h \delta^3} & \text{for 2D} \\ \frac{18K}{\pi \delta^4} & \text{for 3D} \end{cases} \tag{7.20}$$

where E and K are Young's modulus and bulk modulus, respectively. K' is the 2D bulk modulus given by

$$K' = \begin{cases} \frac{E}{2(1-\nu)} & \text{plane stress} \\ \frac{E}{2(1-2\nu)(1+\nu)} & \text{plane strain} \end{cases} \tag{7.21}$$

in which ν is Poisson's ratio.

In Eq. (7.20), the geometric shapes of bulk material for calibration are line segment, circle, and sphere for 1D, 2D, and 3D cases, respectively. A is the cross-sectional area in 1D, h is the thickness in 2D. Detailed derivations of those constants can be found in (Madenci and Oterkus, 2014).

PM material can be extended to account for brittle facture, which leads to the *Prototype Micro-elastic Brittle* (PMB) material. The corresponding spring function is given by

$$f\left(\boldsymbol{\xi}, \boldsymbol{\eta}\right) = \begin{cases} cs & s \leqslant s_c \\ 0 & s > s_c \end{cases} \tag{7.22}$$

where s_c is the critical strain. Note that the bond breakage is irreversible.

The critical strain can also be calibrated using a similar method (Bobaru et al., 2016) and is given by

$$s_c = \begin{cases} \sqrt{\frac{3G_0}{E\delta}} & \text{for 1D} \\ \sqrt{\frac{\pi G_0}{3K'\delta}} & \text{for 2D} \\ \sqrt{\frac{5G_0}{9K\delta}} & \text{for 3D} \end{cases} \tag{7.23}$$

where G_0 is the fracture energy per unit area.

There are several drawbacks of the bond-based material preventing its practical applications (Silling et al., 2007). For isotropic linear elastic materials, the bond-based model leads to a fixed Poisson's ratio of 1/4 (for 3D), which is due to an oversimplification to assume that each bond connected to a material point responds independently of all the other bonds. In fact, there is only one independent elastic constant in isotropic linear elastic PM material as shown in Eq. (7.20). Other difficulties associated with bond-based material are incompatibility with material constitutive laws in CCM and issues on modeling metal plasticity, etc.

7.2.2 State-based materials

The above shortcomings of the bond-based PD motivate the development of the state-based PD, which is first introduced by Silling et al. in (Silling et al., 2007). The PD *states* and related mathematical operations are introduced in Appendix D. In the state-based PD, the force density between material points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is represented by

$$\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t) = \boldsymbol{t}(\boldsymbol{x}', \boldsymbol{x}, t) - \boldsymbol{t}(\boldsymbol{x}, \boldsymbol{x}', t) \tag{7.24}$$

where $\boldsymbol{t}$ is the bond force density vector given by

$$\boldsymbol{t}(\boldsymbol{x}', \boldsymbol{x}, t) = \underline{\boldsymbol{T}}\left[\boldsymbol{x}, t\right] \langle \boldsymbol{\xi} \rangle \tag{7.25}$$

in which $\underline{\boldsymbol{T}}\left[\boldsymbol{x}, t\right]$ is the *force density vector state* at material point $\boldsymbol{x}$ and associates with each bond $\boldsymbol{\xi}$ in the family of $\boldsymbol{x}$.

The state-based force density between material points $\boldsymbol{x}$ and $\boldsymbol{x}'$ depends not only on the particular bond $\boldsymbol{\xi}$ but also other bonds connected to these material points collectively. It can also be shown that Eq. (7.24) satisfies the conservation law of linear momentum:

$$\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t) = \boldsymbol{t}(\boldsymbol{x}', \boldsymbol{x}, t) - \boldsymbol{t}(\boldsymbol{x}, \boldsymbol{x}', t) = -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{x}', t) \tag{7.26}$$

7.2.2.1 Ordinary state-based PD

In the ordinary state-based PD, the force density vector state is parallel to the deformed bond. Therefore,

$$\underline{\boldsymbol{T}} = \underline{t}\boldsymbol{M} \tag{7.27}$$

in which $\underline{t}$ is the *scalar force density state*, $\boldsymbol{M}$ is a unit vector defined in Eq. (7.14). It can be shown that Eq. (7.27) automatically satisfies the conservation law of angular momentum.

An example of simple ordinary state-based material is called *Linear Peridynamic Solid* (LPS) material model. The scalar force density state of LPS is given by

$$\underline{t} = \frac{3k\theta}{m}\underline{\omega x} + \alpha\underline{\omega e}^{\mathrm{dev}} \tag{7.28}$$

where k and α are positive constants. Other variables in the above equation are given as follows.

The *reference position vector state* is defined by

$$\underline{\boldsymbol{X}}\langle\boldsymbol{\xi}\rangle = \boldsymbol{\xi} \tag{7.29}$$

and the corresponding scalar state is obtained by

$$\underline{x} = |\underline{\boldsymbol{X}}| \tag{7.30}$$

Similarly, the *deformation vector state* is defined by

$$\underline{\boldsymbol{Y}}\langle\boldsymbol{\xi}\rangle = \boldsymbol{\xi} + \boldsymbol{\eta} \tag{7.31}$$

and the corresponding scalar state is obtained by

$$\underline{y} = |\underline{\boldsymbol{Y}}| \tag{7.32}$$

Now, we can define the *extension scalar state* by

$$\underline{e} = \underline{y} - \underline{x} \tag{7.33}$$

and the weighted volume is given by

$$m = \underline{\omega x} \cdot \underline{x} \tag{7.34}$$

Further, we define the scalar-valued *dilatation* function

$$\theta\left(\underline{e}\right) = \frac{3}{m}\underline{\omega x} \cdot \underline{e} \tag{7.35}$$

The extension scalar state can be decomposed into isotropic (hydrostatic) and deviatoric parts respectively by

$$\underline{e}^{\mathrm{hyd}} = \frac{\theta}{3}\underline{x} \tag{7.36}$$

and

$$\underline{e}^{\mathrm{dev}} = \underline{e} - \underline{e}^{\mathrm{hyd}} \tag{7.37}$$

For isotropic linear elastic material, constants in Eq. (7.28) can be calibrated by an approach similar to bond-based PM material, which leads to

$$k = K \tag{7.38}$$

and

$$\alpha = \frac{15\mu}{m} \tag{7.39}$$

where K and μ are the bulk and shear moduli, respectively.

Therefore, we have

$$\underline{t} = \frac{3k\theta}{m}\left(K\theta\underline{\omega x} + 5\mu\underline{\omega e}^{\mathrm{dev}}\right) \tag{7.40}$$

In comparison, the isotropic linear elastic constitutive law in CCM is given by

$$\boldsymbol{\sigma} = 3K\boldsymbol{\varepsilon}^{\mathrm{hyd}} + 2\mu\varepsilon^{\mathrm{dev}} \tag{7.41}$$

where $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ are the stress and strain tensors, respectively. It can be seen that Eqs. (7.40) and (7.41) are similar in their mathematical structures.

7.2.2.2 Non-ordinary state-based PD

In contrast to the ordinary state-based PD, the force density vector state is not necessarily parallel to the deformed bond in the non-ordinary state-based PD and the conservation of angular momentum must be considered in the material model. One class of non-ordinary material model is called the *correspondence* material, which enables direct application of constitutive models in CCM and greatly expands the material library in PD.

The process of constructing PD with correspondence material model is illustrated as follows: first, the *shape tensor* is defined by

$$\boldsymbol{K} = \underline{\boldsymbol{X}} \otimes \underline{\boldsymbol{X}} \tag{7.42}$$

which is symmetric positive definite.

Then we introduce the *deformed shape tensor* as

$$\boldsymbol{k} = \underline{\boldsymbol{Y}} \otimes \underline{\boldsymbol{X}} \tag{7.43}$$

Based on the above definitions, the nonlocal approximate deformation gradient tensor in correspondence material is given by

$$\boldsymbol{F} = \boldsymbol{k} \cdot \boldsymbol{K}^{-1} \tag{7.44}$$

The force density vector state is then given by

$$\underline{\boldsymbol{T}} \langle \boldsymbol{\xi} \rangle = \underline{\omega} \langle \boldsymbol{\xi} \rangle \boldsymbol{P} (\boldsymbol{F}) \cdot \boldsymbol{K}^{-1} \cdot \boldsymbol{\xi} \tag{7.45}$$

in which $\boldsymbol{P}(\boldsymbol{F})$ is the nominal stress tensor that can be described by constitutive laws in CCM literature.

For linear elastic material under small deformation, we have

$$\boldsymbol{P} = \boldsymbol{\sigma} = \boldsymbol{C} : \boldsymbol{\varepsilon} \tag{7.46}$$

where $\boldsymbol{C}$ is the elasticity tensor and the strain tensor can be obtained by

$$\boldsymbol{\varepsilon} = \frac{1}{2} \left(\boldsymbol{F} + \boldsymbol{F}^T \right) - \boldsymbol{I} \tag{7.47}$$

where $\boldsymbol{I}$ is the identity tensor.

7.2.2.3 Stabilized non-ordinary state-based PD

Although the non-ordinary state-based correspondence model greatly extends the material library available to PD, it is known to suffer from instability caused by the so called zero energy modes. It was initially considered as an issue in numerical implementation and several strategies were proposed to alleviate the adverse effects caused by these spurious modes (Bessa et al., 2014; Breitenfeld et al., 2014; Tupek and Radovitzky, 2014; Ganzenmüller et al., 2015; Silling et al., 2015). However, it was later discovered by Silling (Silling, 2017) that the instability roots in the correspondence material instead of the meshless numerical implementation. Based on this finding, a number of new approaches were developed to stabilize the correspondence materials (Silling, 2017; Breitzman and Dayal, 2018; Li et al., 2018; Luo and Sundararaghavan, 2018; Chowdhury et al., 2019). A recent article by Chowdhury et al. (Chowdhury et al., 2019) provides a brief review on these stabilization techniques.

In this work, we adopt the stabilized formulation proposed by Silling (Silling, 2017). First, we define the nonuniform part of the deformation state by

$$\underline{\boldsymbol{Z}} = \underline{\boldsymbol{Y}} - \boldsymbol{F} \cdot \boldsymbol{\xi} \tag{7.48}$$

and its contribution to the approximate deformation gradient tensor vanishes:

$$\underline{\boldsymbol{Z}} \otimes \underline{\boldsymbol{X}} \cdot \boldsymbol{K}^{-1} = (\underline{\boldsymbol{Y}} - \boldsymbol{F} \cdot \boldsymbol{\xi}) \otimes \underline{\boldsymbol{X}} \cdot \boldsymbol{K}^{-1} = \boldsymbol{F} - \boldsymbol{F} \cdot \boldsymbol{K} \cdot \boldsymbol{K}^{-1} = \boldsymbol{0} \tag{7.49}$$

which indicates that the deformation state $\underline{\boldsymbol{Y}}$ is nonunique for the same approximate deformation gradient.

To suppress the material instability, the force density vector state is modified by adding a stabilization term:

$$\underline{\boldsymbol{T}}\langle\boldsymbol{\xi}\rangle = \underline{\omega}\langle\boldsymbol{\xi}\rangle \left(\boldsymbol{P}(\boldsymbol{F}) \cdot \boldsymbol{K}^{-1} \cdot \boldsymbol{\xi} + \frac{GC}{\omega_0} \underline{\boldsymbol{Z}}\langle\boldsymbol{\xi}\rangle \right) \tag{7.50}$$

where G is a positive constant of order 1, $C = c/\delta$ is the nominal micro-modulus in bond-based material and

$$\omega_0 = \int_{\mathcal{H}} \underline{\omega} \langle \boldsymbol{\xi} \rangle \, dV \tag{7.51}$$

Note that Eq. (7.50) reduces to Eq. (7.45) for $G = 0$.

7.3 Numerical implementation

7.3.1 Discretization

In most cases, the numerical implementation of PD leads to a meshfree formulation (Silling and Askari, 2005). In such a formulation, material points are represented by discrete nodes with associated volumes. In practice, spatial discretizations are first created by using available tools such as FEA packages or dedicated mesh generation packages. The FE mesh is then converted to PD nodes using two approaches. In the first approach, the element centroids or Gauss quadrature points are taken as PD nodes with the associated elemental volumes (or integration weights). However, this method naturally leads to an inaccurate representation of the geometrical boundaries. In the other approach, the FE nodes are kept as PD nodes so the geometry is accurate. Nodal volume is obtained by summation of volume fractions of elements associated to the particular node.

In this work, we adopt the second approach to generate PD nodes. Note that for simple cases, such as 1D cases or lattice/crystal structures in 2D and 3D, we can directly create PD nodes without using any mesh generation tools. In addition, for simplicity, we employ uniform nodal spacing for structured distribution of nodes and quasi-uniform nodal spacing for unstructured cases in this implementation.

7.3.2 Construction of family

Once the spatial discretization is established, the next step is to create bonds between nodes based on their relative position and the horizon radius, which is given as an input. Note

that in this work we employ constant horizon radius, which is typically around three times the average nodal spacing (Bobaru et al., 2016).

Construction of family generally involves two steps. The first step is searching for neighbor nodes within the horizon, which is a classical task in computer science. The exhaustive searching algorithm, i.e. computing distance between all node-pairs and then comparing it with the horizon radius, is simple and straightforward in terms of implementation. However, it leads to the worst computational time complexity of $\mathcal{O}(N^2)$ in which N is the number of nodes. In fact, searching algorithms with much better computational complexities, usually around $\mathcal{O}(N \log N)$, are readily available in the computer science community. In this work, we employ the k-d tree searching algorithm (Bentley, 1975) for general cases and the brute-force searching algorithm is also used only for small cases.

The second step of family construction is to establish the bonds between neighboring nodes and store all the family information. In general, a specific bond can be identified by using two indices, one for the particular node it is associated with and the other for the local bond index. This naturally leads to a matrix-like 2D data structure. However, this direct implementation is not efficient in terms of both storage space and access speed. Here, we propose a simple and efficient data structure for storage and access of the family information. The data structure is similar to the compressed sparse row (CSR) for sparse matrix storage introduced in Chapter 3. In this format, we have multiple 1D arrays with the size of total number of bonds to store all the bond-associated data and states, such as node-pair indices, bond deformation, bond status, etc. To access these bond data associated with a node, an 1D array of pointers with size of $N + 1$ is constructed to give the start and end positions of bonds associated with the particular node in the previous state arrays. This data format allows a very efficient numerical implementation of PD family in terms of both computational time and storage cost. All bond-related operations and calculations are vectorized in this way and can be accelerated by parallel computing as well. In addition,

all the bond-associated states are stored continuously in memory, which allows fast access during computation and further improves numerical efficiency.

7.3.3 Time integration

The discretized PD equation of motion is given by

$$\rho(\boldsymbol{x})\ddot{\boldsymbol{u}}(\boldsymbol{x},t) = \sum_{\mathcal{H}_{\boldsymbol{x}}} \boldsymbol{f}(\boldsymbol{x}',\boldsymbol{x},t)\Delta V_{\boldsymbol{x}'} + \boldsymbol{b}(\boldsymbol{x},t) \tag{7.52}$$

which is a system of second-order ordinary differential equations that can be solved using conventional time integration algorithms developed for other numerical approaches. In the PD community, *Velocity Verlet* has been widely used for explicit time integration. Other algorithms can also be used for PD. For example, Wada (Wada et al., 2018) recently developed a PD solver based on the *time-discontinuous Galerkin* approach introduced in Section 2.2 and demonstrated good accuracy and stability with a 1D wave propagation example.

In this work, we mainly focus on wave propagation and dynamic fracture problems. Hence, we employ an explicit scheme based on the *central difference* algorithm to solve Eq. (7.52), which is summarized in Table 7.1. The PD material models, either bond-based or state-based, are solved in Line 3: internal force density calculation. All PD bond-related calculations as well as state updates are also performed in this step. The external body force densities in Line 4 include both the applied loading and the forces due to contact. In the velocity update step (Line 6), we introduce an artificial damping coefficient $\alpha \leqslant 2$. There is no damping when $\alpha = 2$. In general, the value α should be smaller than but close to 2. The central difference algorithm is conditionally stable. Therefore, the employed time step size must be smaller than a critical value, which can be determined using the classical *Courant–Friedrichs–Lewy* (CFL) condition or the approach for PD described in (Bobaru et al., 2016).

Table 7.1. Pseudocode for explicit time integration of Peridynamics

Line number	Operation
1	Initialization
2	**DO** $n = 1$ to total number of time increments
3	Get internal force densities $\boldsymbol{f}_n^{\text{int}}$
4	Get external body force densities $\boldsymbol{f}_n^{\text{ext}}$
5	Compute accelerations $\boldsymbol{a}_n = \left(\boldsymbol{f}_n^{\text{int}} + \boldsymbol{f}_n^{\text{ext}}\right)/\rho\left(\boldsymbol{x}\right)$
6	Update velocities $\boldsymbol{v}_{n+1/2} = \begin{cases} \boldsymbol{v}_n + 1/2\Delta t\boldsymbol{a}_n & \text{if } n = 1 \\ (\alpha - 1)\boldsymbol{v}_{n-1/2} + 1/2\alpha\Delta t\boldsymbol{a}_n & \text{if } n > 1 \end{cases}$
7	Enforce velocity boundary conditions
8	Update displacements $\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \Delta t\boldsymbol{v}_{n+1/2}$
9	Enforce displacement boundary conditions
10	Write output
11	**END DO**

7.3.4 Internal force density calculation

The internal force density calculation shown in Line 3 of Table 7.1 differs PD with other numerical methods in CCM. In PD, it is based on an integration over family nodes and involves zero spatial partial differentiation. The first step of PD internal force density calculation is updating the bond-associated states, which is presented in Table 7.2. At this step, the deformed relative position and direction vectors are updated based on current configuration. Bond breakage criteria are also implemented at this stage. Note that the simple brittle fracture criterion based on bond critical strain is shown here.

The second step is to calculate the internal force densities, which is discussed separately for bond-based and non-ordinary state-based materials. For bond-based materials, we consider the simple *prototype micro-elastic brittle* model as an example. The algorithm is shown in Table 7.3. It is worth noting that in Line 3 we obtain the start and end bond pointers in state arrays of a particular node using the efficient data structure proposed in Section 7.3.2. It allows a simple and fast access to the related states as shown in Lines $5 \sim 6$. In Line 10, we calculate the nodal damage value, which is defined by the ratio of the number of broken

Table 7.2. Pseudocode for updating bond states in Peridynamics

Line number	Operation
1	**DO** $n = 1$ to total number of bonds
2	Get bond associated nodal indices I_n and J_n
3	Get displacement vectors $\boldsymbol{u}_{I_n}$ and $\boldsymbol{u}_{J_n}$
4	Update deformed relative position vector $\boldsymbol{y}_n = \boldsymbol{u}_{J_n} - \boldsymbol{u}_{I_n} + \boldsymbol{\xi}_n$
5	Update deformed bond direction vector $\boldsymbol{M}_n = \boldsymbol{y}_n / \left\|\boldsymbol{y}_n\right\|$
6	Update bond strain $s_n = \left\|\boldsymbol{y}_n\right\| / \left\|\boldsymbol{\xi}_n\right\| - 1$
7	Break bond by setting influence function $\omega_n = 0 \quad$ if $s_n > s_c$
8	**END DO**

Table 7.3. Pseudocode for bond-based force density calculation

Line number	Operation
1	**DO** $m = 1$ to total number of nodes
2	Set $\boldsymbol{f}_m^{\text{int}} = \boldsymbol{0}, W_m = 0$
3	Get start and end positions of associated bonds in state arrays
4	**DO** $n =$ start to end positions in state arrays
5	Get the neighbor node volume ΔV_n
6	Get bond states $\omega_n, s_n, \boldsymbol{M}_n$
7	Compute internal force density $\boldsymbol{f}_m^{\text{int}} \mathrel{+}= cs_n\omega_n\Delta V_n\boldsymbol{M}_n$
8	Compute deformed bond $W_m \mathrel{+}= \omega_n$
9	**END DO**
10	Update node damage $D_m = 1 - W_m / N_m^{\text{bonds}}$
11	**END DO**

bonds to the number of all bonds associated with the particular node. Zero value means no damage and value of 1 means that all bonds associated with the node are broken and the node is fully disconnected from the rest of the model. Our implementation of the above bond counting is based on the influence function, which has a value of 0 for broken bonds and 1 otherwise.

For non-ordinary state-based PD with correspondence materials, the internal force density calculation is slightly more complicated compared to the bond-based material. It is done in two steps. First, the force density state is calculated by the algorithm shown in Table 7.4.

Table 7.4. Pseudocode for force density state calculation

Line number	Operation
1	**DO** $m = 1$ to total number of nodes
2	Set deformed shape tensor $\boldsymbol{k}_m = \boldsymbol{0}$
3	Get start and end positions of associated bonds in state arrays
4	**DO** n = start to end positions in state arrays
5	Get the neighbor node volume ΔV_n
6	Get bond states $\boldsymbol{\xi}_n, \boldsymbol{y}_n$
7	Compute deformed shape tensor $\boldsymbol{k}_m \mathrel{+}= \omega_n \Delta V_n \boldsymbol{y}_n \otimes \boldsymbol{\xi}_n$
8	**END DO**
9	Compute nonlocal deformation gradient tensor $\boldsymbol{F}_m = \boldsymbol{k}_m \cdot \boldsymbol{K}_m^{-1}$
10	Compute nominal stress tensor using CCM models $\boldsymbol{P}_m = \boldsymbol{P}_m(\boldsymbol{F}_m)$
11	**DO** n = start to end positions in state arrays
12	Compute force density state $\boldsymbol{T}_n = \omega_n \left(\boldsymbol{P}_m \boldsymbol{K}_m^{-1} \boldsymbol{\xi}_n + GC/\omega_0 \left(\boldsymbol{y}_n - \boldsymbol{F}_m \boldsymbol{\xi}_n\right)\right)$
13	**END DO**
14	**END DO**

Lines $4 \sim 8$ compute the deformed shape tensor. The nonlocal deformation gradient tensor is then calculated in Line 9. The nominal stress tensor, which is the transpose of the 1st Piola-Kirchhoff stress tensor (note that we adopt the notation in (Belytschko et al., 2014)), is calculated using constitutive models available in CCM. After that, another loop over the associated bonds is performed to compute the force density state. Here, we also included the stabilization force state in Line 12. The constant family volume ω_0 is defined in Eq. (7.51) and computed in the pre-processing stage.

Once the force density state is computed, we can move on to the calculation of state-based force density. The algorithm is summarized in Table 7.5. A special note is that, in Line 6, the second force density state is given by $\boldsymbol{T}'_n \langle -\boldsymbol{\xi}_n \rangle = \boldsymbol{T}[\boldsymbol{x}_{J_n}] \langle \boldsymbol{\xi}_n \rangle$, which is the force density state on the neighbor node with the corresponding bond that is reverse of the current bond. Such a bond is termed as pairing bond in this work. The indices to these pairing bonds are found and stored during the construction of the family stage in pre-processing.

Table 7.5. Pseudocode for state-based force density calculation

Line number	Operation
1	**DO** $m = 1$ to total number of nodes
2	Set $\boldsymbol{f}_m^{\text{int}} = \mathbf{0}$
3	Get start and end positions of associated bonds in state arrays
4	**DO** n = start to end positions in state arrays
5	Get the neighbor node volume ΔV_n
6	Get bond states $\boldsymbol{T}_n \langle \boldsymbol{\xi}_n \rangle, \boldsymbol{T}'_n \langle -\boldsymbol{\xi}_n \rangle$
7	Compute internal force density $\boldsymbol{f}_m^{\text{int}} \mathrel{+}= (\boldsymbol{T}_n + \boldsymbol{T}'_n)\, \Delta V_n$
8	**END DO**
9	**END DO**

Force density calculation subroutine is typically the most time-consuming part of the PD implementation. Its counterpart in FEM is solving the material constitutive models on Gauss quadrature points, which is usually expensive as well. It is easy to show that the above algorithms presented in Table 7.2 to Table 7.5 are essentially SIMD-type computing tasks, i.e. there is no data racing when parallel expanding the loop or the outer one if there are multiple loops. Therefore, the numerical implementation of PD developed in this work is well-suited for parallel computing with CPUs/GPUs. In the current work, we employ the OpenMP multithreading technique to accelerate the computing process.

7.4 Numerical examples

In this section, we provide several numerical examples to demonstrate the advantages of PD. The examples considered here include elastic wave propagation problems and dynamic brittle fracture problems.

7.4.1 Longitudinal vibration of a bar

In the first example, we consider a 1D bar of the length $L = 0.2$ m subjected to an initial strain $\varepsilon_0 = 0.001$. The bar has a uniform cross-sectional area $A = 1 \times 10^{-6}$ m^2 and is

made of a material with Young's modulus $E = 70$ GPa and density $\rho = 2700$ kg/m^3. It is fixed at one end and free at the other end. At time $t = 0$, the initial stretch is released. To simulate this problem, we employ four different versions of PD, namely bond-based (BBPD), ordinary state-based (OPD), non-ordinary state-based (NOPD) and the stabilized (SNOPD) formulations. The bar is discretized by evenly distributed nodes with a nodal spacing $\Delta X = 0.001$ m. The total simulation time is 4×10^{-4} s. For all the cases, the horizon radius is given by $\delta = n\Delta X$ and $n = 1, 2, 3, 4$. For SNOPD, the stabilization parameter is given by $G = 0.5$.

A closed-form solution to this problem can be found in (Rao, 2011) and is given by

$$u(x,t) = \frac{8\varepsilon_0 L}{\pi^2} \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^2} \sin\left(\frac{(2n+1)\pi}{2} x\right) \cos\left(\sqrt{\frac{E}{\rho}} \frac{(2n+1)\pi}{2} t\right) \tag{7.53}$$

which is employed here for the purpose of comparison.

Figure 7.1 shows the displacement histories at $x = 0.1$ m obtained by the BBPD. It can be seen that the results agree well with the closed-form solution for different horizons. One may also notice that with the increasing horizon radius, numerical solutions are slightly deviated from the exact solution, which is caused by the dispersive wave propagation behavior due to nonlocality in PD.

The results obtained by OPD simulations are shown in Figure 7.2, which also demonstrated a good agreement with the closed-form solution.

Figure 7.3 illustrates the displacement histories obtained by NOPD. In Section 7.2.2.3, we have shown that NOPD is unstable due to material instability, which leads to spurious zero-energy modes. This adverse effect can be clearly observed in Figure 7.3 (b) $\sim$ (d). With a larger horizon, the instability becomes more evident, which is expected due to the increasing nonlocality.

Finally, the numerical solutions obtained from SNOPD are shown in Figure 7.4. With the stabilization force densities introduced into the equation of motion, the zero-energy modes

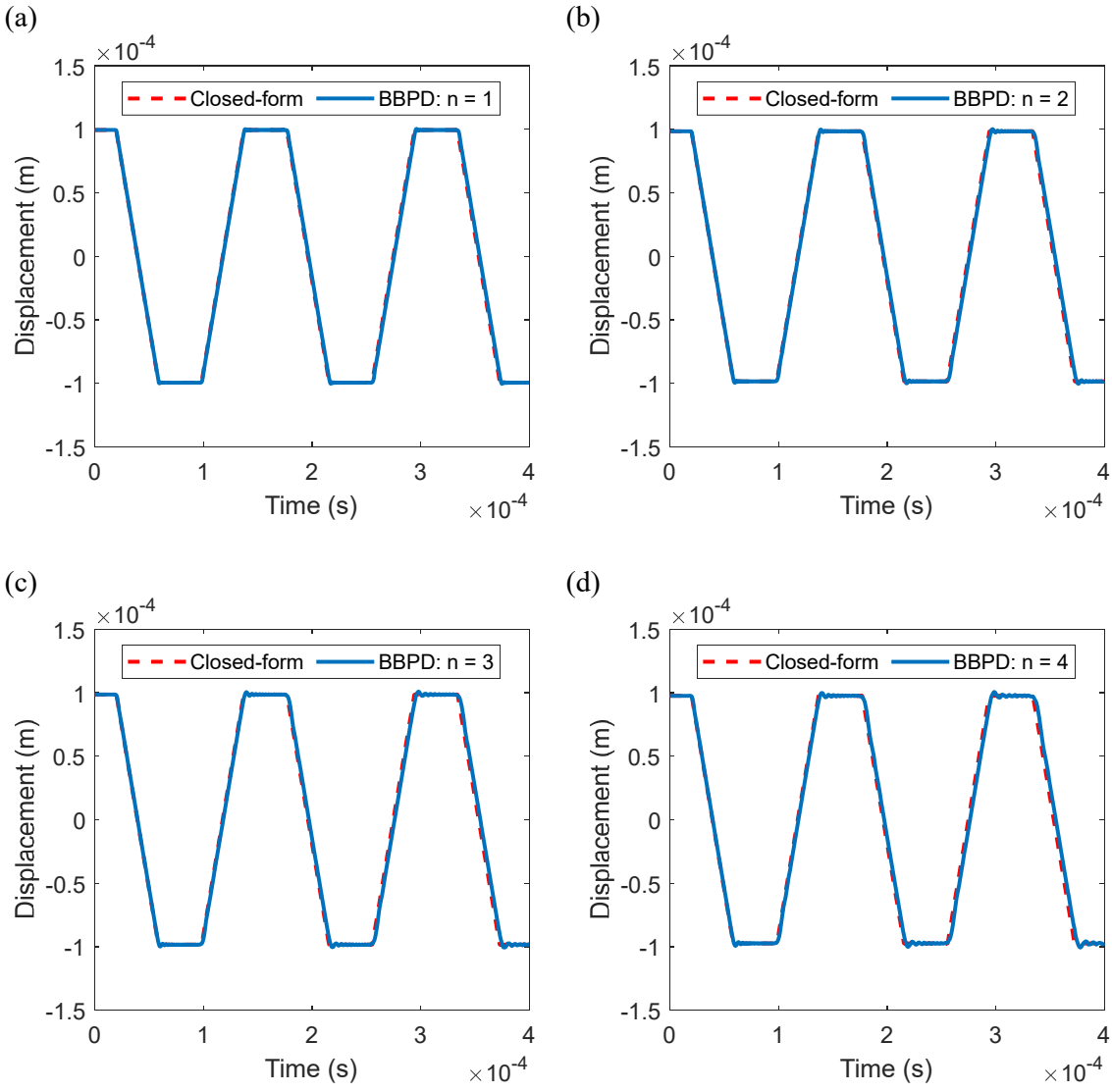


Figure 7.1. Displacement history at middle of the 1D bar obtained by bond-based PD (BBPD) with various neighborhood size: (a) $n = 1$, (b) $n = 2$, (c) $n = 3$, and (4) $n = 4$

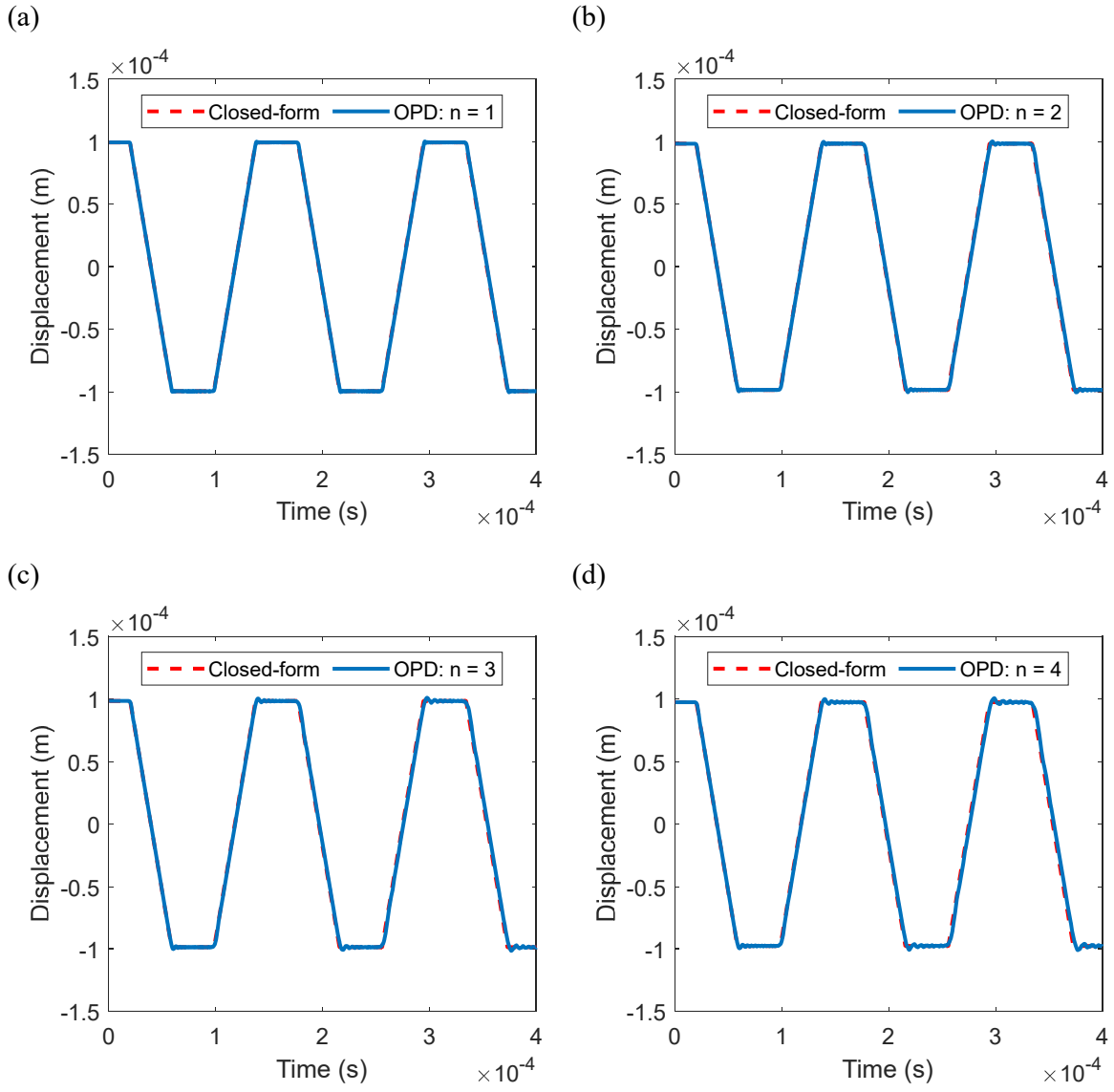


Figure 7.2. Displacement history at middle of the 1D bar obtained by ordinary state-based PD (OPD) with various neighborhood size: (a) $n = 1$, (b) $n = 2$, (c) $n = 3$, and (4) $n = 4$

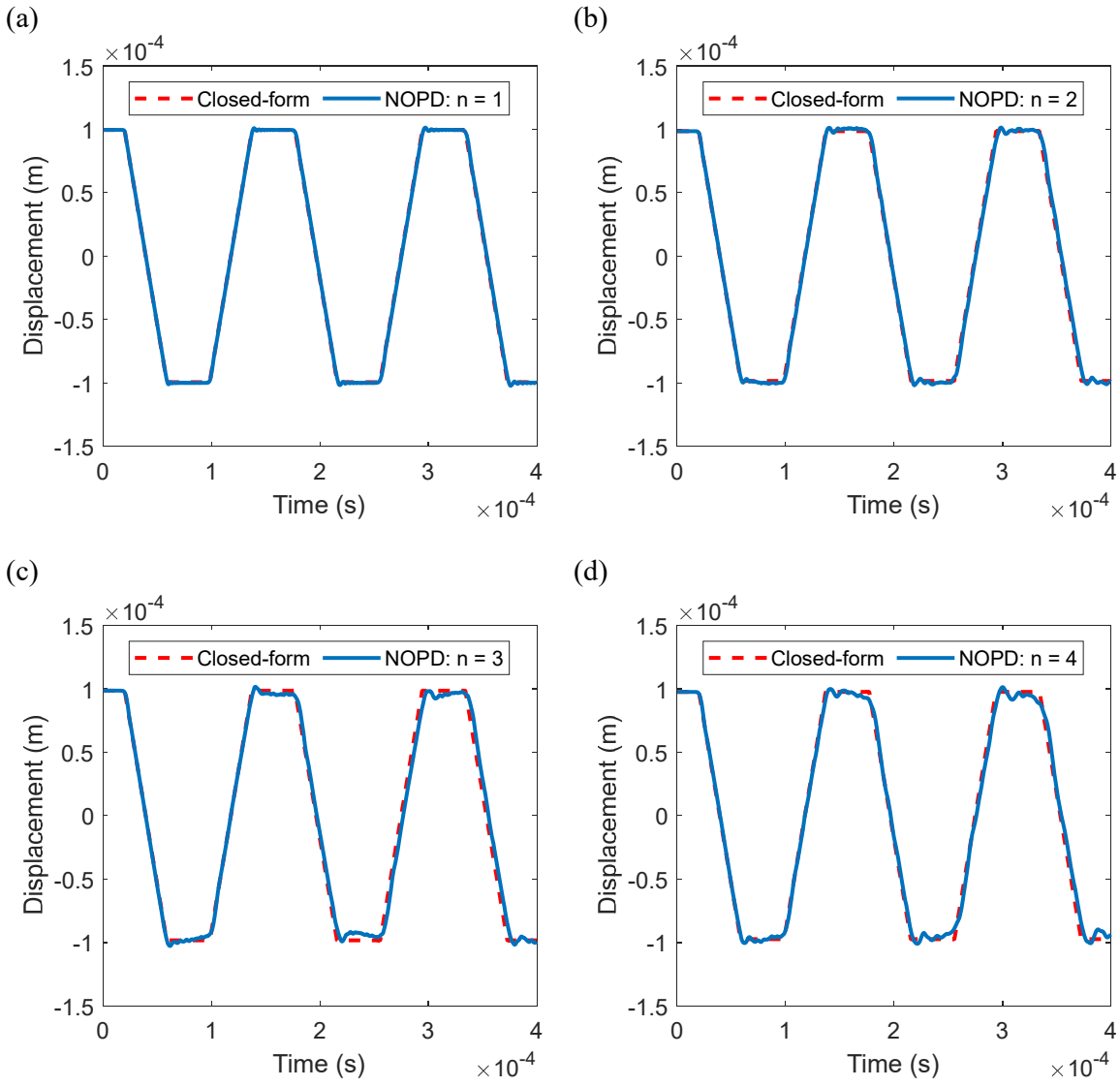


Figure 7.3. Displacement history at middle of the 1D bar obtained by non-ordinary state-based PD (NOPD) with various neighborhood size: (a) $n = 1$, (b) $n = 2$, (c) $n = 3$, and (4) $n = 4$

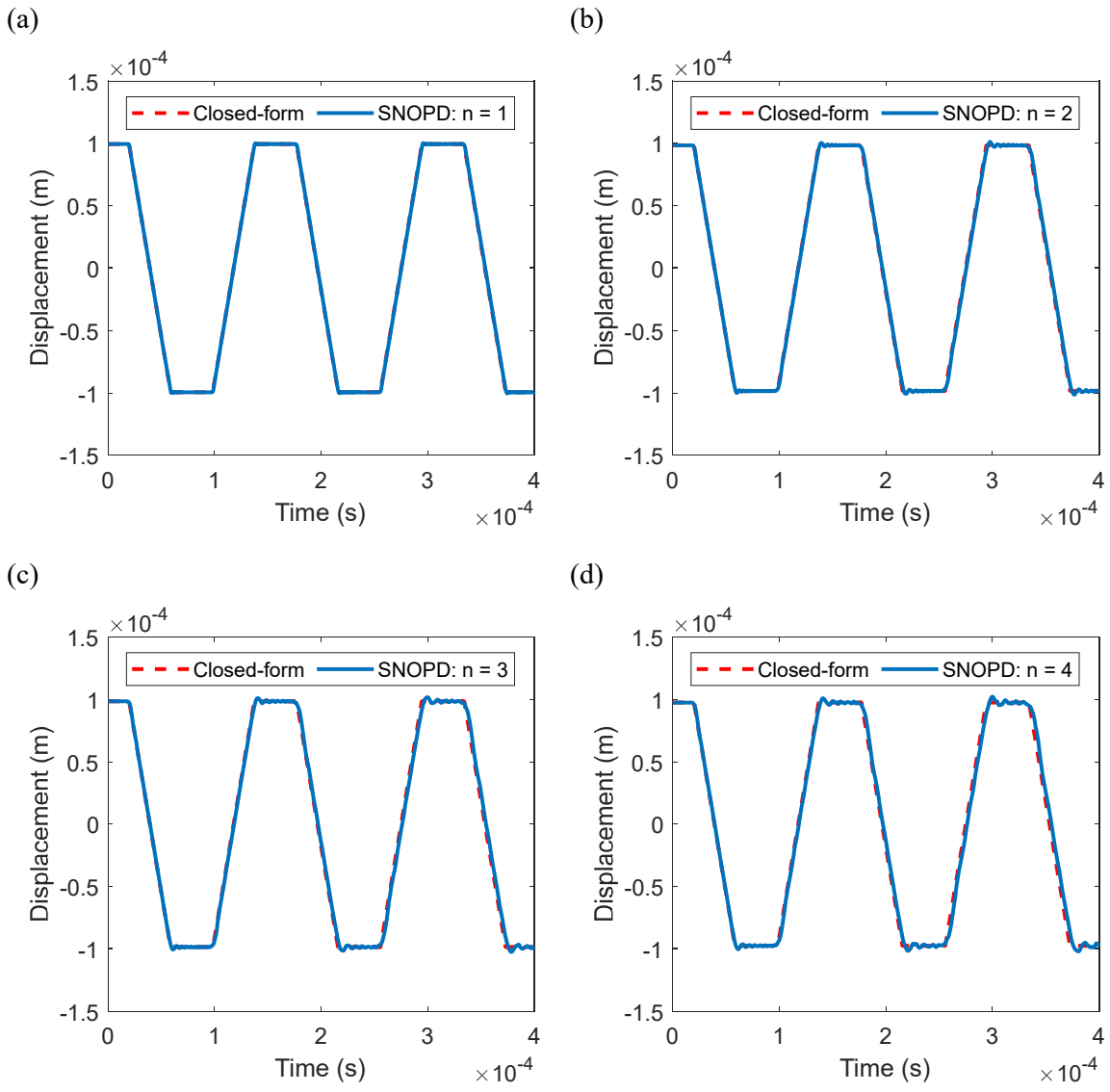


Figure 7.4. Displacement history at middle of the 1D bar obtained by stabilized non-ordinary state-based PD (SNOPD) with $G = 0.5$ and various neighborhood size: (a) $n = 1$, (b) $n = 2$, (c) $n = 3$, and (4) $n = 4$

are effectively eliminated, which is clear by comparing the solutions shown in Figure 7.4 (b) $\sim$ (d) with the counterparts shown in Figure 7.3. The accuracy of the numerical solutions is greatly improved. The SNOPD solution is as good as the BBPD or OPD, which does not suffer from the material instability issue.

7.4.2 Harmonic wave propagation of a bar

To further demonstrate the instability in NOPD, we consider another wave propagation problem in a 1D bar. The parameters are given as follows: bar length $L = 500$, cross-sectional area $A = 1$, Young's modulus $E = 57.1464$, density $\rho = 1$, nodal spacing $\Delta X = 1$, horizon radius $\delta = 3$ and $G = 0.5$ for SNOPD.

The bar is free at the both ends. A harmonic wave profile is imposed in the following form:

$$u(x,t) = u_0(1 - \cos t) \qquad \text{for } x \in [L/2 - 2\delta, L/2 - 2\delta], t \in [0, 2\pi] \tag{7.54}$$

where $u_0 = 0.0025$.

Figure 7.5 provides a comparison of displacement fields captured at different time steps for NOPD and SNOPD. The instability caused by zero-energy modes is clearly observed in the NOPD solutions. In contrast, the wave propagation solutions obtained by SNOPD are smooth and no longer suffer from the material instability.

7.4.3 Kalthoff-Winkler experiment

In this example, we employ the bond-based PD with the PMB material to simulate the Kalthoff-Winkler experiment (Kalthoff and Winkler, 1988), which is a classical benchmark problem for dynamic fracture simulation. In this experiment, a steel plate with two notches is impacted by a cylindrical projectile. Detailed problem statement can be found in both (Madenci and Oterkus, 2014) and (Belytschko et al., 2014). The geometric dimensions are shown in Figure 7.6 and thickness of the plate is 9 mm. The plate is initially at rest and free of constraints. The initial velocity of the projectile is $v_0 = 16.5$ m/s and it is modeled as a discretized rigid body with a total mass of 1.57 kg. Contact between the plate and the projectile is modeled by a frictionless, normal-force only behavior with the simple node-to-surface contact penetration detection algorithm. Material properties of the plate are

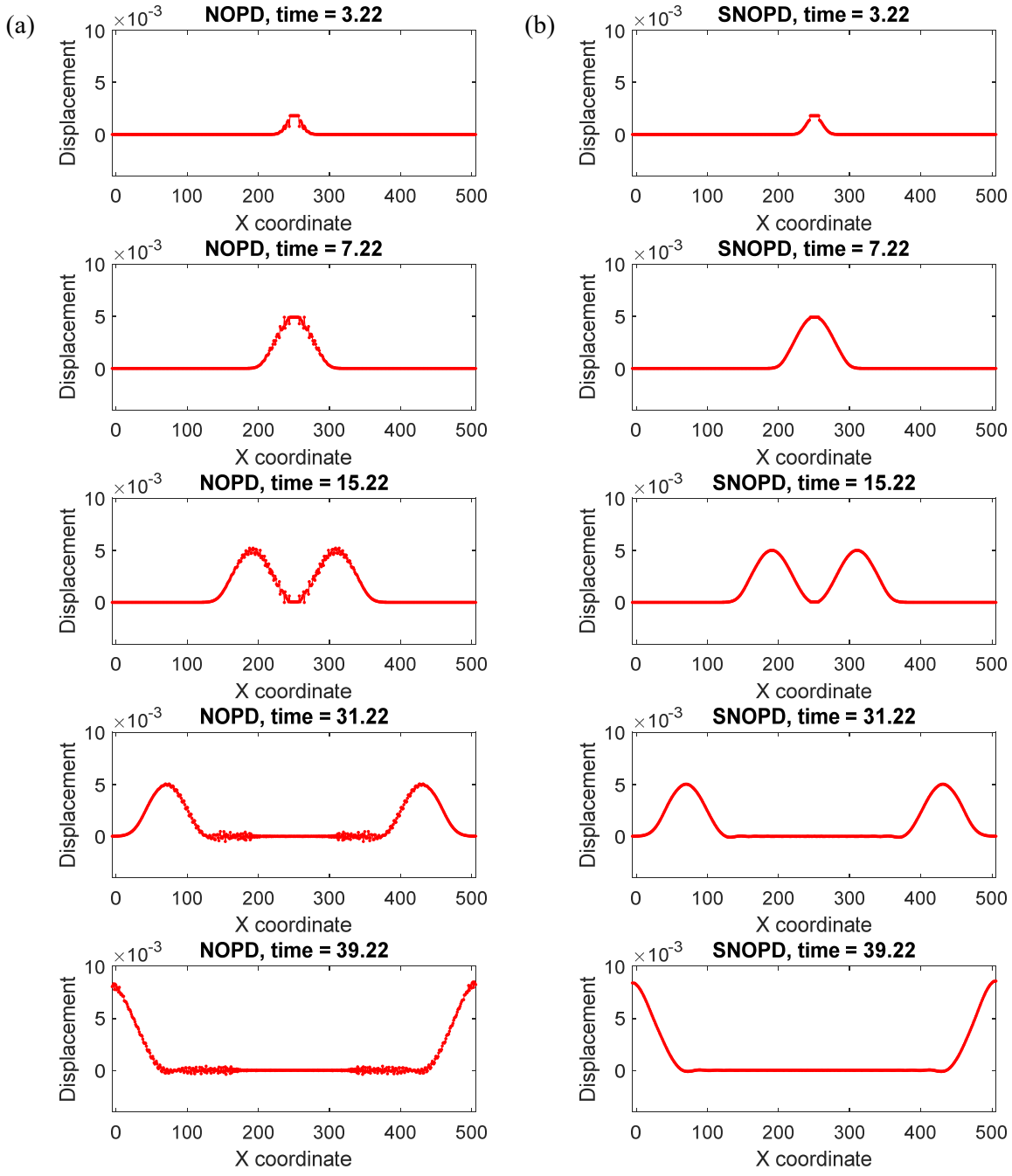


Figure 7.5. Timeframes of wave propagation in the 1D bar obtained by (a) non-ordinary state-based Peridynamics (NOPD) with correspondence materials and $n = 3$ and (b) the stabilized version (SNOPD) with $n = 3$ and $G = 0.5$

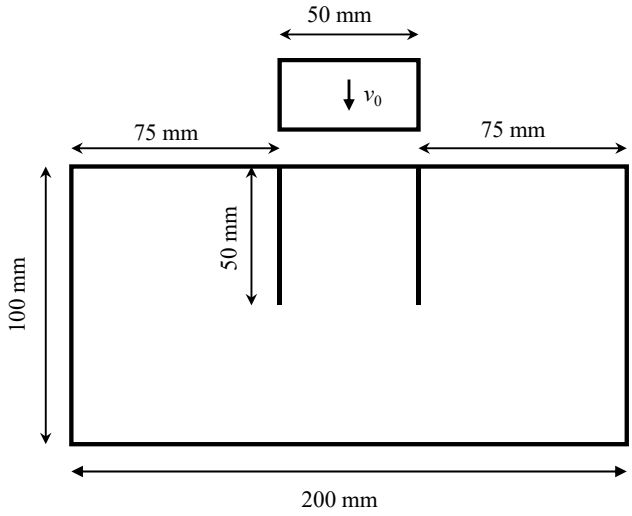


Figure 7.6. Kalthoff-Winkler experiment dimensions

given as follows: density $\rho = 8000$ kg/m^3, Young's modulus $E = 191$ GPa and critical strain $s_c = 0.01$. The plate is discretized in 2D with uniformly distributed nodes with the nodal spacing $\Delta X = 1$ mm, which leads to 20,301 nodes. We employ the horizon radius of $\delta = 3.015\Delta X$. The time increment size is $\Delta t = 8.7 \times 10^{-8}$ s and the total number of time steps is 1,350. The two initial notches are modeled by breaking the bonds that intercept with the notch line segments at the family construction stage.

Simulation results are shown in Figure 7.7. Nodal velocity magnitude fields at different timeframes are captured and plotted in Figure 7.7 (a). The impact-induced waves propagate in the notched plate and reflect at the plate boundaries as well as the growing crack surfaces, which gradually lead to very complicated dynamic wave interactions. To better observe the propagation of the cracks, contour plots of nodal damage field at a series of time steps are provided in Figure 7.7 (b). The crack paths and their propagations can be clearly identified by the color of damage variables. It is worth noting that the damage in PD is nothing but a counting of broken bonds and there is no explicit damage evolution law. There is also no criterion as to when the crack propagates or in which direction it propagates. All the

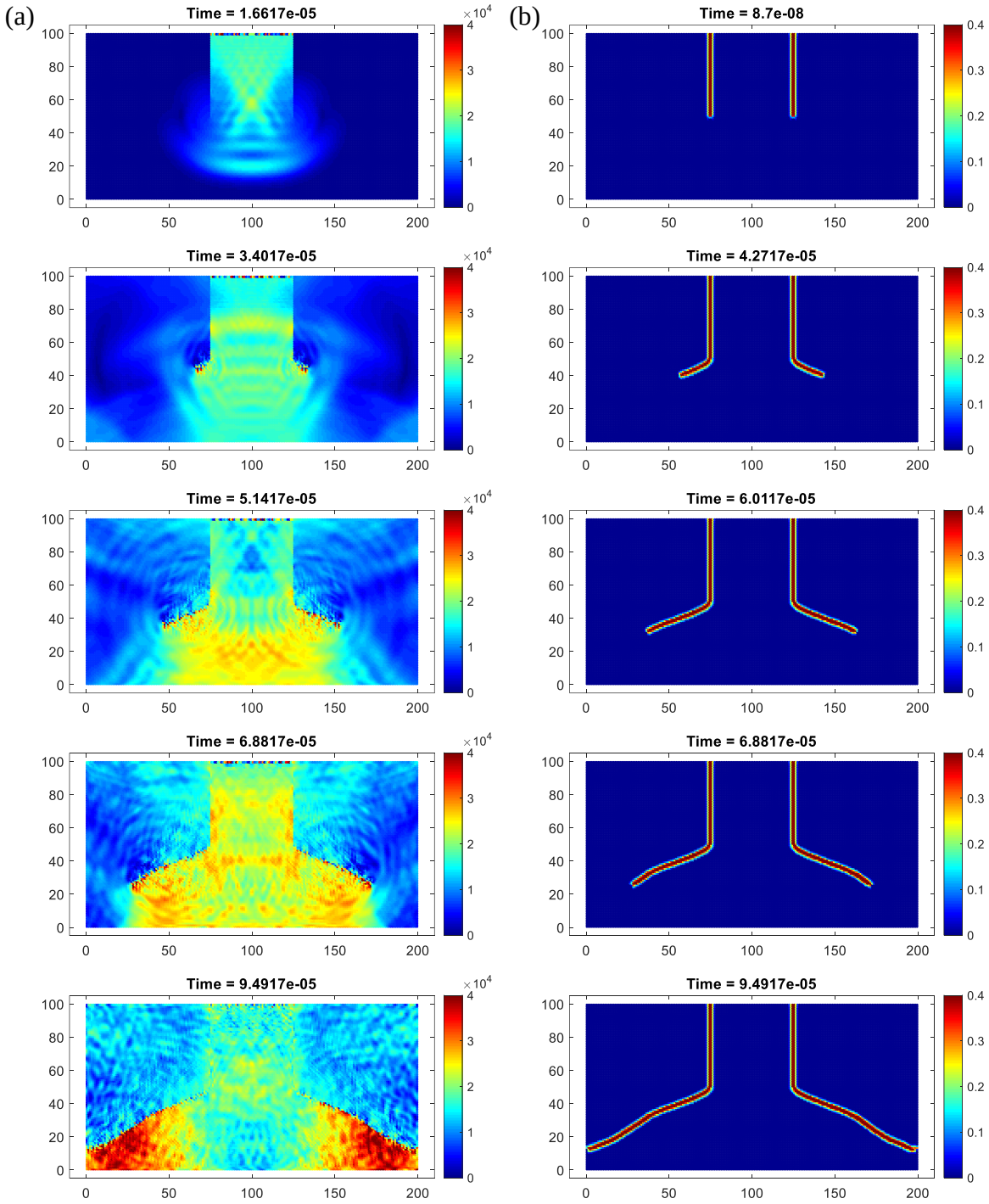


Figure 7.7. Kalthoff-Winkler simulation results: (a) velocity magnitude and (b) damage

complex behaviors of cracks are governed only by a simple brittle fracture rule which breaks bonds when its strain exceeds the critical value. Despite the very simple PMB material model employed here, the simulated crack angle is around 68° to 69°, which is very close to experiment observed angle of 70° and much better compared to the result of 58° reported in (Song et al., 2008) using XFEM.

7.4.4 Dynamic fracture of particle reinforced composite

In this example, we study the dynamic fracture of particle reinforced composite material using SNOPD in 2D with plane stress formulation. The representative volume element (RVE) of the composite material has a dimension of 50×50 mm^2 with a unit thickness. The matrix material is rubber and modeled by the modified Mooney-Rivlin Hyperelastic model with the parameter as follows: density $\rho = 930$ kg/m^3, Mooney-Rivlin constants $c_{10} = 1.0$ MPa and $c_{01} = 0.5$ MPa. The material behavior for particles is isotropic linear elastic with density $\rho = 9300$ kg/m^3, Young's modulus $E = 88$ MPa and Poisson's ratio $\nu = 0.49$. Damage in rubber matrix and matrix/particle interface are modeled by the brittle fracture behavior. For bonds connecting rubber material points, the critical strain is 0.25. For bonds between matrix and particle material points, the critical strain is 0.05, which is weaker than that of the matrix.

Particles are generated with a random distribution as shown in Figure 7.8 (a). A total of 20 circular particles with a radius of 3 mm are created. The particle volume fraction is around 22%. The geometry is discretized by uniformly distributed nodes with a distance of 1 mm, which is shown in Figure 7.8 (b). The horizon radius is chosen to be 3 mm and we employ the SNOPD stabilization parameter $G = 1$. Bottom edge of the RVE is fixed while the top edge is subjected to a constant velocity at 50 mm/s.

The simulation result is shown in Figure 7.9. It can be seen that bonds start to break at the weaker particle/matrix interfaces, which lead to autonomous initiation of multiple

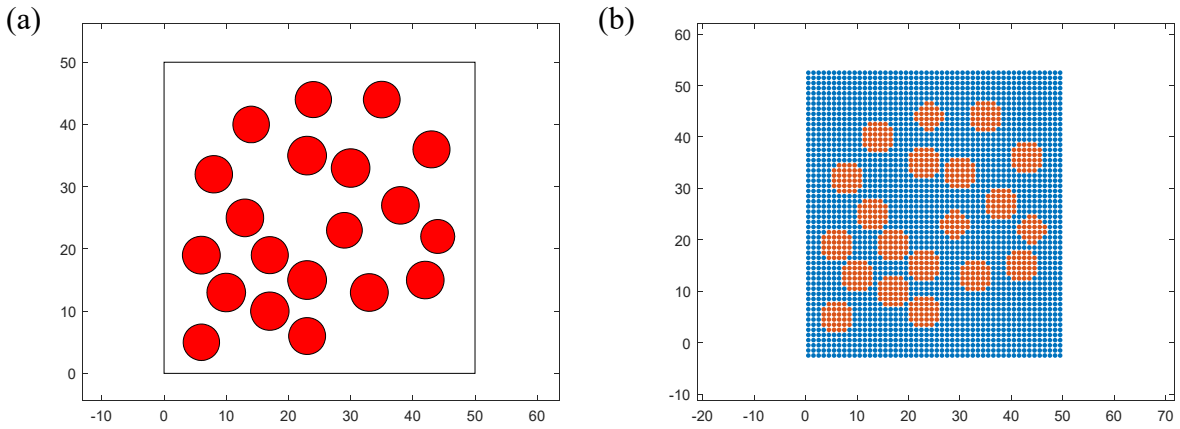


Figure 7.8. Particle reinforced composite material: (a) RVE and (b) PD discretization

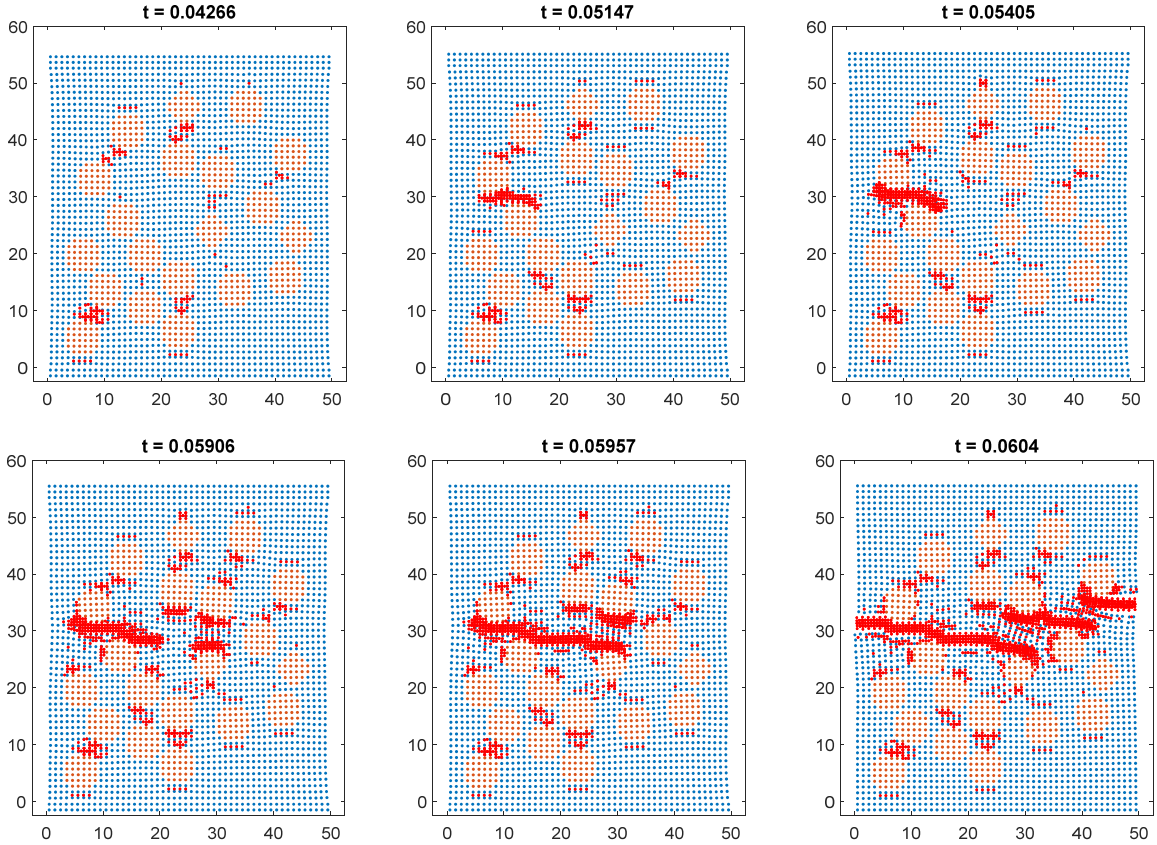


Figure 7.9. Autonomous cracks initiation and growth in SNOPD simulation on the particle reinforced composite, broken bonds are indicated by red dots

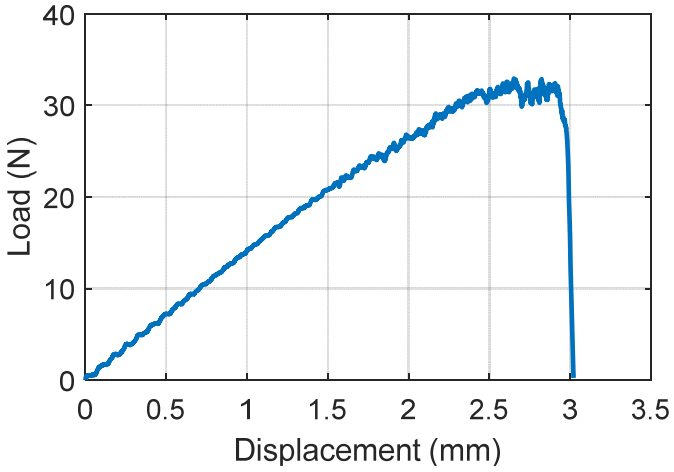


Figure 7.10. Load-displacement curve obtained by SNOPD simulation on the particle reinforced composite

cracks. With the increased load, cracks propagate along the interface and interact with each other. Bonds in the matrix material start to break as well. Finally, several major cracks form and propagate through the entire width of the RVE, which leads to fracture failure.

Figure 7.10 shows the load-displacement curve obtained from the SNOPD simulation. Since we impose velocity boundary condition on the top edge, the load is obtained from the total reaction force. The load-displacement curve characterizes homogenized mechanical behavior of the composite RVE. It shows a linear relationship until reach the maximum load. A short yielding range can be observed and follows by a brittle fracture. With SNOPD, we can further study the effects of particle shape, volume fraction, material properties, etc. on the macroscale behavior of composite materials, which is very helpful in material design.

7.5 Summary

In this chapter, we reviewed the mathematical formulations of both the bond-based and state-based PD. Efficient algorithms are developed for the numerical implementation of PD. The numerous numerical examples presented in this chapter clearly demonstrated the advan-

tages of PD over *Classical Continuum Mechanics* in terms of simulating dynamic fracture problems. However, PD leads to a high computational cost also due to its intrinsic nonlocality. The advantages as well as the limitations of PD motivates us to develop an accurate and efficient concurrent multiscale framework by coupling PD with CCM numerical methods.

CHAPTER 8

WAVE DISPERSION ANALYSIS FOR PERIDYNAMICS

8.1 Introduction

Wave dispersion relation, $\omega(\kappa)$, relates the wavenumber κ of a wave to its frequency ω. It is of fundamental importance to numerical methods, especially the nonlocal ones such as *Peridynamics* (PD), for studying dynamic problems (Bažant et al., 2016; Gu et al., 2016; Butt et al., 2017; Dayal, 2017; Nicely et al., 2018; Nicely, 2018). In nonlocal theories, the dispersion relationship is nonlinear and the resulting wave propagation behavior is dispersive. In this chapter, wave dispersion analysis for PD is performed with simplified wave propagation examples for discretized PD in both 1D and 2D cases. In this work, we focus on two specific PD formulations introduced in Chapter 7: bond-based PD (BBPD) and stabilized non-ordinary state-based PD with correspondence materials (SNOPD). The discretized wave dispersion relations for these two formulations are derived and studied in this chapter.

8.2 Dispersion relations in 1D

For 1D case, we consider an infinitely long bar with a unit cross-sectional area and zero body force. The material behavior is isotropic linear elastic. The bar is discretized by uniformly distributed nodes with a spacing of ΔX. We set the horizon radius to be $\delta = n\Delta X$, in which $n = 1, 2, 3, \cdots$. Dispersion relations under the assumption of small deformation, i.e. $|\boldsymbol{\xi}| \gg |\boldsymbol{\eta}|$, will be derived for PD formulations introduced in the previous section.

8.2.1 Bond-based PD

The discretized bond-based PD equation of motion for I-th node is given by

$$\rho\ddot{u}_I\left(x_I, t\right) = \frac{2E}{\left(n\Delta X\right)^2} \sum_{\substack{p=-n \\ p\neq 0}}^{n} W_p \frac{u_{I+p} - u_I}{|p|} \tag{8.1}$$

where W_p is the volume correction factor defined by

$$W_p = \begin{cases} \frac{1}{2} & |p| = n \\ 1 & \text{otherwise} \end{cases} \tag{8.2}$$

It can be shown that for local case, i.e. $n = 1$, Eq. (8.1) reduces to the central difference scheme for approximating second order derivative:

$$\rho \ddot{u}_I (x_I, t) = \frac{E (u_{I-1} - 2u_I + u_{I+1})}{(\Delta X)^2} \tag{8.3}$$

To derive dispersion relation, we first introduce the following harmonic wave solution

$$u (x, t) = \bar{u} e^{\mathrm{i}(\omega t + \kappa x)} \tag{8.4}$$

where $\bar{u}$ is the wave amplitude, $\mathrm{i} = \sqrt{-1}$, ω is the angular frequency, κ is the wavenumber (also known as the spatial frequency).

Suppose $x_I = I\Delta X$ and substituting Eq. (8.4) into Eq. (8.1) yield

$$-\rho\omega^2 = \frac{2E}{(n\Delta X)^2} \sum_{\substack{p=-n \\ p\neq 0}}^{n} W_p \frac{\left(e^{\mathrm{i}\kappa p \Delta X} - 1\right)}{|p|} \tag{8.5}$$

which is evaluated at $I = 0$.

By using Euler's formula, we have

$$-\rho\omega^2 = \frac{2E}{(n\Delta X)^2} \sum_{\substack{p=-n \\ p\neq 0}}^{n} \frac{W_p}{|p|} \left(\cos(\kappa p \Delta X) - 1 + \mathrm{i}\sin(\kappa p \Delta X)\right) \tag{8.6}$$

Noticing that the summation in Eq. (8.6) is over a symmetric domain and the sine function is odd, its contribution vanishes and we obtain

$$-\rho\omega^2 = \frac{2E}{(n\Delta X)^2} \sum_{\substack{p=-n \\ p\neq 0}}^{n} \frac{W_p}{|p|} \left(\cos(\kappa p \Delta X) - 1\right) \tag{8.7}$$

which leads to the final dispersion relation given by

$$\omega(\kappa) = \frac{2\sqrt{2}c}{n\Delta X} \sqrt{\sum_{p=1}^{n} \frac{W_p}{p} \sin^2\left(\frac{\kappa p \Delta X}{2}\right)} \tag{8.8}$$

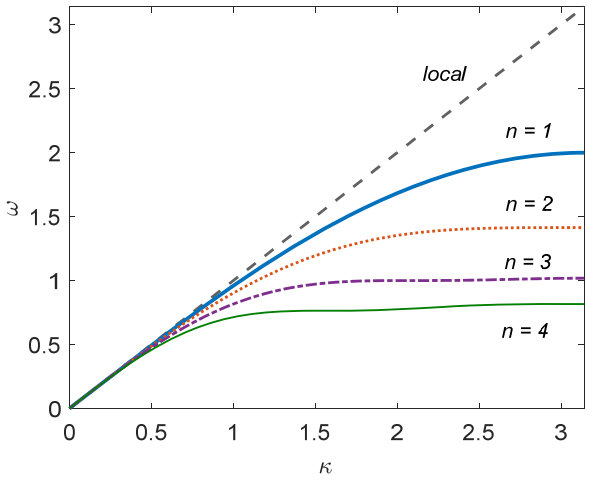


Figure 8.1. Wave dispersion relation for BBPD in 1D

where $c = \sqrt{E/\rho}$ is the speed of sound of the corresponding medium.

Let $c = 1$ and $\Delta X = 1$, the bond-based PD dispersion relation derived above is plotted in Figure 8.1 for cases of $n = 1 \sim 4$. For the purpose of comparison, the local case from CCM where the wave behavior is nondispersive, i.e. $\omega = c\kappa$, is also plotted. As it can be seen, in the limit of long wave ($\kappa \to 0$) all PD cases converged to the local theory. In addition, larger n leads to more dispersive wave behavior due to the nonlocality of PD.

8.2.2 State-based PD

Now we derive wave dispersion relation for stabilized non-ordinary state-based PD with correspondence materials. The continuous state-based equation of motion is given by

$$\rho(\boldsymbol{x})\,\ddot{\boldsymbol{u}}(\boldsymbol{x},t) = \int_{\mathcal{H}_{\boldsymbol{x}}} \left(\underline{\boldsymbol{T}}[\boldsymbol{x},t]\langle\boldsymbol{\xi}\rangle - \underline{\boldsymbol{T}}[\boldsymbol{x}',t]\langle-\boldsymbol{\xi}\rangle\right) dV_{\boldsymbol{x}'} \tag{8.9}$$

Correspondingly, the discretized equation of motion is given by

$$\rho\ddot{u}_I(x_I,t) = \sum_{\substack{p=-n \\ p\neq 0}}^{n} \left(\underline{T}[x_I]\langle x_{I+p} - x_I\rangle - \underline{T}[x_{I+p}]\langle x_I - x_{I+p}\rangle\right)\Delta X \tag{8.10}$$

In order to derive the force density state, we first evaluate the shape tensor as

$$K\left(x_I\right) = a_n \left(\Delta X\right)^3 \tag{8.11}$$

where

$$a_n = \frac{n(n+1)(2n+1)}{3} \tag{8.12}$$

Similarly, the deformed shape tensor is

$$k\left(x_I\right) = \sum_{\substack{q=-n \\ q\neq 0}}^{n} \left(q\Delta X + u_{I+q} - u_I\right)\left(q\Delta X\right)\Delta X \tag{8.13}$$

which can be further simplified by using symmetry as

$$k\left(x_I\right) = K\left(x_I\right) + \left(\Delta X\right)^2 \sum_{\substack{q=-n \\ q\neq 0}}^{n} q u_{I+q} \tag{8.14}$$

Then we find the approximate deformation gradient as

$$F\left(x_I\right) = \frac{k\left(x_I\right)}{K\left(x_I\right)} = 1 + \frac{1}{a_n \Delta X} \sum_{\substack{q=-n \\ q\neq 0}}^{n} q u_{I+q} \tag{8.15}$$

which leads to strain

$$\varepsilon\left(x_I\right) = \frac{1}{a_n \Delta X} \sum_{\substack{q=-n \\ q\neq 0}}^{n} q u_{I+q} \tag{8.16}$$

and stress

$$\sigma\left(x_I\right) = \frac{E}{a_n \Delta X} \sum_{\substack{q=-n \\ q\neq 0}}^{n} q u_{I+q} \tag{8.17}$$

For a node at $x_I = I\Delta X$, its force density state regarding node $x_J = J\Delta X$ is obtained by

$$\underline{T}\left[x_I\right]\left\langle x_J - x_I\right\rangle = \frac{(J-I)E}{a_n^2 (\Delta X)^3} \sum_{\substack{q=-n \\ q\neq 0}}^{n} q u_{I+q} \tag{8.18}$$

where we assumed that x_J is a neighbor of x_I, i.e. $|J - I| \leqslant n$.

Substituting Eq. (8.18) into Eq. (8.10) yields

$$\rho\ddot{u}_I\left(x_I,t\right)=\frac{E}{\left(a_n\Delta X\right)^2}\sum_{\substack{p=-n\\p\neq 0}}^{n}p\sum_{\substack{q=-n\\q\neq 0}}^{n}qu_{I+q}+\frac{E}{\left(a_n\Delta X\right)^2}\sum_{\substack{p=-n\\p\neq 0}}^{n}p\sum_{\substack{q=-n\\q\neq 0}}^{n}qu_{I+p+q} \tag{8.19}$$

Note that the first term in the right-hand-side of the above equation vanishes after summation over p. Therefore, we obtain the final form of the discretized equation of motion as

$$\rho\ddot{u}_I\left(x_I,t\right)=\frac{E}{\left(a_n\Delta X\right)^2}\sum_{\substack{p=-n\\p\neq 0}}^{n}\sum_{\substack{q=-n\\q\neq 0}}^{n}pqu_{I+p+q} \tag{8.20}$$

For the case of local neighborhood, i.e. $n=1$, the equation of motion is reduced to

$$\rho\ddot{u}_I\left(x_I,t\right)=\frac{E\left(u_{I-2}-2u_I+u_{I+2}\right)}{\left(2\Delta X\right)^2} \tag{8.21}$$

where a notable difference comparing to Eq. (8.3) (BBPD) is the node-skipping pattern that neglects nearest neighbor interactions, which leads to the non-physical zero-energy modes.

Substituting the wave solution Eq. (8.4) into Eq. (8.20) and evaluating at $I=0$ yield

$$\rho\omega^2=\frac{4E}{\left(a_n\Delta X\right)^2}\left(\sum_{p=1}^{n}p\sin\left(\kappa p\Delta X\right)\right)^2 \tag{8.22}$$

from which we take the positive root as the dispersion relation:

$$\omega(\kappa)=\frac{6c\left|\sum_{p=1}^{n}p\sin\left(\kappa p\Delta X\right)\right|}{n(n+1)(2n+1)\Delta X} \tag{8.23}$$

Let $c=1$, $\Delta X=1$, and $n=1\sim 4$, we plot the dispersion relation Eq. (8.23) as shown in Figure 8.2. It can be seen that angular frequency curves of all cases intercept the horizontal axis ($\omega=0$) at particular nonzero spatial frequencies. At these interception points, the phase velocity ($v_p=\frac{\omega}{\kappa}$) of wave is zero and the group velocity ($v_g=\frac{d\omega}{d\kappa}$) is undefined. This behavior is a manifestation of the zero-energy modes and leads to instability (Belytschko et al., 2000; Silling, 2017).

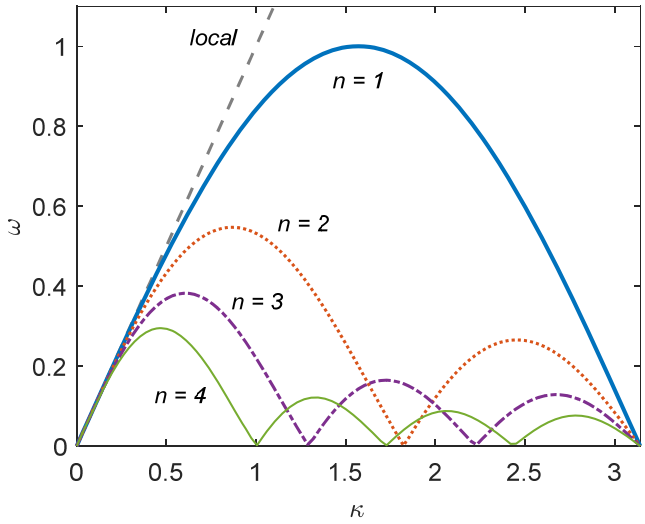


Figure 8.2. Wave dispersion relation for NOPD in 1D

For the stabilized force density state as shown in Eq. (7.50), the nonuniform part of the deformation state is given by

$$\underline{Z}\left[x_I\right]\left\langle x_J - x_I\right\rangle = \left(u_J - u_I\right) - \frac{J-I}{a_n}\sum_{\substack{q=-n\\q\neq 0}}^{n} q u_{I+q} \tag{8.24}$$

and its contribution to the internal force density is

$$\frac{GC}{\omega_0}\sum_{\substack{p=-n\\p\neq 0}}^{n}\left(\underline{Z}\left[x_I\right]-\underline{Z}\left[x_{I+p}\right]\right)\Delta X = \frac{EG}{n^4(\Delta X)^3}\sum_{\substack{p=-n\\p\neq 0}}^{n}\left(2\left(u_{I+p}-u_I\right)-\frac{p}{a_n}\sum_{\substack{q=-n\\q\neq 0}}^{n} q u_{I+p+q}\right) \tag{8.25}$$

where $C = \frac{2E}{A\delta^3}$.

Therefore, the discretized equation of motion with stabilization is given by

$$\begin{aligned}\rho\ddot{u}_I\left(x_I,t\right) = {} & \frac{E}{\left(a_n\Delta X\right)^2}\left(1-\frac{a_n G}{n^4\Delta X}\right)\sum_{\substack{p=-n\\p\neq 0}}^{n}\sum_{\substack{q=-n\\q\neq 0}}^{n} pq u_{I+p+q} \\ & + \frac{2EG}{n^4\left(\Delta X\right)^3}\sum_{\substack{p=-n\\p\neq 0}}^{n}\left(u_{I+p}-u_I\right)\end{aligned} \tag{8.26}$$

Let $n = 1$ we obtain the local equation of motion as

$$\begin{aligned}\rho\ddot{u}_I\left(x_I,t\right) = &\frac{E}{\left(2\Delta X\right)^2}\left(1-\frac{2G}{\Delta X}\right)\left(u_{I-2}-2u_I+u_{I+2}\right)\\ &+\frac{2EG}{\left(\Delta X\right)^3}\left(u_{I-1}-2u_I+u_{I+1}\right)\end{aligned} \tag{8.27}$$

in which the last term shows that nearest neighbor interactions are accounted for.

To derive the stabilized dispersion relation, we substitute the wave solution Eq. (8.4) into equation of motion Eq. (8.26) and obtain

$$\begin{aligned}\rho\omega^2 = &\frac{4E}{\left(a_n\Delta X\right)^2}\left(1-\frac{a_nG}{n^4\Delta X}\right)\left(\sum_{p=1}^{n}p\sin\left(\kappa p\Delta X\right)\right)^2\\ &-\frac{4EG}{n^4\left(\Delta X\right)^3}\sum_{p=1}^{n}\left(\cos\left(\kappa p\Delta X\right)-1\right)\end{aligned} \tag{8.28}$$

which leads to the following dispersion relation

$$\omega = 2c\sqrt{\frac{1}{\left(a_n\Delta X\right)^2}\left(1-\frac{a_nG}{n^4\Delta X}\right)\left(\sum_{p=1}^{n}p\sin\left(\kappa p\Delta X\right)\right)^2-\frac{G}{n^4(\Delta X)^3}\sum_{p=1}^{n}\left(\cos\left(\kappa p\Delta X\right)-1\right)} \tag{8.29}$$

Figure 8.3 shows the stabilized dispersion relation in Eq. (8.29) by setting $c = 1, \Delta X = 1, G = 0 \sim 0.5$, and $n = 1 \sim 4$. Note that by setting $G = 0$ it recovers the original, instable dispersion relation. For all cases with $G > 0$ the zero-energy modes are eliminated and larger G leads to a higher nonzero angular frequency at the zero-energy wavenumbers. However, value of G should not be too large otherwise the artificial stabilizing force overwhelms the original material response and affects the solution adversely.

8.3 Dispersion relations in 2D

The above derivation of 1D wave dispersion relations can be further extended to 2D. Here we consider a 2D uniform square lattice grid with a unit thickness and free of body force. The material behavior is isotropic linear elastic. The nodal spacings in both the x and the y directions are ΔX and the horizon radius is assumed to be $\delta = \Delta X$, i.e. the nearest

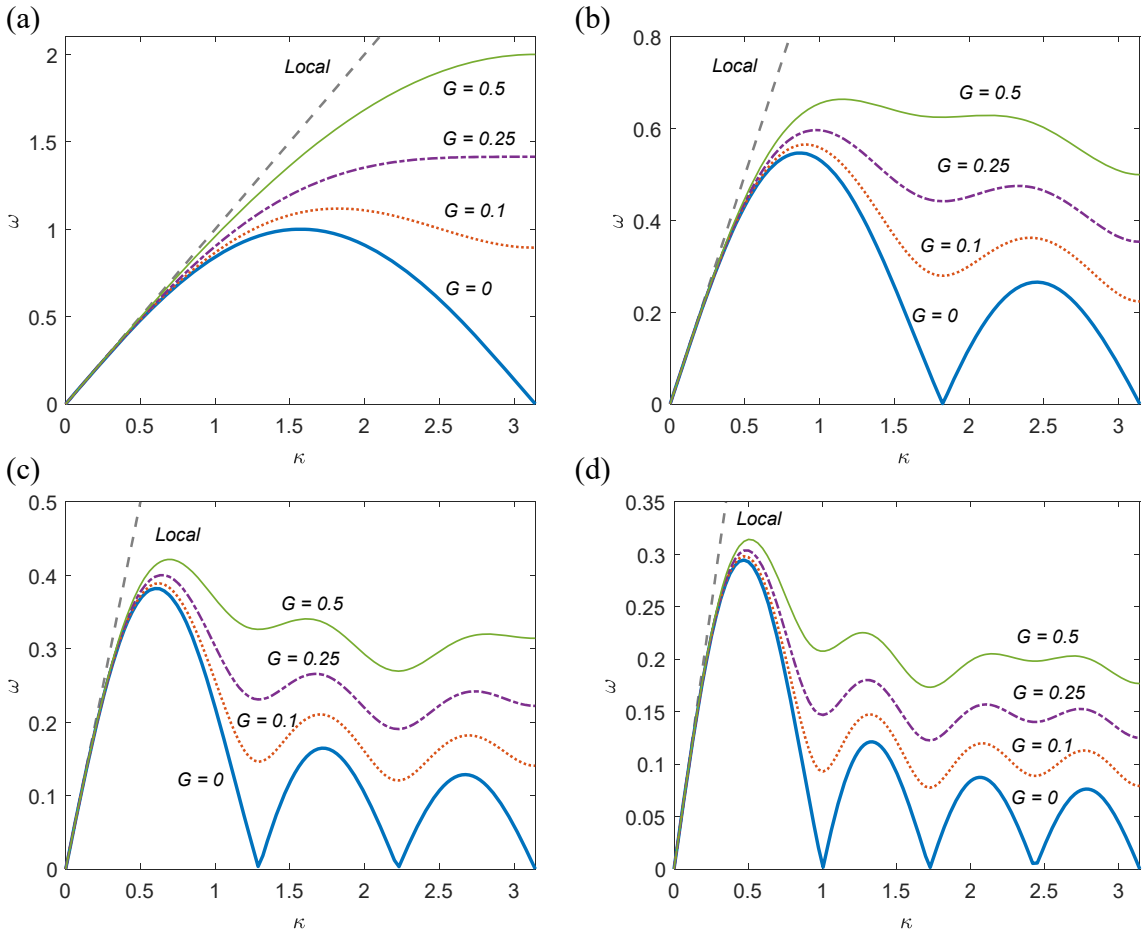


Figure 8.3. Wave dispersion relation for SNOPD in 1D: (a) $n = 1$, (b) $n = 2$, (c) $n = 3$ and (d) $n = 4$

neighborhood case. In addition, we assume small deformation for this analysis and focus mainly on non-ordinary state-based PD and its stabilized version. The 2D dispersion analysis presented in this section is an extension from the previous work by Nicely et al. (Nicely et al., 2018).

First, the shape tensor is given by

$$\boldsymbol{K} = 2(\Delta X)^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{8.30}$$

Similarly, the deformed shape tensor at node $\boldsymbol{x}_{I,J} = \begin{Bmatrix} I \\ J \end{Bmatrix} \Delta X$ is given by

$$\boldsymbol{k} = \Delta X \begin{bmatrix} 2\Delta X - u_{I-1,J} + u_{I+1,J} & -u_{I,J-1} + u_{I,J+1} \\ -v_{I-1,J} + v_{I+1,J} & 2\Delta X - v_{I,J-1} + v_{I,J+1} \end{bmatrix} \tag{8.31}$$

where u and v are displacement components in the x and the y directions, respectively.

Then the deformation gradient is

$$\boldsymbol{F} = \boldsymbol{k} \cdot \boldsymbol{K}^{-1} = \frac{1}{2\Delta X} \begin{bmatrix} 2\Delta X - u_{I-1,J} + u_{I+1,J} & -u_{I,J-1} + u_{I,J+1} \\ -v_{I-1,J} + v_{I+1,J} & 2\Delta X - v_{I,J-1} + v_{I,J+1} \end{bmatrix} \tag{8.32}$$

and the strain under small deformation assumption is

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \frac{-u_{I-1,J}+u_{I+1,J}}{2\Delta X} & \frac{-u_{I,J-1}+u_{I,J+1}-v_{I-1,J}+v_{I+1,J}}{4\Delta X} \\ \frac{-u_{I,J-1}+u_{I,J+1}-v_{I-1,J}+v_{I+1,J}}{4\Delta X} & \frac{-v_{I,J-1}+v_{I,J+1}}{2\Delta X} \end{bmatrix} \tag{8.33}$$

Assuming a plane stress condition and the matrix form constitutive law is given by

$$\boldsymbol{\sigma} = \frac{E}{1-\nu^2} \left((1-\nu)\, \boldsymbol{\varepsilon} + \nu \cdot \mathrm{tr}\left(\boldsymbol{\varepsilon}\right) \boldsymbol{I} \right) \tag{8.34}$$

Substituting Eq. (8.33) into Eq. (8.34) yields

$$\boldsymbol{\sigma} = \begin{bmatrix} \frac{E\left(u_{I-1,J}-u_{I+1,J}+\nu\left(v_{I,J-1}-v_{I,J+1}\right)\right)}{2\Delta X(\nu^2-1)} & -\frac{E\left(u_{I,J-1}-u_{I,J+1}+v_{I-1,J}-v_{I+1,J}\right)}{4\Delta X(1+\nu)} \\ -\frac{E\left(u_{I,J-1}-u_{I,J+1}+v_{I-1,J}-v_{I+1,J}\right)}{4\Delta X(1+\nu)} & \frac{E\left(\nu\left(u_{I-1,J}-u_{I+1,J}\right)+v_{I,J-1}-v_{I,J+1}\right)}{2\Delta X(\nu^2-1)} \end{bmatrix} \tag{8.35}$$

8.3.1 Non-ordinary state-based PD

For $G = 0$ (NOPD), the discretized equation of motion can be obtained as

$$\rho \ddot{\boldsymbol{u}}\left(\boldsymbol{x}_{I,J}\right) = \frac{E}{8(\Delta X)^2\left(\nu^2-1\right)} \begin{Bmatrix} F_x \\ F_y \end{Bmatrix} \tag{8.36}$$

where

$$\begin{aligned} F_x = &-2\left(u_{I-2,J}+u_{I+2,J}\right)+(\nu-1)\left(u_{I,J-2}+u_{I,J+2}\right)-2(\nu-3)\,u_{I,J} \\ &+(1+\nu)\left(-v_{I-1,J-1}+v_{I-1,J+1}+v_{I+1,J-1}-v_{I+1,J+1}\right) \end{aligned} \tag{8.37}$$

and

$$\begin{aligned} F_y = &(1+\nu)\left(-u_{I-1,J-1}+u_{I-1,J+1}+u_{I+1,J-1}-u_{I+1,J+1}\right) \\ &+(\nu-1)\left(v_{I-2,J}+v_{I+2,J}\right)-2\left(v_{I,J-2}+v_{I,J+2}\right)-2(\nu-3)\,v_{I,J} \end{aligned} \tag{8.38}$$

The 2D harmonic wave solution is given by

$$\begin{Bmatrix} u_{I,J} \\ v_{I,J} \end{Bmatrix} = \begin{Bmatrix} \bar{u} \\ \bar{v} \end{Bmatrix} e^{\mathrm{i}(\omega t+\kappa_x I\Delta X+\kappa_y J\Delta X)} \tag{8.39}$$

Substituting Eq. (8.39) into Eq. (8.36) leads to

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{Bmatrix} \bar{u} \\ \bar{v} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{8.40}$$

where the coefficients are

$$C_{11} = \frac{-c^2\left((\nu-3)+2\cos\left(2\kappa_x\Delta X\right)-(\nu-1)\cos\left(2\kappa_y\Delta X\right)\right)}{4(\Delta X)^2\left(\nu^2-1\right)}+\omega^2 \tag{8.41}$$

$$C_{12} = C_{21} = \frac{c^2\sin\left(\kappa_x\Delta X\right)\sin\left(\kappa_y\Delta X\right)}{2(\Delta X)^2(\nu-1)} \tag{8.42}$$

$$C_{22} = \frac{-c^2\left((\nu-3)+2\cos\left(2\kappa_y\Delta X\right)-(\nu-1)\cos\left(2\kappa_x\Delta X\right)\right)}{4(\Delta X)^2\left(\nu^2-1\right)}+\omega^2 \tag{8.43}$$

Further we introduce the incidence angle of the 2D wave, α, such that

$$\begin{Bmatrix} \kappa_x \\ \kappa_y \end{Bmatrix} = \begin{Bmatrix} \kappa\cos\alpha \\ \kappa\sin\alpha \end{Bmatrix} \tag{8.44}$$

The dispersion relation is derived by setting the determinant of the coefficient matrix in Eq. (8.40) equals zero, i.e.

$$C_{11}C_{22}-C_{12}^2 = 0 \tag{8.45}$$

from which we solve for ω and the solutions are given by

$$\omega = \frac{c\sqrt{\sin^2(\Delta X\kappa\cos\alpha) + \sin^2(\Delta X\kappa\sin\alpha)}}{\Delta X}\left\{\begin{array}{c} \frac{1}{\sqrt{1-\nu^2}} \\ \frac{1}{\sqrt{2(1+\nu)}} \end{array}\right\} \tag{8.46}$$

Note that Eq. (8.45) has 4 distinct roots and only the two positive solutions are shown above.

By letting $\Delta X = 1, \nu = 1/4$, and $\alpha = 0$ (unidirectional wave) the wave dispersion relation in Eq. (8.46) is simplified as

$$\omega = c\sqrt{\frac{\sin^2\kappa}{15}}\left\{\begin{array}{c} 4 \\ \sqrt{6} \end{array}\right\} \tag{8.47}$$

8.3.2 Stabilized non-ordinary state-based PD

For $G > 0$ (SNOPD), the extra stabilizing force term is added to the right-hand-side of Eq. (8.36) and the stabilized equation of motion is given by

$$\rho\ddot{\boldsymbol{u}}(\boldsymbol{x}_{I,J}) = \frac{E}{8(\Delta X)^2(\nu^2-1)}\left\{\begin{array}{c} F_x \\ F_y \end{array}\right\} + \frac{3EG}{4\pi(\Delta X)^4(\nu-1)}\left\{\begin{array}{c} S_x \\ S_y \end{array}\right\} \tag{8.48}$$

where

$$\begin{aligned} S_x =& u_{I-2,J} - 4u_{I-1,J} + u_{I,J-2} - 4u_{I,J-1} + 12u_{I,J} \\ & - 4u_{I,J+1} + u_{I,J+2} - 4u_{I+1,J} + u_{I+2,J} \end{aligned} \tag{8.49}$$

and

$$\begin{aligned} S_y =& v_{I-2,J} - 4v_{I-1,J} + v_{I,J-2} - 4v_{I,J-1} + 12v_{I,J} \\ & - 4v_{I,J+1} + v_{I,J+2} - 4v_{I+1,J} + v_{I+2,J} \end{aligned} \tag{8.50}$$

Similarly, the dispersion relation for SNOPD is obtained by substituting the 2D wave solution Eq. (8.39) into Eq. (8.48) and enforcing zero determinant of the coefficient matrix. However, the general expression is very complicated. Here we made simplifications by letting $\Delta X = 1, \nu = 1/4$, and $\alpha = 0$, which leads to

$$\omega = \frac{2}{\sqrt{15\pi}}\left|\sin\frac{\kappa}{2}\right|\left\{\begin{array}{c} \sqrt{8\pi(1+\cos\kappa) + 30G(1-\cos\kappa)} \\ \sqrt{(3\pi(1+\cos\kappa) + 30G(1-\cos\kappa))} \end{array}\right\} \tag{8.51}$$

It can be seen that Eq. (8.51) reduces to Eq. (8.46) when $G = 0$.

8.3.3 Larger horizons

By following the same procedure and with the same simplification parameters, 2D NOPD ($G = 0$) and SNOPD ($G > 0$) dispersion relations for larger horizons ($n = 2, 3, 4$) are given by the following equations:

$$\omega_{\delta=2\Delta X} = \frac{c}{7\sqrt{30\pi}} \left|\sin\frac{\kappa}{2}\right| \left\{ \sqrt{\begin{array}{c} 64\pi\left(29 + 45\cos\kappa + 20\cos 2\kappa + 4\cos 3\kappa\right) \\ +5G\left(203 - 77\cos\kappa - 56\cos 2\kappa - 70\cos 3\kappa\right) \end{array}} \right\} \tag{8.52}$$

$$\omega_{\delta=3\Delta X} = \frac{c}{153\sqrt{35\pi}} \left|\sin\frac{\kappa}{2}\right| \left\{ \begin{array}{l} \sqrt{\begin{array}{l} 1512\pi\left(\begin{array}{l} 502 + 870\cos\kappa + 576\cos 2\kappa \\ +277\cos 3\kappa + 78\cos 4\kappa + 9\cos 5\kappa \end{array}\right) \\ +340G\left(\begin{array}{l} 502 + 122\cos\kappa - 240\cos 2\kappa \\ -267\cos 3\kappa - 58\cos 4\kappa - 59\cos 5\kappa \end{array}\right) \end{array}} \\ \sqrt{\begin{array}{l} 567\pi\left(\begin{array}{l} 502 + 870\cos\kappa + 576\cos 2\kappa \\ +277\cos 3\kappa + 78\cos 4\kappa + 9\cos 5\kappa \end{array}\right) \\ +340G\left(\begin{array}{l} 502 + 122\cos\kappa - 240\cos 2\kappa \\ -267\cos 3\kappa - 58\cos 4\kappa - 59\cos 5\kappa \end{array}\right) \end{array}} \end{array} \right\} \tag{8.53}$$

$$\omega_{\delta=4\Delta X}=\frac{c}{384\sqrt{15\pi}}\left\{\begin{array}{l}\sqrt{128\pi\begin{pmatrix}486+736\cos\kappa+273\cos2\kappa\\-140\cos3\kappa-406\cos4\kappa-476\cos5\kappa\\-337\cos6\kappa-120\cos7\kappa-16\cos8\kappa\end{pmatrix}+15G\begin{pmatrix}4326-1952\cos\kappa-1071\cos2\kappa\\-2060\cos3\kappa+554\cos4\kappa-476\cos5\kappa\\+623\cos6\kappa-120\cos7\kappa+176\cos8\kappa\end{pmatrix}}\\2\left|\sin\frac{\kappa}{2}\right|\sqrt{16\pi\begin{pmatrix}3066+5646\cos\kappa+4424\cos2\kappa\\+2929\cos3\kappa+1574\cos4\kappa+625\cos5\kappa\\+152\cos6\kappa+16\cos7\kappa\end{pmatrix}+5G\begin{pmatrix}3066+1806\cos\kappa-568\cos2\kappa\\-1871\cos3\kappa-1114\cos4\kappa-911\cos5\kappa\\-232\cos6\kappa-176\cos7\kappa\end{pmatrix}}\end{array}\right\} \tag{8.54}$$

Finally, with Eqs. (8.51) to (8.54), we plot the 2D wave dispersion relations by letting $c = 1$ and $G = 0 \sim 0.5$, as shown in Figure 8.4. Similar to the 1D cases, nonzero G values of order 1 effectively eliminated the zero-energy modes in 2D cases as well. Note that the larger dispersion relation from the two positive roots is shown in Figure 8.4 for each horizon radius.

8.4 Summary

In this chapter, the wave dispersion analysis is conducted for discretized PD in both 1D and 2D cases. The dispersion relations for 3D case can also be derived analytically. The intrinsic nonlocality in PD leads to highly nonlinear dispersion relations and dispersive wave propagation behavior at higher wavenumbers. This dispersion analysis paves the way for

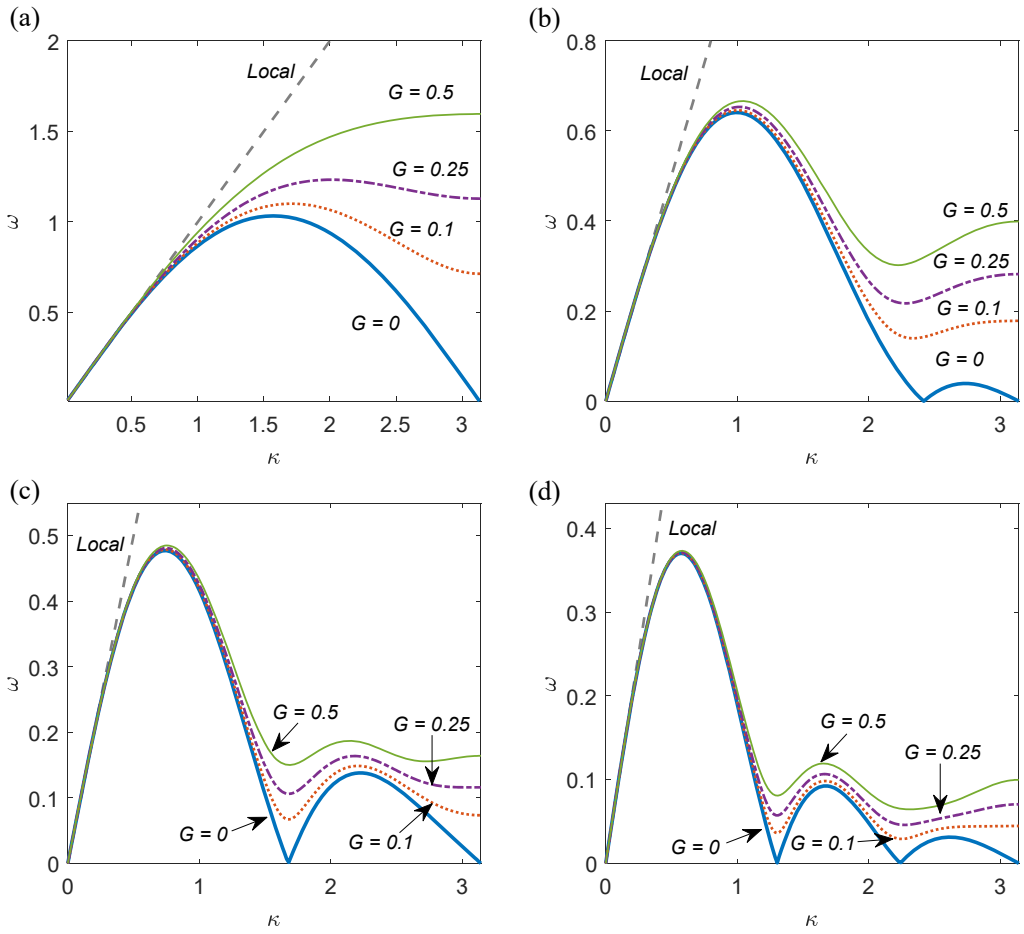


Figure 8.4. Wave dispersion relation for SNOPD in 2D: (a) $n = 1$, (b) $n = 2$, (c) $n = 3$ and (d) $n = 4$

developing a coupling scheme between PD and CCM, which will be discussed in the next chapter.

CHAPTER 9

CONCURRENT MULTISCALE COUPLING METHOD

9.1 Introduction

In this chapter, we establish a concurrent multiscale framework by coupling Peridynamics (PD) with the classical continuum mechanics, e.g. finite element method (FEM). We start from a bridging-scale additive decomposition of the total displacement field. The coarse and fine scale components of the total displacement are represented by FEM and PD, respectively. Then multiscale Lagrangian and equations of motion are established to provide the basis of the proposed computational framework. Since we are interested in simulating the fine scale explicitly only in a small region of the entire domain, numerical boundaries of the PD region are handled by a class of nonlocal matching boundary conditions (NMBC) to eliminate the fine-scale DOFs in the coarse-scale subdomain. The coarse-scale DOFs in the fine-scale region is obtained from a bridging-scale projection of the fine-scale solution onto the coarse-scale basis. Finally, an adaptive scheme is established to dynamically prescribe the local PD region to track the crack propagation.

9.2 Concurrent multiscale framework

9.2.1 Bridging-scale decomposition

First, we decompose the total displacement field $\boldsymbol{u}$ into coarse and fine scales, given by

$$\boldsymbol{u} = \bar{\boldsymbol{u}} + \boldsymbol{u}' \tag{9.1}$$

where $\bar{\boldsymbol{u}}$ and $\boldsymbol{u}'$ represent the coarse-scale and the fine-scale components, respectively. Note that such an additive bridging-scale decomposition has been previously used for coupling *Molecular Dynamics* (MD) with continuum mechanics simulations (Wagner and Liu, 2003;

Qian et al., 2004; Qian and Chirputkar, 2014). In current work, the MD part in the original bridging-scale approach is replaced by PD simulation.

The coarse scale is represented everywhere in the domain using FE interpolation with larger discretizations in both space and time. Therefore, the coarse-scale displacement field is approximated by

$$\bar{\boldsymbol{u}} = \boldsymbol{N}\boldsymbol{d} \tag{9.2}$$

where $\boldsymbol{N}$ is the FE shape function and $\boldsymbol{d}$ is the nodal displacement vector.

The PD region is limited to subdomain where the physics cannot be captured by the coarse scale. In such subdomain, the PD displacement field $\boldsymbol{q}$ is equivalent to the total displacement field $\boldsymbol{u}$. The fine-scale displacement field $\boldsymbol{u}'$ can be obtained by subtracting the coarse-scale component from $\boldsymbol{q}$, i.e.

$$\boldsymbol{u}' = \boldsymbol{q} - \boldsymbol{P}\boldsymbol{q} \tag{9.3}$$

where $\boldsymbol{P}$ is a projection operator defined by

$$\boldsymbol{P} = \boldsymbol{N}\big(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\big)^{-1}\boldsymbol{N}^T\boldsymbol{M}_p \tag{9.4}$$

in which $\boldsymbol{M}_p$ is the diagonal mass matrix in PD region.

Also, we define the complimentary projector $\boldsymbol{Q}$ as

$$\boldsymbol{Q} = \boldsymbol{I} - \boldsymbol{P} \tag{9.5}$$

where $\boldsymbol{I}$ is the identity matrix with the appropriate size.

Eq. (9.3) can be equivalently represented by

$$\boldsymbol{u}' = \boldsymbol{Q}\boldsymbol{q} \tag{9.6}$$

Substituting Eqs. (9.2) and (9.6) into (9.1), the final two-scale decomposition is given by

$$\boldsymbol{u} = \boldsymbol{N}\boldsymbol{d} + \boldsymbol{Q}\boldsymbol{q} \tag{9.7}$$

Remarks on some properties of $\boldsymbol{P}$ and $\boldsymbol{Q}$:

a) $\boldsymbol{PP} = \boldsymbol{P}$ as required for a projection operation, which can be shown by

$$\begin{aligned}\boldsymbol{PP} &= \boldsymbol{N}\left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\boldsymbol{M}_p \\ &= \boldsymbol{N}\left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)^{-1}\left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)\left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\boldsymbol{M}_p \\ &= \boldsymbol{N}\left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\boldsymbol{M}_p \\ &= \boldsymbol{P}\end{aligned} \tag{9.8}$$

Similarly, $\boldsymbol{QQ} = (\boldsymbol{I}-\boldsymbol{P})(\boldsymbol{I}-\boldsymbol{P}) = \boldsymbol{I} - 2\boldsymbol{P} + \boldsymbol{PP} = \boldsymbol{Q}$.

b) $\boldsymbol{Pu'} = \boldsymbol{P}(\boldsymbol{q}-\boldsymbol{Pq}) = \boldsymbol{0}$ which shows that the projection of fine-scale displacement field onto coarse-scale basis is zero.

c) Last, we can show that

$$\begin{aligned}\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{Q} &= \boldsymbol{N}^T\boldsymbol{M}_p\left(\boldsymbol{I} - \boldsymbol{N}\left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\boldsymbol{M}_p\right) \\ &= \boldsymbol{N}^T\boldsymbol{M}_p - \boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\boldsymbol{M}_p \\ &= \boldsymbol{0}\end{aligned} \tag{9.9}$$

which can be used to decouple the kinetic energy in the Lagrangian equation.

9.2.2 Multiscale Lagrangian and equations of motion

In the concurrent multiscale framework, PD represents the fine scale. Therefore, derivation of multiscale equations of motion starts from the PD Lagrangian (Madenci and Oterkus, 2014). Note that we have $\boldsymbol{q} = \boldsymbol{u}$ if PD is applied to the entire domain.

First, the total kinetic energy T in the discretized form is given by

$$T = \sum_{i=1}^{\infty}\frac{1}{2}\rho_{(i)}\dot{\boldsymbol{u}}_{(i)}\cdot\dot{\boldsymbol{u}}_{(i)}V_{(i)} \tag{9.10}$$

where $\rho_{(i)}$ and $V_{(i)}$ are mass density and volume of the i-th node, respectively.

In the absence of external force for simplicity, the total potential energy U is given by

$$U = \sum_{i=1}^{\infty}W_{(i)}V_{(i)} \tag{9.11}$$

where the strain energy density, W, is given by

$$W_{(k)} = \frac{1}{2}\sum_{j=1}^{\infty}\frac{1}{2}\left(w_{(k)(j)} + w_{(j)(k)}\right)V_{(j)} \tag{9.12}$$

in which the scalar-valued micro-potentials are given by

$$w_{(k)(j)} = w_{(k)(j)}\left(\boldsymbol{y}_{\left(1^k\right)} - \boldsymbol{y}_{(k)}, \boldsymbol{y}_{\left(2^k\right)} - \boldsymbol{y}_{(k)}, \ldots\right) \tag{9.13}$$

and

$$w_{(j)(k)} = w_{(j)(k)}\left(\boldsymbol{y}_{(1^j)} - \boldsymbol{y}_{(j)}, \boldsymbol{y}_{(2^j)} - \boldsymbol{y}_{(j)}, \ldots\right) \tag{9.14}$$

where $\boldsymbol{y}_{(k)}$ is the position vector of point $\boldsymbol{x}_{(k)}$ in the deformed configuration and $\boldsymbol{y}_{\left(1^k\right)}$ is the position vector of the 1st material point that interacts with point $\boldsymbol{x}_{(k)}$.

Using Eqs. (9.12) to (9.14), the potential energy Eq. (9.11) can be rewritten as

$$U = \sum_{i=1}^{\infty}\left[\frac{1}{2}\sum_{j=1}^{\infty}\frac{1}{2}\left(w_{(i)(j)} + w_{(j)(i)}\right)V_{(j)}\right]V_{(i)} \tag{9.15}$$

The total Lagrangian L can now be written as

$$L = T - U = \sum_{i=1}^{\infty}\frac{1}{2}\rho_{(i)}\dot{\boldsymbol{u}}_{(i)}\cdot\dot{\boldsymbol{u}}_{(i)}V_{(i)} - \sum_{i=1}^{\infty}\left[\frac{1}{2}\sum_{j=1}^{\infty}\frac{1}{2}\left(w_{(i)(j)} + w_{(j)(i)}\right)V_{(j)}\right]V_{(i)} \tag{9.16}$$

or using matrix expression, it can be simplified as

$$L = \frac{1}{2}\dot{\boldsymbol{u}}^T\boldsymbol{M}_p\dot{\boldsymbol{u}} - U\left(\boldsymbol{u}\right) \tag{9.17}$$

where $\boldsymbol{M}_p$ is the diagonal mass matrix defined by

$$\boldsymbol{M}_p = \begin{bmatrix} \rho_{(1)}V_{(1)}\boldsymbol{I} & & \\ & \rho_{(2)}V_{(2)}\boldsymbol{I} & \\ & & \ddots \end{bmatrix} \tag{9.18}$$

Using the multiscale decomposition introduced in Eq. (9.7), the total Lagrangian can be rewritten as

$$L = \frac{1}{2}\dot{\boldsymbol{d}}^T\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\dot{\boldsymbol{d}} + \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{Q}^T\boldsymbol{M}_p\boldsymbol{Q}\dot{\boldsymbol{q}} - U\left(\boldsymbol{N}\boldsymbol{d} + \boldsymbol{Q}\boldsymbol{q}\right) \tag{9.19}$$

Note that we used Eq. (9.9) to decouple the kinetic energy, i.e. the cross-terms are vanished.

Now, we define

$$\boldsymbol{M} = \boldsymbol{N}^T \boldsymbol{M}_p \boldsymbol{N} \tag{9.20}$$

and

$$\boldsymbol{\mathcal{M}} = \boldsymbol{Q}^T \boldsymbol{M}_p \boldsymbol{Q} = \boldsymbol{M}_p \boldsymbol{Q} = \boldsymbol{Q}^T \boldsymbol{M}_p \tag{9.21}$$

in which

$$\boldsymbol{Q}^T \boldsymbol{M}_p = \left(\boldsymbol{I} - \boldsymbol{M}_p \boldsymbol{N} \boldsymbol{M}^{-1} \boldsymbol{N}^T\right) \boldsymbol{M}_p = \boldsymbol{M}_p \left(\boldsymbol{I} - \boldsymbol{N} \boldsymbol{M}^{-1} \boldsymbol{N}^T \boldsymbol{M}_p\right) = \boldsymbol{M}_p \boldsymbol{Q} \tag{9.22}$$

and

$$\boldsymbol{Q}^T \boldsymbol{M}_p \boldsymbol{Q} = \boldsymbol{Q}^T \boldsymbol{Q}^T \boldsymbol{M}_p = \boldsymbol{M}_p \boldsymbol{Q} \boldsymbol{Q} = \boldsymbol{M}_p \boldsymbol{Q} \tag{9.23}$$

Finally, the multiscale Lagrangian is given by

$$L\left(\dot{\boldsymbol{d}}, \boldsymbol{d}; \dot{\boldsymbol{q}}, \boldsymbol{q}\right) = \frac{1}{2} \dot{\boldsymbol{d}}^T \boldsymbol{M} \dot{\boldsymbol{d}} + \frac{1}{2} \dot{\boldsymbol{q}}^T \boldsymbol{\mathcal{M}} \dot{\boldsymbol{q}} - U\left(\boldsymbol{N}\boldsymbol{d} + \boldsymbol{Q}\boldsymbol{q}\right) \tag{9.24}$$

The multiscale equations of motion are derived from the multiscale Lagrangian according to the Euler-Lagrange equations:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{d}}}\right) - \frac{\partial L}{\partial \boldsymbol{d}} = 0 \tag{9.25}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L}{\partial \boldsymbol{q}} = 0 \tag{9.26}$$

Therefore, we have

$$\boldsymbol{M} \ddot{\boldsymbol{d}} = -\frac{\partial U}{\partial \boldsymbol{d}} \tag{9.27}$$

$$\boldsymbol{\mathcal{M}} \ddot{\boldsymbol{q}} = -\frac{\partial U}{\partial \boldsymbol{q}} \tag{9.28}$$

Using chain rule, we obtain the following multiscale equations of motion

$$\boldsymbol{M}\ddot{\boldsymbol{d}} = -\left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{d}}\right)^T \frac{\partial U}{\partial \boldsymbol{u}} = \boldsymbol{N}^T \boldsymbol{f} \tag{9.29}$$

$$\boldsymbol{\mathcal{M}}\ddot{\boldsymbol{q}} = -\left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{q}}\right)^T \frac{\partial U}{\partial \boldsymbol{u}} = \boldsymbol{Q}^T \boldsymbol{f} \tag{9.30}$$

in which

$$\boldsymbol{f} := -\frac{\partial U}{\partial \boldsymbol{u}} \tag{9.31}$$

Using Eq. (9.21), Eq. (9.30) can also be rewritten as

$$\boldsymbol{M}_p \ddot{\boldsymbol{q}} = \boldsymbol{f} \tag{9.32}$$

which is exactly the PD equation of motion.

From the original Lagrangian Eq. (9.17), the total displacement field $\boldsymbol{u}$ satisfies

$$\boldsymbol{M}_p \ddot{\boldsymbol{u}} = \boldsymbol{f} \tag{9.33}$$

which, again, shows that the PD displacement field $\boldsymbol{q}$ is equivalent to the total displacement field $\boldsymbol{u}$. It also indicates that Eq. (9.29) is redundant in the subdomain where PD and FEM coexist since the coarse-scale solution is the projection of fine scale, i.e. $\boldsymbol{N}\boldsymbol{d} = \boldsymbol{P}\boldsymbol{q}$.

9.2.3 Computational domain partition

As we mentioned before, the computational domain is partitioned into two non-overlapping parts: (1) the subdomain where FEM and PD coexists and (2) the rest where only FEM presents, as shown in Figure 9.1. Based on this partition, we can rewrite the total displacement field $\boldsymbol{u}$ as

$$\boldsymbol{u} = \begin{cases} \boldsymbol{u}_1 = \bar{\boldsymbol{u}}_1 + \boldsymbol{u}_1' & \text{in the overlapping region} \\ \boldsymbol{u}_2 = \bar{\boldsymbol{u}}_2 + \boldsymbol{u}_2' & \text{otherwise} \end{cases} \tag{9.34}$$

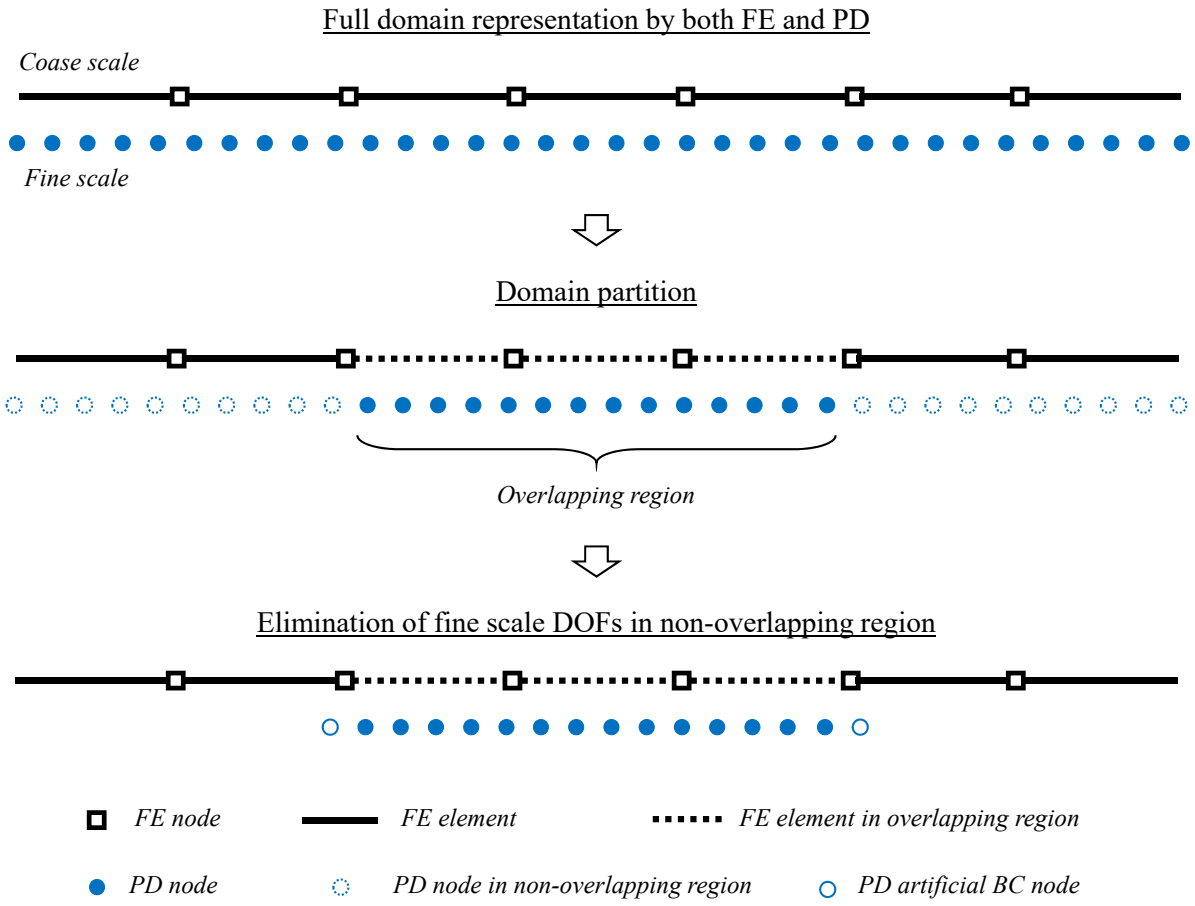


Figure 9.1. Schematic illustration of the concurrent multiscale framework in 1D

where the subscript 1 refers to the subdomain where PD and FEM coexists and subscript 2 refers to the FEM (coarse scale) only subdomain.

In Eq. (9.34), the coarse-scale components $\bar{\boldsymbol{u}}_1$ and $\bar{\boldsymbol{u}}_2$ can be obtained by solving the FE equation of motion Eq. (9.29) and note that the former can also be obtained by the projection of PD solution of Eq. (9.2.2), i.e. $\bar{\boldsymbol{u}}_1 = \boldsymbol{P}\boldsymbol{q}_1 = \boldsymbol{P}\boldsymbol{u}_1$. The fine-scale component $\boldsymbol{u}'_1$ can be obtained from PD solution using projection $\boldsymbol{u}'_1 = \boldsymbol{Q}\boldsymbol{q}_1$. The fine-scale component in FE only region, $\boldsymbol{u}'_2$, cannot be obtained from solving either FE or PD equation of motion thus must be approximated.

We assumed that in the FE-only region the unknown field is smooth enough to be captured by the coarse-scale shape functions, i.e. $\boldsymbol{u}_2 \approx \bar{\boldsymbol{u}}_2 = \boldsymbol{N}\boldsymbol{d}_2$. Thus, we implicitly indicated that $\boldsymbol{u}_2' = \boldsymbol{0}$. However, at the PD boundaries (PD/FE interfaces) the effect of $\boldsymbol{u}_2'$ cannot be ignored. According to (Wagner and Liu, 2003), one possible method to account for $\boldsymbol{u}_2'$ at the numerical interfaces is a Langevin type of *time history kernel* (THK) approach, which is briefly introduced in Appendix E. However, computational implementation of THK-based interfacial boundary condition is not an easy task for general cases. In this work, we employ a simple but yet effective method to eliminate the fine-scale PD nodes in the FE-only region, as shown in Figure 9.1. This method is referred to as the *matching boundary conditions* (MBCs), which is originally proposed as the so-called absorbing or non-reflective boundary conditions for lattice dynamics in MD (Wang and Tang, 2013). The MBCs for PD is developed in the next section.

9.3 Matching Boundary Conditions

9.3.1 MBCs for Molecular Dynamics

MBCs are first proposed by Tang (Tang, 2008) and Wang and Tang (Wang and Tang, 2010, 2013) as linear constraints of displacement and velocity at the boundary atoms to suppress wave reflections. MBCs is conceptually illustrated in 1D as shown in Figure 9.2. Here we consider an infinitely long 1D chain of atoms and we are only interested in explicit MD simulation of the nodes represented by solid circles. For the rest of the infinite number of atoms, their motions are collectively mimicked by a small number of MBC nodes such that computational cost is saved.

Motions of these boundary nodes in the reduced MD simulation are governed by the following MBC:

$$\sum_{j=0}^{N} c_j \dot{u}_j - \sum_{j=0}^{N} b_j u_j = 0 \tag{9.35}$$

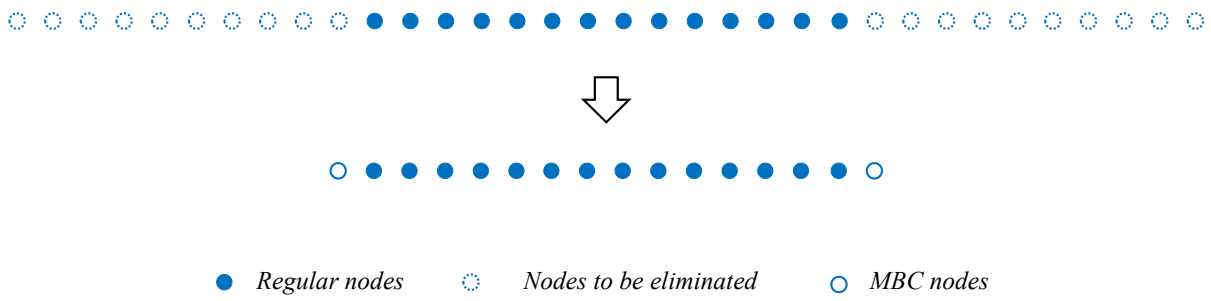


Figure 9.2. Conceptual illustration of matching boundary conditions in 1D

Figure 9.3. Numbering of atoms in 1D MBC

where c and b are unknown coefficients to be solved by matching dispersion relation near the boundary with a minimal number of atoms. N is the order of the MBC and the numbering of atoms are shown in Figure 9.3. The N-th order MBC is simply denoted as MBCN.

9.3.2 MBC for Peridynamics

The same idea of MBCs can be borrowed in PD for the purpose of developing concurrent multiscale simulation. The corresponding MBCs for PD is recently developed by Nicely and co-workers (Nicely et al., 2018; Nicely, 2018). Similar work for 1D wave propagation problems has been done by Wang et al. (Wang et al., 2017) and termed as the Peridynamic transmiting boundary conditions (PTBCs). Here we follow the former to derive the MBCs for PD.

A monochromatic harmonic wave solution in 1D for the I-th node is given by

$$u_I = \bar{u} e^{\mathrm{i}(\omega t + \kappa I \Delta X)} \tag{9.36}$$

where $\bar{u}$ is the wave amplitude, $\mathrm{i} = \sqrt{-1}$, ω is the angular frequency and κ is the wavenumber.

Substituting Eq. (9.36) into Eq. (9.35) yields

$$\Delta(\kappa) \equiv \mathrm{i}\,\omega(\kappa) \sum_{j=0}^{N} c_j e^{\mathrm{i}(\kappa j \Delta X)} - \sum_{j=0}^{N} b_j e^{\mathrm{i}(\kappa j \Delta X)}, \qquad \kappa \in [0, \pi] \tag{9.37}$$

Note that the wave dispersion relation $\omega(\kappa)$ for PD is derived in Chapter 8.

The residual, $\Delta(\kappa)$ as defined in Eq. (9.37), measures the deviation of MBCs for the exact outgoing wave solution given by Eq. (9.36). A zero residual $\Delta(\kappa) = 0$ indicates a perfectly matching dispersion relation at the wavenumber of κ near the boundary and automatically leads to a non-reflective numerical interface. However, in general, $\Delta(\kappa) \neq 0$ for all κ. The condition can be relaxed by enforcing it and its derivatives at particular wavenumbers. This leads to two types of MBCs. The first one is called the *Taylor type* MBC, which focuses on the long wave limit, i.e. $\kappa \to 0^+$, that dominates the energy in many cases. The second type is termed as the *Taylor-Newton type* MBC to deal additional wavenumbers.

In this work, we focus on the Taylor type MBC, in which the unknown coefficients are solved by enforcing

$$\left.\frac{d^n \Delta(\kappa)}{d\kappa^n}\right|_{\kappa=0} = 0, \qquad \forall\ n = 0, 1, 2, 3, \ldots, 2N \tag{9.38}$$

Taylor expansion around $\kappa = 0$ for the terms in Eq. (9.37) are

$$\mathrm{i}\omega(\kappa) = \sum_{n=0}^{\infty} a_n (\mathrm{i}\kappa)^n, \qquad \text{with } a_{2m} = 0,\ a_{2m+1} = \frac{\omega^{(2m+1)}(\kappa)}{(2m+1)!},\ m = 0, 1, 2, \ldots \tag{9.39}$$

and

$$e^{j\Delta X(\mathrm{i}\kappa)} = \sum_{n=0}^{\infty} h_{nj} (\mathrm{i}\kappa)^n, \qquad \text{with } h_{nj} = \frac{(j\Delta X)^n}{n!} \tag{9.40}$$

Substituting these terms into Eq. (9.38) leads to the following linear algebraic system of order $2N + 2$:

$$\begin{bmatrix} \boldsymbol{e}_1^T & \boldsymbol{z}^T \\ \boldsymbol{AH} & -\boldsymbol{H} \end{bmatrix} \begin{Bmatrix} \boldsymbol{C} \\ \boldsymbol{B} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{e}_1^T \\ \boldsymbol{z}^T \end{Bmatrix} \tag{9.41}$$

where

$$\boldsymbol{e}_1 = \begin{Bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{Bmatrix}, \boldsymbol{z} = \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix}, \boldsymbol{C} = \begin{Bmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{Bmatrix}, \boldsymbol{B} = \begin{Bmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{Bmatrix} \tag{9.42}$$

$$\boldsymbol{A} = \begin{bmatrix} a_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ a_{2N} & a_{2N-1} & \cdots & a_0 \end{bmatrix}, \ \boldsymbol{H} = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0N} \\ h_{10} & h_{11} & \ddots & h_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ h_{2N\,0} & h_{2N\,1} & \cdots & h_{2N\,N} \end{bmatrix} \tag{9.43}$$

The unknown coefficients in MBCs are solved from Eq. (9.41).

As an example, we let $N = 1$ and Eq. (9.43) is simplified as

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 \\ a_1 & 0 & 0 \\ 0 & a_1 & 0 \end{bmatrix}, \ \boldsymbol{H} = \begin{bmatrix} 1 & 1 \\ 0 & \Delta X \\ 0 & (\Delta X)^2/2 \end{bmatrix} \tag{9.44}$$

Substituting Eq. (9.44) into Eq. (9.41) yields

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ a_1 & a_1 & 0 & -\Delta X \\ 0 & a_1 \Delta X & 0 & -(\Delta X)^2/2 \end{bmatrix} \begin{Bmatrix} c_0 \\ c_1 \\ b_0 \\ b_1 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{9.45}$$

from which we find the solution to be

$$c_0 = c_1 = 1, \ b_0 = -b_1 = -\frac{2a_1}{\Delta X} \tag{9.46}$$

It is shown in (Nicely, 2018) that $a_1 = c = \sqrt{E/\rho}$ for all the PD formulations introduced in Chapter 7 with an arbitrary horizon radius. Therefore, the MBC1 for PD is given by

$$\dot{u}_0 + \dot{u}_1 = \frac{2c}{\Delta X}(-u_0 + u_1) \tag{9.47}$$

Table 9.1. MBCN coefficients for $N = 1 \sim 3$

N	c_0	c_1	c_2	c_3	b_0	b_1	b_2	b_3
1	1	1			I_0	$-I_0$		
2	1	I_1	1		I_2	0	$-I_2$	
3	1	I_3	I_3	1	I_4	I_5	$-I_5$	$-I_4$

which is a very simple boundary condition compared to the time history kernel approach.

For higher-order MBCs, the coefficients are solved in (Nicely, 2018) and summarized here in Table 9.1. The constants in Table 9.1 are given by

$$I_0 = -\frac{2a_1}{\Delta X} \tag{9.48}$$

$$I_1 = \frac{4\left(a_1(\Delta X)^2 + 3a_3\right)}{a_1(\Delta X)^2 - 6a_3} \tag{9.49}$$

$$I_2 = -\frac{3a_1^2\Delta X}{a_1(\Delta X)^2 - 6a_3} \tag{9.50}$$

$$I_3 = \frac{3\left(3a_1(\Delta X)^4 + 40\left(a_3(\Delta X)^2 + a_5\right)\right)}{a_1(\Delta X)^4 - 120a_5} \tag{9.51}$$

$$I_4 = \frac{720\left({a_3}^2 - a_1a_5\right) + 120a_1a_3(\Delta X)^2 + 11a_1^2(\Delta X)^4}{360a_5\Delta X - 3a_1(\Delta X)^5} \tag{9.52}$$

$$I_5 = \frac{3\left(-240a_3^2 + 240a_1a_5 + 40a_1a_3(\Delta X)^2 + 3{a_1}^2(\Delta X)^4\right)}{\Delta X\left(120a_5 - a_1(\Delta X)^4\right)} \tag{9.53}$$

In the above constants, the Taylor coefficients (Eq. (9.39)) of bond-based PD (BBPD) dispersion relation are given in Table 9.2 for horizon radius $\delta = 1\Delta X, 2\Delta X, 3\Delta X$.

For non-ordinary state-based PD (NOPD) with correspondence material, the Taylor coefficients for an arbitrary horizon radius $\delta = n\Delta X$ are given by

$$\begin{aligned} a_1 &= c \\ a_3 &= \frac{c}{30}(\Delta X)^2\left(-1 + 3n + 3n^2\right) \\ a_5 &= \frac{c}{840}(\Delta X)^4\left(1 - 3n + 6n^3 + 3n^4\right) \end{aligned} \tag{9.54}$$

Table 9.2. Coefficients of Taylor series expansion of bond-based PD dispersion relation

δ	a_1	a_3	a_5
$1\Delta X$	c	$c\frac{(\Delta X)^2}{24}$	$c\frac{(\Delta X)^4}{1920}$
$2\Delta X$	c	$c\frac{5(\Delta X)^2}{48}$	$c\frac{49(\Delta X)^4}{7680}$
$3\Delta X$	c	$c\frac{5(\Delta X)^2}{24}$	$c\frac{449(\Delta X)^2}{17280}$

For the stabilized NOPD (SNOPD), we have

$$
\begin{aligned}
a_1 &= c \\
a_3 &= -\frac{c\left(-1+3n\left(1+n\right)\right)\Delta X\left(G\left(1+n\right)\left(1+2n\right)-4n^3\Delta X\right)}{120n^3} \\
a_5 &= \frac{c(\Delta X)^2\begin{pmatrix} -7G^2\left(-1+10n^2+15n^3+6n^4\right)^2 \\ -8G\left(-1+n\right)n^3\left(1+n\right)\left(2+n\right)\left(-1+2n\right)\left(1+2n\right)\left(3+2n\right)\Delta X \\ +240n^6\left(1+3n\left(1+n\right)\left(-1+n+n^2\right)\right)(\Delta X)^2 \end{pmatrix}}{201600n^6}
\end{aligned}
\tag{9.55}
$$

where G is the stabilization parameter.

Remarks on MBCs with $N > 1$:

It is shown that the MBC coefficients for non-ordinary state-based PD are singular when $N \geqslant 2n$ (Nicely, 2018). Therefore, all higher-order MBC coefficients are singular for the case of nearest neighborhood ($n = 1$). For larger neighborhoods ($1 < n < 2N$), solutions of Eq. (9.41) do exist but not stable due to negative coefficient c_i. An exception is the stabilized version. Proper choice of the stabilization parameter G can lead to stable higher-order MBCs with arbitrary horizon radius. The stable range of G is constrained by two conditions: (1) coefficient c_i must be positive and (2) reflection coefficient should be less than one (Nicely, 2018). These constraints lead to the fact that the applicable range of G depends on both the order of MBC as well as the horizon radius, which could be problematic in practical applications. Therefore, we employ only MBC1 in this work.

Figure 9.4. Nodes numbering for nonlocal MBCs in 1D

9.3.3 Nonlocal MBC for Peridynamics

The MBCs developed above was shown to be ineffective in terms eliminating artificial wave reflections at the boundaries, which is attributed to the so-called edge effect (Nicely et al., 2018; Nicely, 2018).

To eliminate the edge effect, the MBCs is recast in a nonlocal form given by

$$\sum_{j=0}^{N} c_j \dot{u}_{j-M} - \sum_{j=0}^{N} b_j u_{j-M} = 0 \qquad \text{for } M = 0, ..., 2n-2, 2n-1 \tag{9.56}$$

which is termed as the NMBCs. In NMBCs, an extra index is introduced to expand the boundary domain to be twice the size of the horizon radius as illustrated in Figure 9.4. It is also shown that the coefficients in Eq. (9.56) are independent of M and are the same with the MBCs.

In this work, the revised NMBCs are employed for the concurrent multiscale framework. For 2D cases, the NMBCs are constructed by using the products of the 1D NMBCs at selected directions of incident waves (Wang and Tang, 2013). It is shown in (Nicely et al., 2018; Nicely, 2018) that for both unidirectional and multidirectional waves, the coefficients for 2D NMBC1 are given by

$$\boldsymbol{C}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \boldsymbol{C}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{9.57}$$

and

$$\boldsymbol{B}_0 = -\frac{8c}{\sqrt{15}\Delta X} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \boldsymbol{B}_1 = \frac{8c}{\sqrt{15}\Delta X} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{9.58}$$

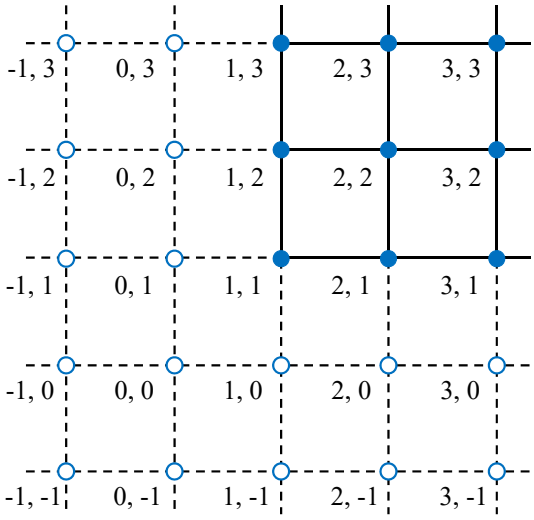


Figure 9.5. Nodes numbering for NMBCs in 2D

which lead to the following 2D NMBC1 for different PD formulations with an arbitrary horizon radius:

$$\dot{\boldsymbol{u}}_{i-M,j} = \left\{ \begin{array}{c} \dot{u}_{i-M,j} \\ \dot{v}_{i-M,j} \end{array} \right\} = \left\{ \begin{array}{c} -\dot{u}_{i-M+1,j} + \frac{c}{\Delta X}\frac{8}{\sqrt{15}}\left(-u_{i-M,j} + u_{i-M+1,j}\right) \\ -\dot{v}_{i-M+1,j} + \frac{c}{\Delta X}\frac{8}{\sqrt{15}}\left(-v_{i-M,j} + v_{i-M+1,j}\right) \end{array} \right\} \tag{9.59}$$

Note that the above equation is for the left boundary domain in PD.

An example of the NMBC nodes numbering in 2D is illustrated in Figure 9.5, where the bottom left corner of a PD boundary domain is shown. In this figure, solid circles represent internal PD nodes while as the empty ones indicate NMBC nodes.

9.3.4 Two-way NMBC

The above developed NMBCs provided an effective mechanism to eliminate artificial wave reflections at the numerical interface. The waves originated form PD region, which are referred to as the outgoing waves, are transmitted at the boundaries by the above one-way NMBCs. For the concurrent multiscale framework, a two-way transmitting numerical interface is required, i.e., the incoming waves, or equivalently the waves originated from the

coarse-scale FE-only region need to be transmitted into the fine-scale PD region as well. A two-way NMBC is developed based on the recent work by (Wang et al., 2017).

In 1D cases, displacement and velocity are decomposed by

$$u = u^{\text{In}} + u^{\text{Out}}, \quad \dot{u} = \dot{u}^{\text{In}} + \dot{u}^{\text{Out}} \tag{9.60}$$

and the outgoing components are treated by one-way NMBCs.

Substituting Eq. (9.60) into Eq. (9.56) yields

$$\sum_{j=0}^{N} c_j \left(\dot{u}_{j-M} - \dot{u}_{j-M}^{\text{In}} \right) - \sum_{j=0}^{N} b_j \left(u_{j-M} - u_{j-M}^{\text{In}} \right) = 0 \tag{9.61}$$

The two-way NMBC is rewritten by

$$c_0 \dot{u}_{-M} = -\sum_{j=1}^{N} c_j \dot{u}_{j-M} + \sum_{j=0}^{N} b_j u_{j-M} + \left(\sum_{j=0}^{N} c_j \dot{u}_{j-M}^{\text{In}} - \sum_{j=0}^{N} b_j u_{j-M}^{\text{In}} \right) \tag{9.62}$$

where the incoming displacement and velocity are obtained from interpolations of the coarse-scale solution.

9.4 Bridging scale projection

In the second section of this chapter, we showed that the FE equations in the overlapping region is redundant and the coarse-scale solutions can be obtained from the projection of the fine-scale PD solutions. Although the projection operator is already given in Eq. (9.4), here we briefly show its derivation.

In the overlapping region, the PD nodal values can also be interpolated using the FE nodal values and the shape functions, which is given by

$$\hat{\boldsymbol{q}} = \sum_{I} \boldsymbol{N}_I \boldsymbol{d}_I \tag{9.63}$$

We can further define a scalar error between the PD solution and the above interpolated values in a quadratic form:

$$e = (\boldsymbol{q} - \hat{\boldsymbol{q}})^T \boldsymbol{M}_p (\boldsymbol{q} - \hat{\boldsymbol{q}}) \tag{9.64}$$

where we choose the fine-scale diagonal mass matrix as the weight matrix.

The projection operator is obtained by minimizing the error defined in Eq. (9.64), which leads to

$$\boldsymbol{N}\boldsymbol{d} = \boldsymbol{N}\left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{q} = \boldsymbol{P}\boldsymbol{q} \tag{9.65}$$

Therefore, the bridging-scale projection (BSP) gives the coarse-scale nodal values in the overlapping region using fine-scale solution as follows

$$\boldsymbol{d} = \left(\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\boldsymbol{M}_p\boldsymbol{q} \tag{9.66}$$

which essentially serves as a homogenization mechanism from fine scale to coarse scale.

9.5 Adaptivity algorithms

In dynamic fracture simulations, the fine-scale PD region is initially placed at the vicinity of the pre-existing crack tip. As the crack propagates under loading, the local PD region needs to be adaptively relocated to the vicinity of the new crack tip and all variables involved in the coupling scheme should be updated appropriately. In this section, we propose such adaptivity algorithms for concurrent multiscale dynamic fracture simulation. For simplicity, we consider only one initial crack in this section, however, the proposed algorithms are applicable for general cases where no or multiple initial cracks exist.

9.5.1 Crack tracking algorithm

First, we need to identify the crack tip in the PD region and keep tracking of the crack path during the entire simulation. The crack tip location is used to activate the trigger for the PD region relocation and the crack path is employed to enrich the coarse-scale finite element. In PD, the crack occurs autonomously in the form of bond breaking without any crack initiation/propagation criteria. Hence, there is no variable in PD to keep tracking of the crack path. Two simple algorithms are proposed here to identify the location of the crack tip.

9.5.1.1 Algorithm 1: nearest neighbor bond

The first crack tip tracking algorithm determines the location of the crack tip based on the broken bonds and the associated nodal damage variables. Note that the damage variable in PD is introduced in Section 7.3.4 and defined as the percentage of broken bonds associated with a node.

Instead of pinpointing the exact location of the crack tip, which is not an easy task by itself, we employ an approximated crack tip location in this multiscale framework. In this algorithm, we define the bond between immediately adjacent nodes (where bond length $= \Delta X$) as the *nearest neighbor bond* (NNB). The potential locations for the approximated crack tip are the geometrical centers of these NNBs. Once a NNB is broken and the associated two nodal damage variables are both greater than a threshold value, e.g. $D_{\text{threshold}} = 0.3$, the NNB is considered as a crack tip candidate. There might be multiple crack tip candidates at a simulation time step, but most of them are fictitious. In order to determine the physically reasonable new crack tip, two criteria are considered here. First, the distance between the new and previous crack tips is limited to an upper bound threshold value, e.g. the horizon radius δ, to avoid sudden jumping of crack tip and to enforce continuity of the crack path. Second, the angle between the new crack path increment directional vector and the previous crack path directional vector is limited to a specified range, e.g. $\pm\pi/2$, to avoid sharp change in crack propagation direction.

The above algorithm is schematically illustrated in Figure 9.6. In this figure, PD nodes are represented by solid circles while the centers of NNBs, i.e. the potential crack tip locations, are indicated by empty squares. Point **a** represents the previous crack tip and points **b** to **d** represent the broken NNBs with nodal damage values greater than $D_{\text{threshold}}$, i.e. the new crack tip candidates. Among these candidate points, **b** violates the first criterion where the valid distance to previous crack tip is indicated by the shaded circle centered at point **a** as

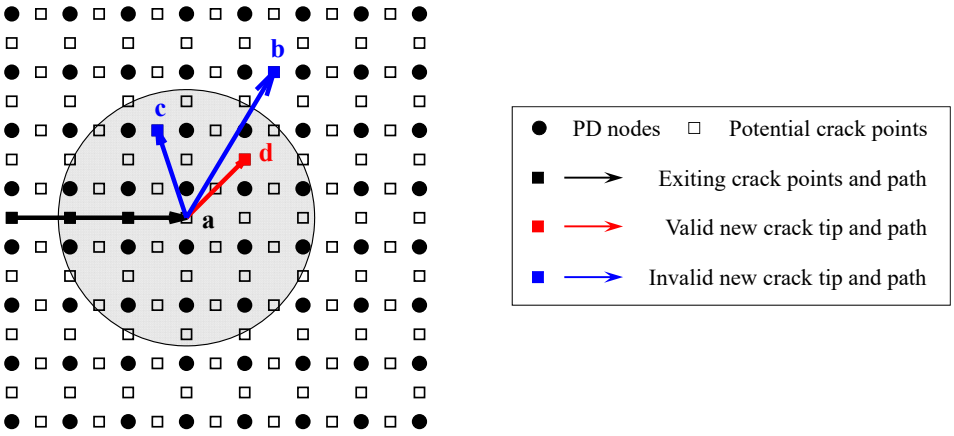


Figure 9.6. Schematic illustration of the crack tip tracking algorithm 1

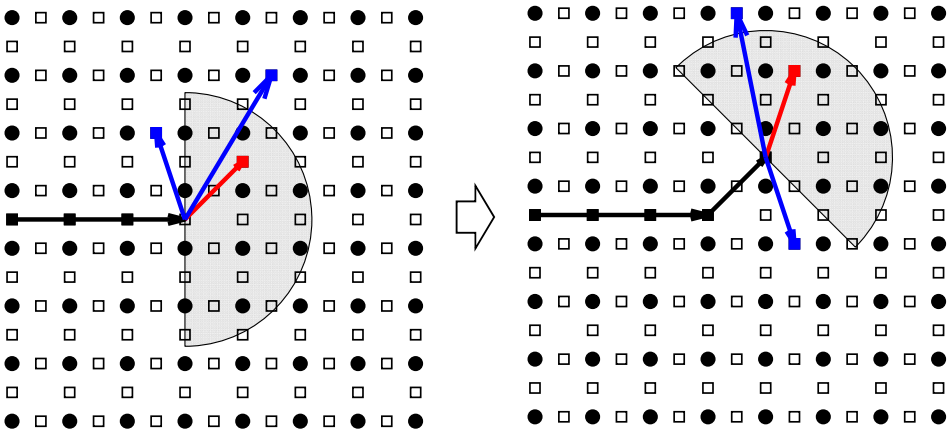


Figure 9.7. Potential crack tip locations in algorithm 1

shown in Figure 9.6. Point **c** is also invalid as it violates the second criterion. Therefore, only the point **d** is valid and determined as the new crack tip.

This algorithm essentially limits the new crack tips to be located within a circular sector centered at the previous crack tip and prevents non-physical crack path. This also makes the numerical implementation very simple and efficient. The shape of the circular sector is controlled by the specified radial and circumferential ranges in the two criteria. For example, the shaded semicircles in Figure 9.7 show the potential crack tip locations at two different

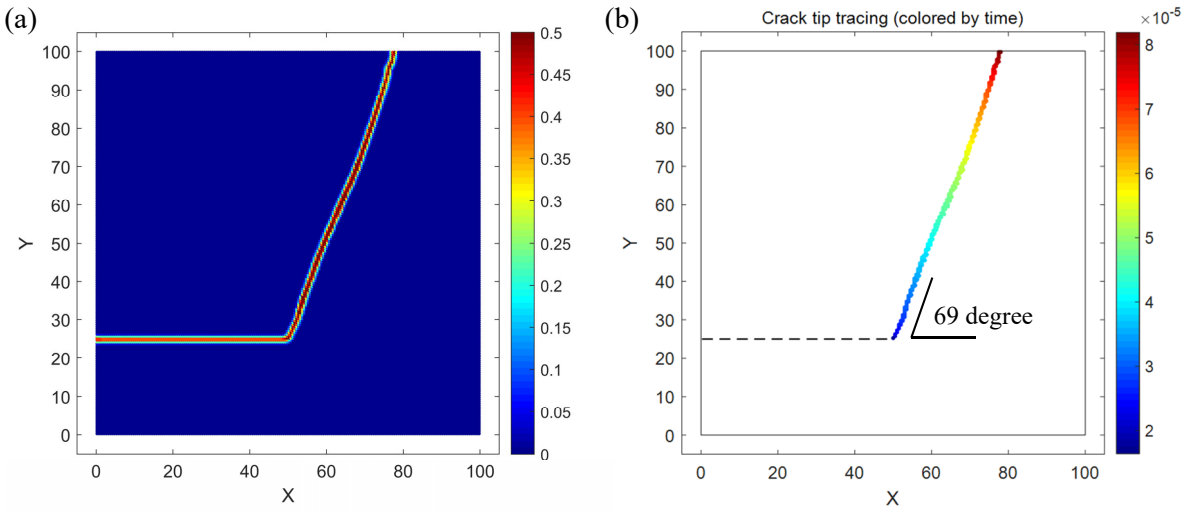


Figure 9.8. Example of crack tip tracking: (a) contour of the nodal damage variables and (b) the identified crack path

time steps. Note that in cases where multiple valid new crack tips coexist at the same time step, it usually indicates the event of crack branching and must be handled properly.

Figure 9.8 shows an example of the above crack tip tracking algorithm. In this example, a half model of the Kalthoff-Winkler experiment studied in Section 7.4.3 is considered with the symmetry boundary condition applied at the bottom edge ($y = 0$). The initial crack is placed at (0, 25) to (50, 25). The constant velocity in the x direction is applied at nodes where $x = 0$ and $y < 25$. Figure 9.8 (a) plots the contour of the nodal damage field at the final configuration, which shows the crack path clearly. The identified crack path is plotted in Figure 9.8 (b) and colored by the step time of when the crack tip is identified. The initial crack prescribed at $t = 0$ is indicated by the dashed line. It can be seen that the crack path is accurately identified using the proposed algorithm. By curve fitting the identified crack path, we find the angle between the initial crack and the crack propagation is 69°, which matches the experiment observation (around 70°) very well.

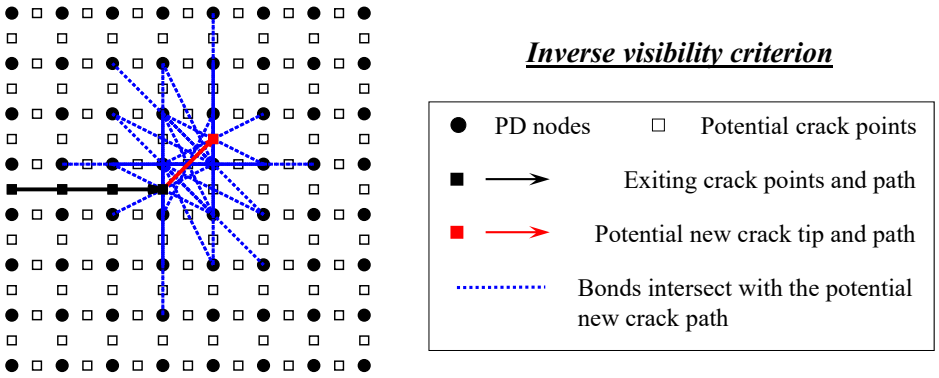


Figure 9.9. Inverse visibility criterion for crack tip tracking

9.5.1.2 Algorithm 2: inverse visibility criterion

The second crack tip tracking algorithm is based on the visibility criterion in meshfree methods (Belytschko and Tabbara, 1996; Rabczuk and Belytschko, 2004), which simply indicates that nodes on one side of the crack surface cannot see the nodes on the opposite side. This method was applied for modeling the initial cracks in PD simulation: all bonds intersecting with a given initial crack are set to broken prior to time integration. The inverse process can be employed to identify new crack path.

The crack tip tracking algorithm based on the inverse visibility criterion is illustrated in Figure 9.9. For a given potential crack path, the status of all the bonds intersecting with it needs to be checked. If all these bonds are broken, then the potential crack path is determined as the new crack path. This algorithm is also simple and free of any parameters compared to the previous one. However, an efficient numerical implementation is required to reduce the total number of intersection checking and the computational cost.

Finally, it is worth noting that in (Giannakeas et al., 2020b) the authors proposed a crack tip tracking algorithm based on nodal damage variables. In their approach, the crack tip is determined as the local minimum of the discrete Laplacian of the damage field. This algorithm can also be incorporated in the current work.

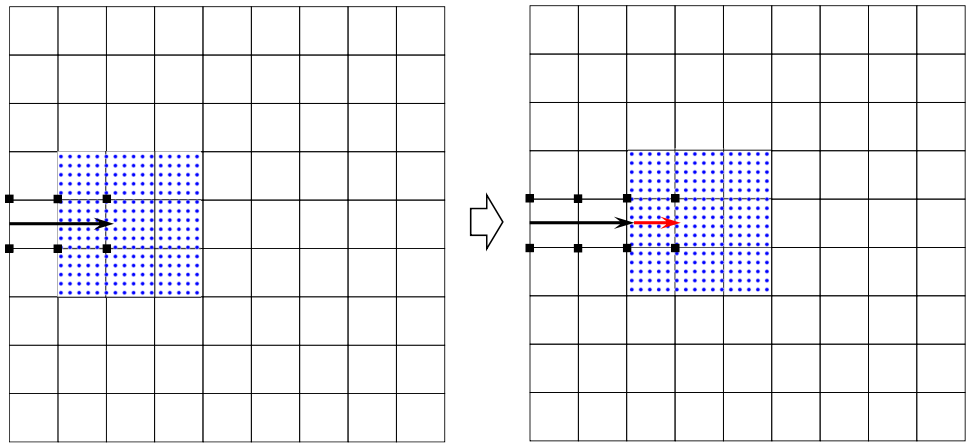


Figure 9.10. Shifting of the PD region due to crack propagation

9.5.2 PD region shifting algorithm

In (Giannakeas et al., 2020b), a two-step expansion/contraction process is proposed for PD region relocation in coupled PD/XFEM simulation. In (Wada, 2017), the author developed a PD region shifting algorithm based on the crack tip location and the coarse-scale FE mesh. In this work, we propose a simple PD region shifting approach which is similar to the latter (Wada, 2017).

First, we partition the fine-scale PD region using a coarser background grid, e.g. a 3×3 uniform grid in 2D cases. For the purpose of illustration, here we employ the overlapping coarse-scale finite elements for partitioning the PD nodes. Figure 9.10 illustrates the shifting scheme. In this figure, PD nodes are represented by solid dots while the grid is the FE mesh. Enriched FE nodes are indicated by solid squares. The PD region is initially placed in such a way that the crack tip is located at the center element in the background grid (i.e. the overlapping elements) as indicated by the tip of the black arrow. During the simulation we keep track of the location of the crack tip as well as its relative position to the background grid (e.g. parent element index). A shifting vector (the red arrow in Figure 9.10) is obtained

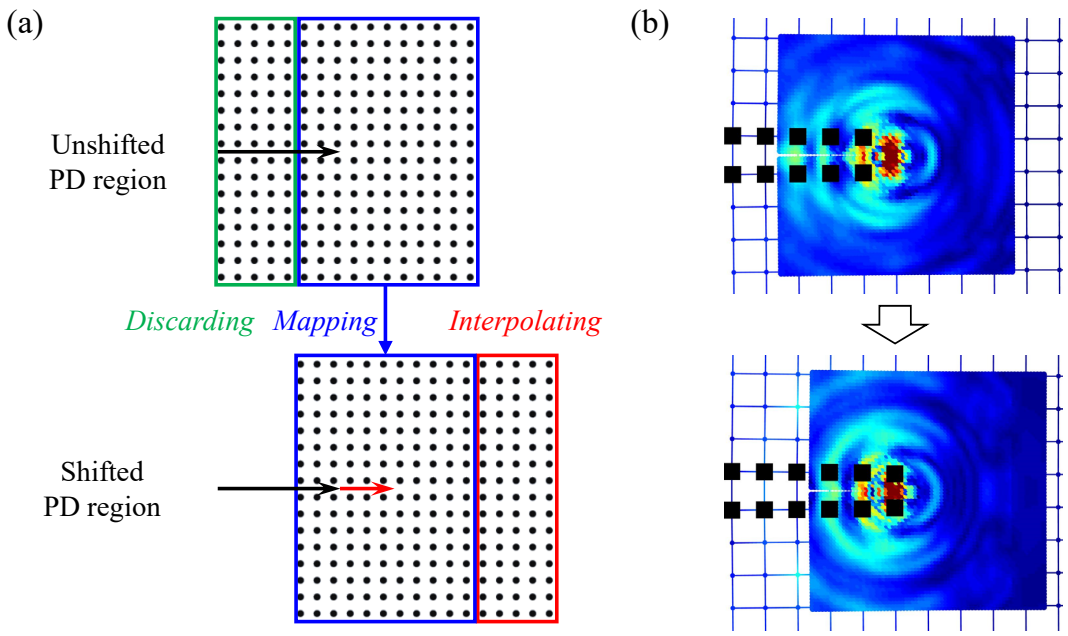


Figure 9.11. Updating state variables in the shifted PD region

once the crack tip propagates outside the original element into a new element. Subsequently, the PD region is shifted by using the shifting vector as shown in Figure 9.10.

By shifting the PD region, the PD nodal coordinates are updated by adding the shifting vector. Since the region is simply shifted, there is no need to search for neighbor nodes again and thus saves computational effort. However, we still need to determine the state variables such as displacement, velocity, damage, bond status (the influence function), etc., for the shifted PD region. Here, a simple mapping/interpolating scheme is established to update the PD state variables. As illustrated in Figure 9.11 (a), the unshifted PD region is divided into two sub-regions: one bounded by the green box, which is far away from crack tip and will be discarded in the shifted PD region; the other one bounded by the blue box which is mapped onto the shifted PD region. Similarly, the shifted PD region can also be partitioned into two sub-regions: one region bounded by the blue box which is overlapping with the unshifted PD region; and the other one bounded by the red box needs approximated state variables. In the overlapping sub-region (the blue boxes), the nodal state variables are directly mapped from

the unshifted PD region to the shifted one. In the red box of the shifted PD region, the nodal values are interpolated from the coarse scale nodal values using the FE shape functions. For bond status, it is mapped if both nodes associated with the bond are located in the blue box, otherwise it is set to be an intact bond. The nodal damage value is re-evaluated after updating the bond status. Figure 9.11 (b) shows an example of the mapping/interpolating algorithm. In this example, a model configuration similar to Figure 9.10 is subjected to the Mode I fracture load. The PD region is colored by velocity magnitude field and the waves generated by debonding are clearly visualized. The two frames illustrated in this figure are captured before and after the PD region shift. It can be seen that the mapping/interpolating algorithm works very well in this example.

9.5.3 Updating the concurrent coupling scheme

Along with the PD region shifting, other components in the concurrent coupling scheme also need to be updated. First of all, to account for the strong discontinuity due the new crack path obtained from the PD region, the corresponding coarse-scale elements are enriched by Heaviside function. Secondly, the FE shape functions associated with the NMBC nodes are updated for calculating the incoming waves to the shifted PD region. Finally, the bridging-scale projection is updated using the shifted PD nodal positions and the updated FE shape functions.

As a brief summary, the proposed adaptivity algorithm relocates the PD region in the following three steps:

1. Tracking the crack tip using broken bonds and nodal damage variables;
2. Shifting the PD region and updating state variables when the shifting trigger is activated; and
3. Updating XFEM, NMBC, and BSP after PD region shifting.

9.6 Numerical implementation

Numerical implementation of the proposed concurrent multiscale framework by coupling PD with FEM is summarized below:

1. Preprocess for FE and PD regions
2. Setup two-way NMBC PD nodes, coefficients and the associated FE DOFs
3. Setup bridging-scale projection operators and the associated FE and PD DOFs
4. Initialization for time integration
5. Start the time stepping
    (a) Solve the FE equation of motion over the entire spatial domain to update the FE nodal acceleration, velocity and displacement vectors
    (b) Solve the PD equation of motion in the local region of interest to update the PD nodal acceleration and velocity vectors
    (c) Apply two-way NMBC to PD region to update PD nodal velocity vectors
    (d) Update PD nodal displacement vectors
    (e) Apply the bridging-scale projection to update FE nodal displacement vectors in the overlapping region
    (f) Apply essential boundary conditions to FE domain
    (g) Check the crack tip location and shift the PD region when necessary
    (h) Proceed to next time step
6. Postprocess

In the above implementation, we assumed that FE and PD simulations have the same time step size. However, FE simulation can generally employ a much bigger time step size due to its coarser spatial discretization. In this case, a multi-time-stepping algorithm can be employed (Qian and Chirputkar, 2014). In the multi-time-stepping algorithm, we

first advance the coarse-scale FE simulation by one large time step. The fine-scale PD simulation is then advanced by several smaller time steps. During the sub-incremental steps, the incoming waves for the two-way NMBC are interpolated from the FE solutions. At the end of sub-incremental steps, the FE nodal values in the overlapping region are updated by the bridging-scale projection with the fine-scale solution.

9.7 Summary

In this chapter, we have developed a concurrent multiscale framework based on PD and FEM. In this framework, coarse-scale FE model is applied to the entire domain while fine-scale PD model is limited to a local region of interest. Coupling between the FEM and PD models are realized with two components. First, a class of two-way nonlocal matching boundary conditions is established to absorb the outgoing waves in PD model and transmits incoming waves from FE model to PD model through the numerical interface. Second, a bridging-scale projection operator is introduced to project the fine-scale solution onto the coarse-scale basis in the PD/FE overlapping region. Finally, we established an adaptive scheme to track crack growth in PD region and mapping back the crack path to FE model using enrichment shape function and extra DOFs. The PD region shifts along with the crack tip to capture crack propagation in dynamic fracture simulations.

CHAPTER 10

APPLICATIONS ON ELASTODYNAMICS AND FRACTURE

10.1 Introduction

In this chapter, we first present several numerical experiments for wave propagation problems to systematically demonstrate the efficiency and accuracy of the proposed concurrent coupling scheme in Chapter 9. To further demonstrate the robustness and computational performance of the developed concurrent multiscale framework, dynamic fracture problems are studied and the results are validated against full PD simulations as well as experiment observations.

10.2 Longitudinal wave propagation in an infinitely long 1D bar

In the first example we consider the longitudinal wave propagation in an infinitely long 1D bar with a uniform cross-sectional area $A = 1$. We only explicitly model the bar with a finite length of $L = 500$. Both ends of the finite-length bar are treated by either MBC or NMBC to transmit outgoing waves into the regions that are not explicitly modeled to achieve the non-reflective or absorbing boundary conditions. To generate the outgoing wave, a harmonic wave profile is imposed at the middle of the model, which is given by

$$u(x,t) = u_0 (1 - \cos(\omega t)) \qquad \text{for } x \in [L/2 - 2\delta, L/2 + 2\delta], t \in [0, 2\pi] \tag{10.1}$$

where the amplitude $u_0 = 0.0025$, angular frequency $\omega = 1$.

Material properties of the bar are given as Young's modulus $E = 57.1464$ and density $\rho = 1$. This 1D model is discretized with evenly distributed nodes with spacing $\Delta X = 1$. A time step size $\Delta t = 0.02$ is employed and the total simulation time is $T = 80$.

10.2.1 NMBC with BBPD

First, we simulate this problem with the bond-based PD (BBPD) and study the performance of MBC and NMBC. Figure 10.1 shows a frame-to-frame comparison of wave profiles between the solutions obtained by MBC and NMBC. Note that the nearest neighborhood, i.e. $\delta = 1\Delta X$, is employed here. It shows that both MBC and NMBC transmit the outgoing wave at the artificial boundaries. In the case of MBC, the majority part of the outgoing waves is transmitted. However, due to the so-called edge effect in MBC, a small portion of the outgoing wave has been reflected at both ends of the model, which is apparent in Figure 10.1 (a) at the last two timeframes. In contrast, the wave reflections in the NMBC case are almost indiscernible as shown in Figure 10.1 (b), which shows superior performance of NMBC in terms of absorbing outgoing waves. Figure 10.2 illustrates a more detailed comparison of the residual waves by the end of the simulation. Note that the reference result is obtained by extending the modeled bar length to the extent such that the outgoing waves cannot reach the artificial boundaries by the end of the simulation and monitoring nodal solutions only within the range of the original modeled bar length, i.e. $x \in [0, 500]$.

To quantitatively study the wave absorption efficiencies of MBC and NMBC, we compare their total energies in the system as a function of time. Figure 10.3 illustrates a comparison between total energy histories obtained from the cases with MBC and NMBC. Note that both energy and time are normalized. Also, a reference total energy history is provided as an ideal case for the purpose of comparison. The reference result is obtained in the same way as explained earlier. Figure 10.3 (a) shows that the normalized total energies remain constant of 1 before the outgoing waves reach the artificial boundaries, then reduces as the outgoing waves are transmitting through the boundaries. After that, the remaining energy essentially provides a good measurement for the efficiency of the absorbing boundary conditions. As shown in Figure 10.3 (a), the final normalized remaining energy for the MBC case is around 1.39×10^{-3}, which is the highest among all the cases and is caused by the edge effect in

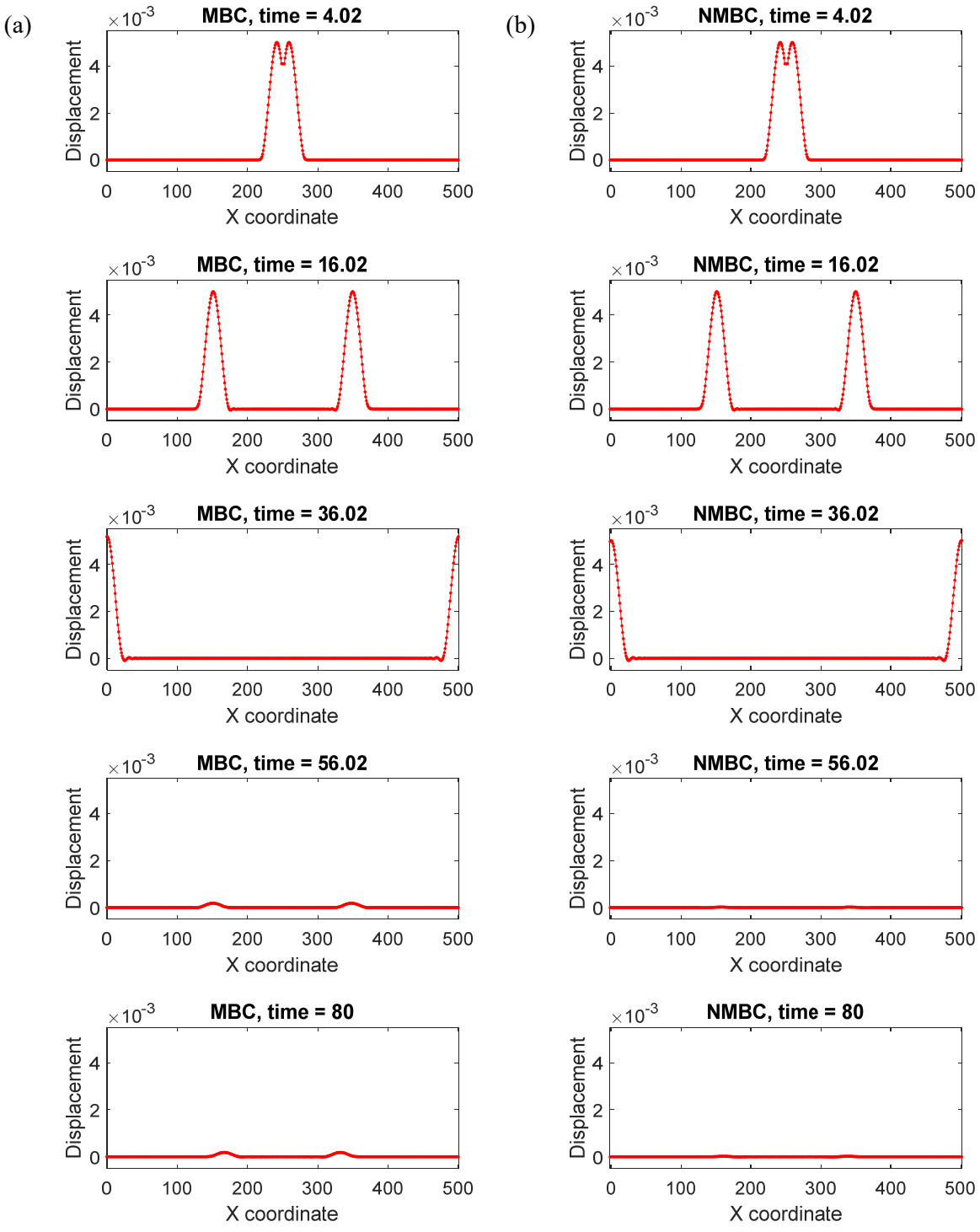


Figure 10.1. Timeframes of wave propagation in the 1D bar obtained by BBPD with $n = 1$ and boundaries are treated by (a) MBC and (b) NMBC

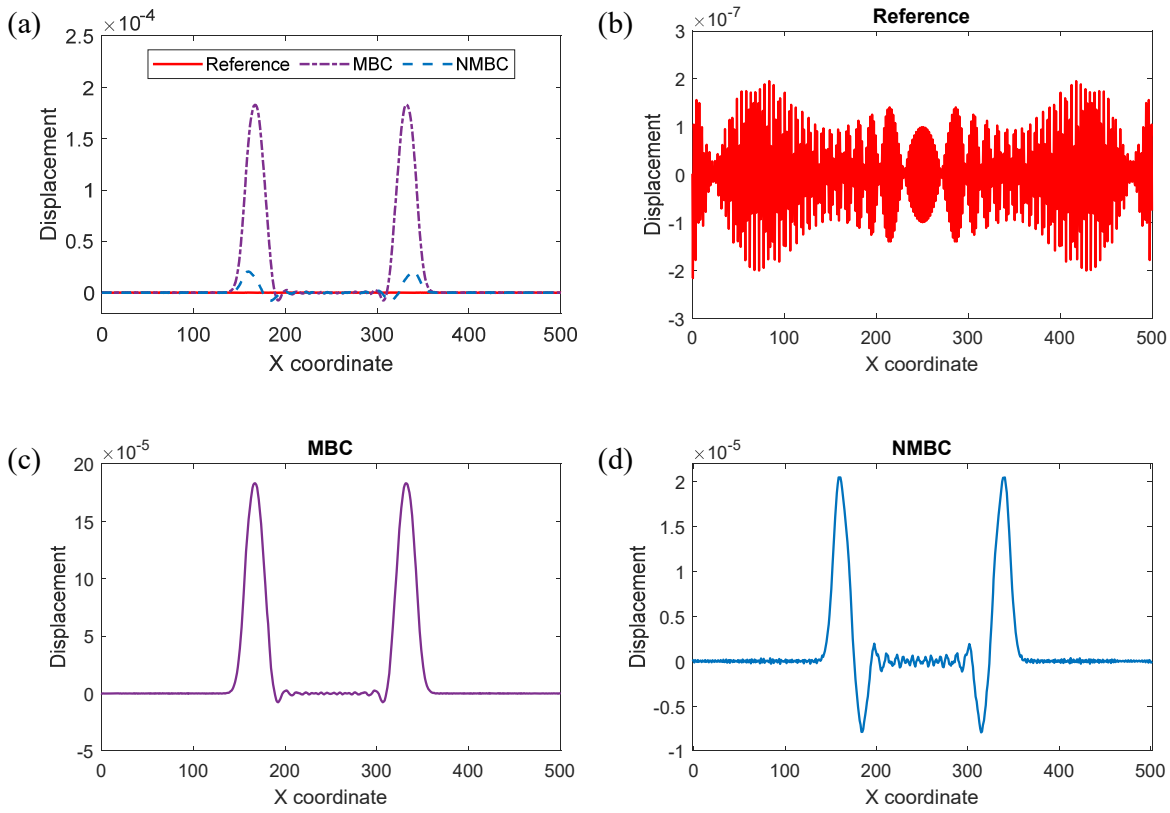


Figure 10.2. Residual waves in the 1D bar by BBPD by the end of the simulation: (a) the comparison, (b) reference solution, (c) MBC and (d) NMBC cases

MBC. Nevertheless, 99.86% of the energy associated with the outgoing waves are transmitted through the MBC implementation. The NMBC case shows a normalized remaining energy around 3.64×10^{-5} by the end of the simulation, which is more than 38 times lower compared to the MBC case. In other words, the wave absorption efficiency of NMBC is greater than 99.996%. Lastly, the normalized remaining energy in the reference solution at the final time step is around 3.60×10^{-6}, which is an order of magnitude lower than the NMBC case and again confirms the superior wave absorption efficiency of the NMBC.

It is worth noting that remaining energies of both the MBC and NMBC cases reach plateau after outgoing waves passed through the artificial boundaries as shown in Figure 10.3 (a), which is due to the fact that the reflected waves cannot reach the absorbing boundaries

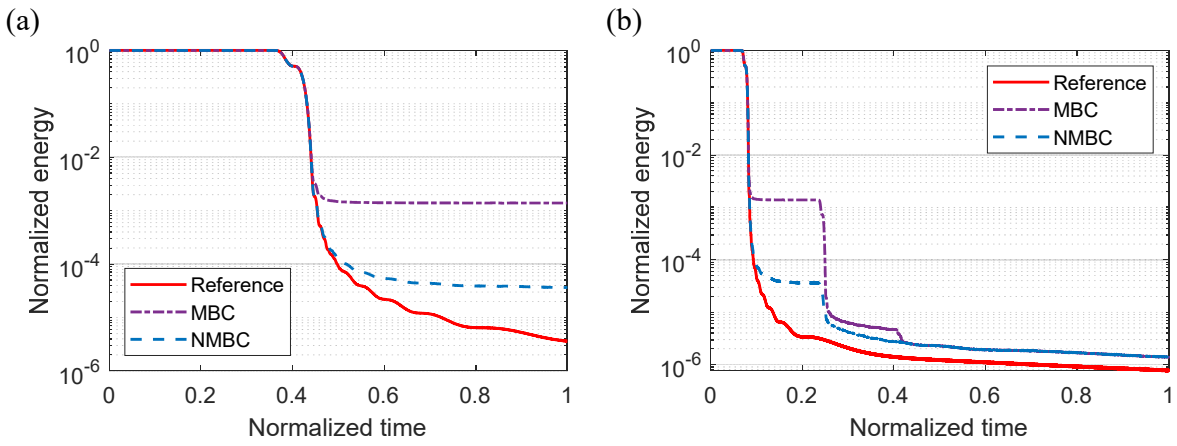


Figure 10.3. Total energy histories for wave propagation in the 1D bar by BBPD with $n = 1$: (a) the original simulation with $T = 80$ and (b) the extended simulation with $T = 400$

by the end of the simulation. To validate this explanation, we extend the total simulation time by five times to allow the reflected waves passing through the artificial boundaries a couple of times. As illustrated in Figure 10.3 (b), the energies of both MBC and NMBC cases drop a few more times and eventually converged to a value that is only slightly higher than the reference case.

Next, we study the NMBC wave absorption efficiency in BBPD by varying the horizon radius from $\delta = 1\Delta X$ to $\delta = 4\Delta X$. Solution of the nearest neighborhood case is employed as a baseline for the purpose of comparison since it already demonstrated an excellent efficiency. The normalized total energies are plotted in Figure 10.4 for cases $n = 1 \sim 4$. At the final time step, the cases with larger n yield higher remaining energies in the system. It indicates that the NMBC becomes less efficient with an increasing horizon radius, which could be possibly attributed to the increased nonlocality in PD with larger horizons. However, the normalized remaining energy is still less than 0.03% in the worst-case scenario with $n = 4$, which shows a remarkable wave absorption efficiency of NMBC regardless the size of horizon radius in BBPD.

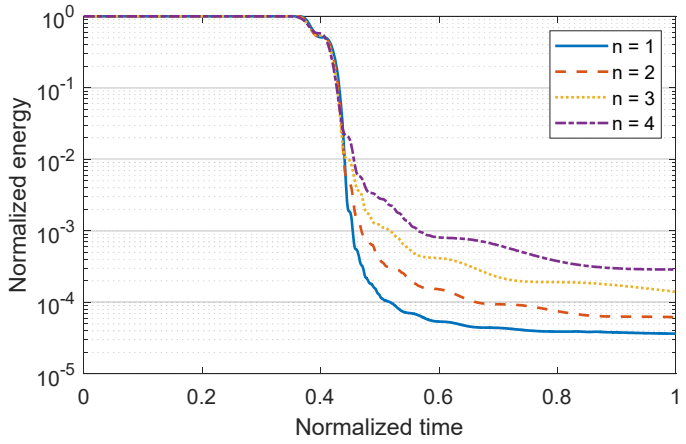


Figure 10.4. Total energy histories for wave propagation in the 1D bar by BBPD with NMBC and various horizons: $n = 1 \sim 4$

To further understand why the NMBC efficiency decreases with growing neighborhood sizes, we captured the wave propagations at different timeframes for the cases of $n = 2$ and $n = 3$, as shown in Figure 10.5. From the first two timeframes, waves with shorter wavelengths in the wake of the main outgoing long waves are clearly observed. These short trailing waves are induced by the increased wave dispersions of PD due to its nonlocality with larger horizons. Recall that the NMBC employed here is the Taylor-type that mainly aims at eliminating the wave reflections in the limit of long wave ($\kappa \to 0$ or $\lambda \to \infty$). Therefore, it is less efficient in terms of absorbing these trailing waves with shorter wavelengths. Nevertheless, the reflected waves are barely visible as shown in the last timeframe in Figure 10.5, which illustrates a good overall efficiency of NMBC on absorbing all these waves. In cases that the current NMBC no longer efficient due to the presence of short waves, the Taylor-Newton-type NMBC can be introduced.

The efficiency of NMBC in BBPD is tested by imposing initial wave profiles with different wavelengths, as shown in Figure 10.6. Three distinct wavelengths range from 50 to 200 are considered here. Similarly, the normalized total energy histories for these cases are plotted in Figure 10.7. Since long wave travels faster than short wave, the wave with $\lambda = 200$ reaches

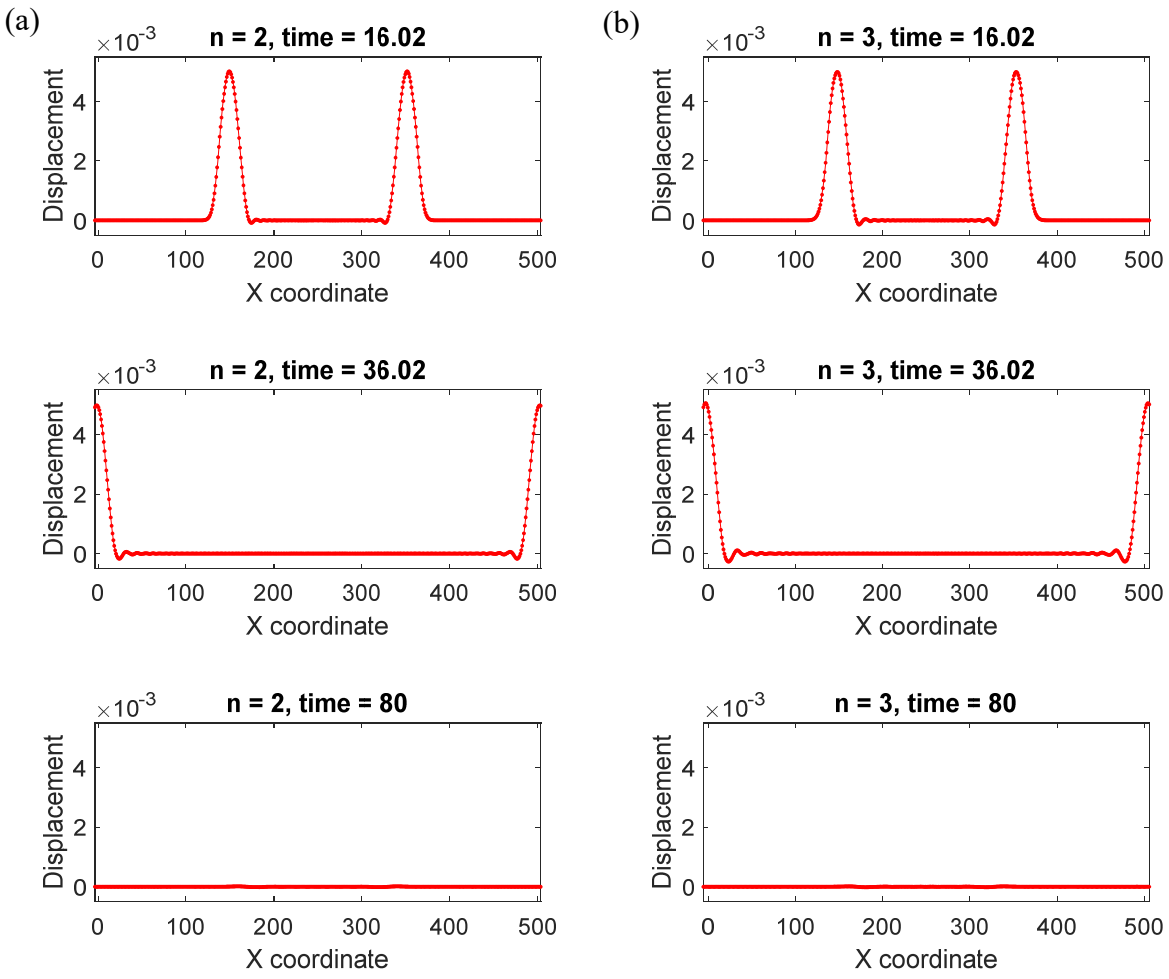


Figure 10.5. Timeframes of wave propagation in the 1D bar obtained by BBPD with NMBC and various horizons: (a) $n = 2$ and (b) $n = 3$

the artificial boundaries first and its total energy drops earlier than the other cases. As expected, the NMBC absorbs the long wave better compared to the shorter ones. But in all cases the normalized remaining energy is less than 0.01% by the end of the simulation.

10.2.2 NMBC with NOPD

A similar study on MBC and NMBC is conducted with non-ordinary state-based PD with correspondence materials (NOPD). Figure 10.8 shows the frame-to-frame comparisons of

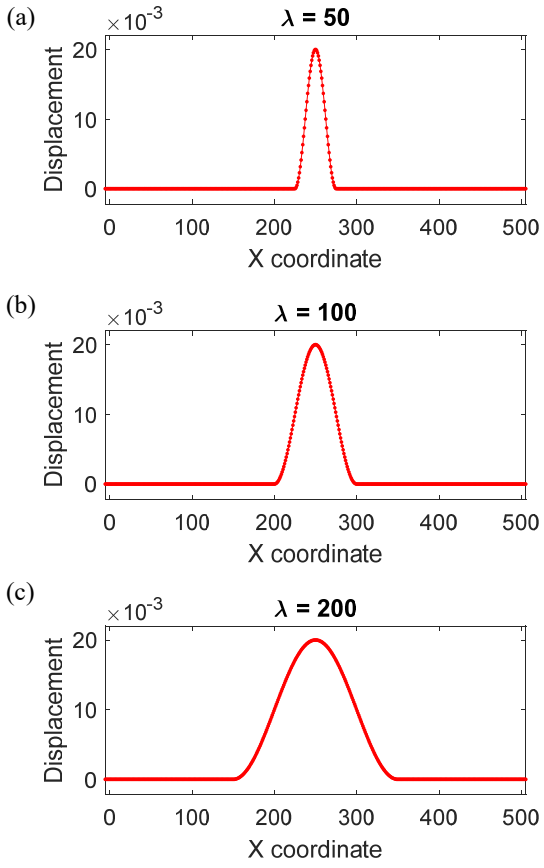


Figure 10.6. Initially imposed wave profiles with different wavelengths: (a) $\lambda = 50$, (b) $\lambda = 100$, and (c) $\lambda = 200$

wave propagation between MBC and NMBC cases obtained by NOPD with $n = 1$. Similar to the results in the BBPD, the MBC case shows reflected waves with much larger amplitude compared to the NMBC case. However, in contrast to the BBPD cases, the wave reflections are more evident in NOPD with both MBC and NMBC.

Quantitative measurements of the wave absorption efficiency are illustrated in Figure 10.9. It shows that the normalized remaining energies are 0.12 and 0.009 for MBC and NMBC, respectively, which are orders of magnitude higher compared to the BBPD cases. In fact, even the remaining energy in the reference solution increases to 2.3×10^{-5}, which is around 10 times greater than the reference result in the BBPD case. This significantly deteriorated

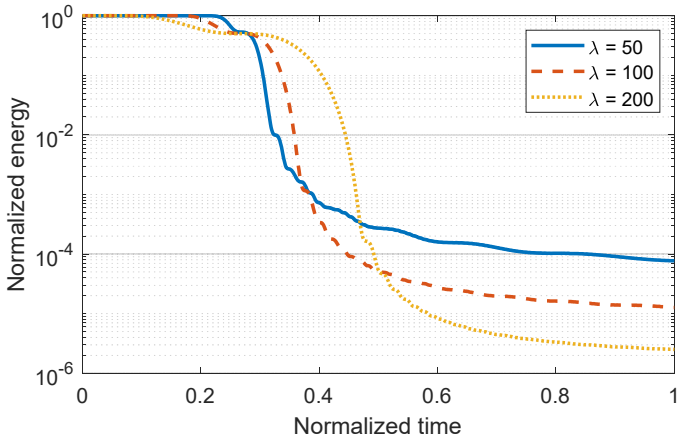


Figure 10.7. Total energy histories for wave propagation in the 1D bar by BBPD ($n = 3$) with NMBC and various initial wavelengths: $\lambda = 50 \sim 200$

performance is attributed to the zero-energy modes due to the inherent material instability in NOPD formulation.

The spurious zero-energy modes are more apparent for larger neighborhood sizes in NOPD, which are clearly illustrated in Figure 10.10. A large number of high frequency waves are generated by these spurious modes spontaneously along with the outgoing wave. NMBC cannot efficiently absorb these waves due to their very short wavelengths. Therefore, large wave reflections are expected and also observed in Figure 10.10. Furthermore, amplitudes of the reflected waves are higher with the increasing horizon radius by comparing the results obtained by the cases of $n = 2$ and $n = 3$.

Similarly, we collected the total energy histories for cases with larger horizon radius ($n = 2 \sim 4$) and plotted them against the local case ($n = 1$) in Figure 10.11. As expected, the efficiency of NMBC decreases as increasing horizon size. However, unlike the BBPD cases, the cases with larger neighborhood in NOPD show escalating remaining energies even after the outgoing waves passed through the absorbing boundaries, which is due to the zero-energy modes. The NOPD formulation is excluded in the proposed concurrent multiscale framework because of its instability and the resulting poor performance.

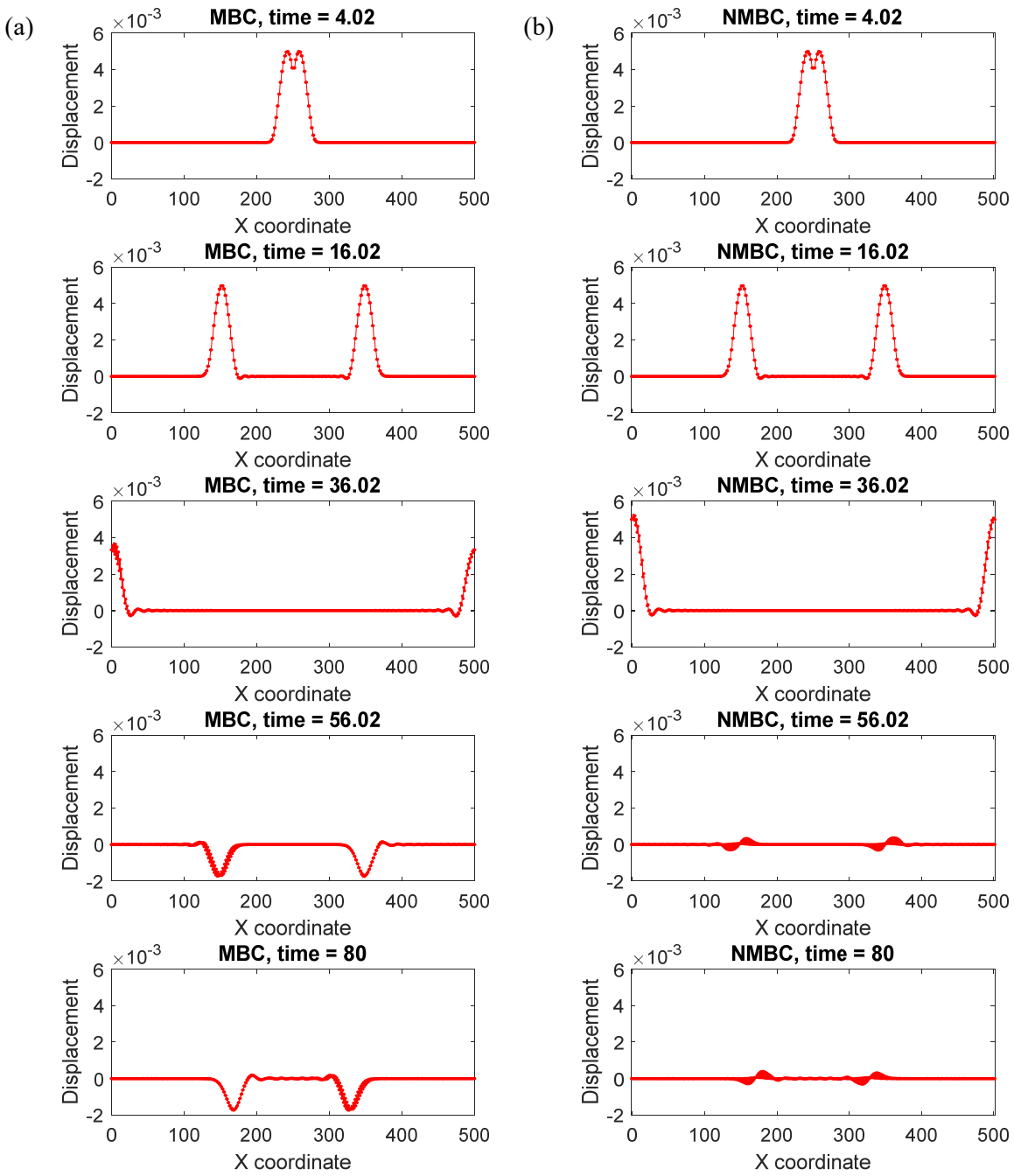


Figure 10.8. Timeframes of wave propagation in the 1D bar obtained by NOPD with $n = 1$ and boundaries are treated by (a) MBC and (b) NMBC

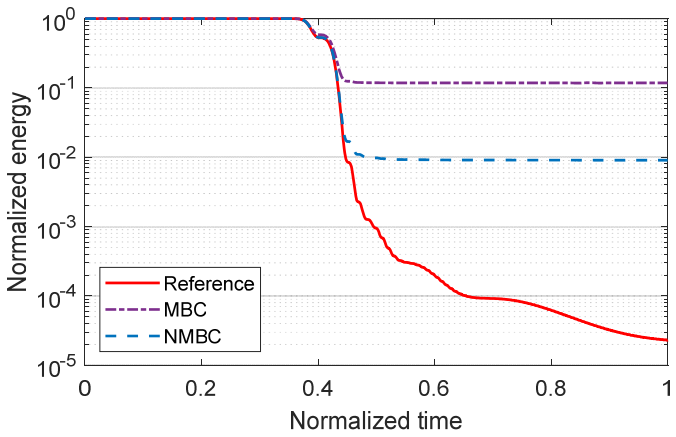


Figure 10.9. Total energy histories for wave propagation in the 1D bar by NOPD with $n = 1$ and boundaries are treated by MBC and NMBC

10.2.3 NMBC with SNOPD

We proceed to study the performance of MBC and NMBC in the stabilized non-ordinary state-based PD with correspondence materials (SNOPD). The stabilization parameter of $G = 0.5$ is employed here unless otherwise stated. Figure 10.12 illustrates a comparison of wave propagation profiles between MBC and NMBC with the nearest neighborhood. Figure 10.12 (a) shows that the MBC in SNOPD still leads to significant wave reflections due to the edge effect. On the other hand, wave reflections observed in the NMBC with NOPD case are substantially suppressed by virtue of the extra stabilization force introduced in SNOPD. Figure 10.12 (b) shows that the performance of NMBC is fully restored as compared to the BBPD case.

We also plotted the normalized energy histories in Figure 10.13. The remaining energy in the MBC case is around 0.04, which is improved only 3 times compared to the NOPD case. In the NMBC case the final remaining energy reduces to 3.1×10^{-5}, which is improved roughly 290 times compared to the NOPD case and even slightly better than the BBPD case. The remaining energy in the reference case is 1.8×10^{-6}.

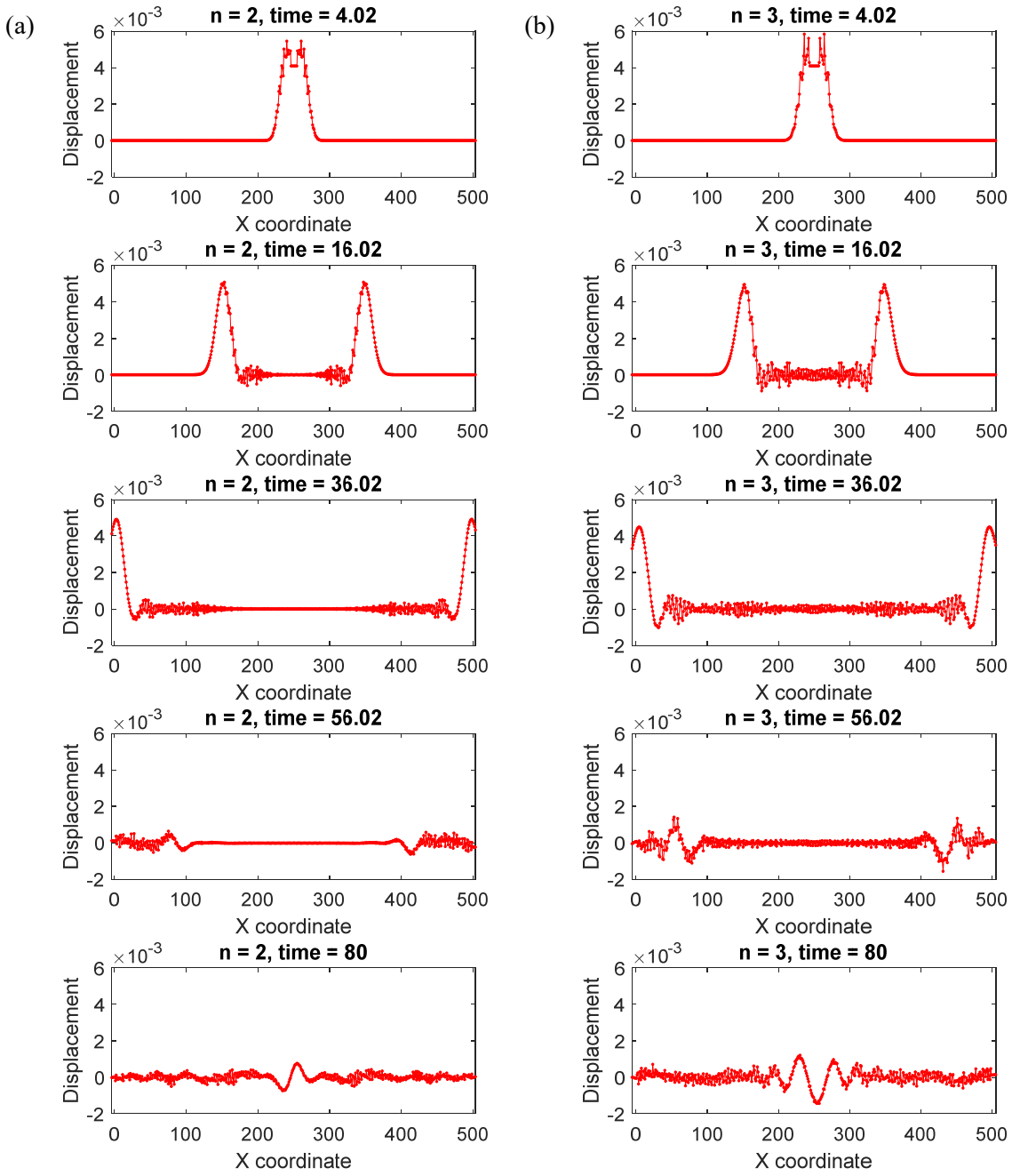


Figure 10.10. Timeframes of wave propagation in the 1D bar obtained by NOPD with NMBC and various horizons: (a) $n = 2$ and (b) $n = 3$

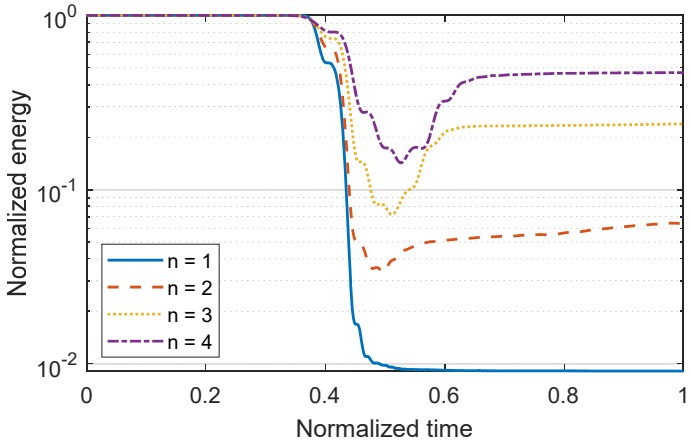


Figure 10.11. Total energy histories for wave propagation in the 1D bar by NOPD with NMBC and various horizons: $n = 1 \sim 4$

NMBC performance in SNOPD with larger neighborhood sizes are also investigated. The wave propagation profiles captured at different timeframes for the cases of $n = 2$ and $n = 3$ are illustrated in Figure 10.14 (a) and (b), respectively. Compared to the NOPD cases, the abundant wave reflections are almost eliminated by NMBC in SNOPD with larger horizon radii. In addition, as already seen in the BBPD cases, we also observe from Figure 10.14 that larger horizon radii lead to more apparent trailing waves with short wavelengths, which weaken the overall wave absorption efficiency of NMBC.

Normalized energy histories for different horizon radii in SNOPD are shown in Figure 10.15 to quantitatively study the NMBC performance. As we anticipated, the wave absorption efficiency of NMBC drops as the horizon radius increases in virtue of nonlocality in PD. For the widely employed $n = 3$ case, the final remaining energy is 0.0015. To further improve the NMBC efficiency in this case, we tune the stabilization parameter G in SNOPD as shown in Figure 10.16. It shows that the efficiency improves with larger G. With $G = 2$, the remaining energy reduces to 0.0004, which is roughly 4 times lower than the previous case.

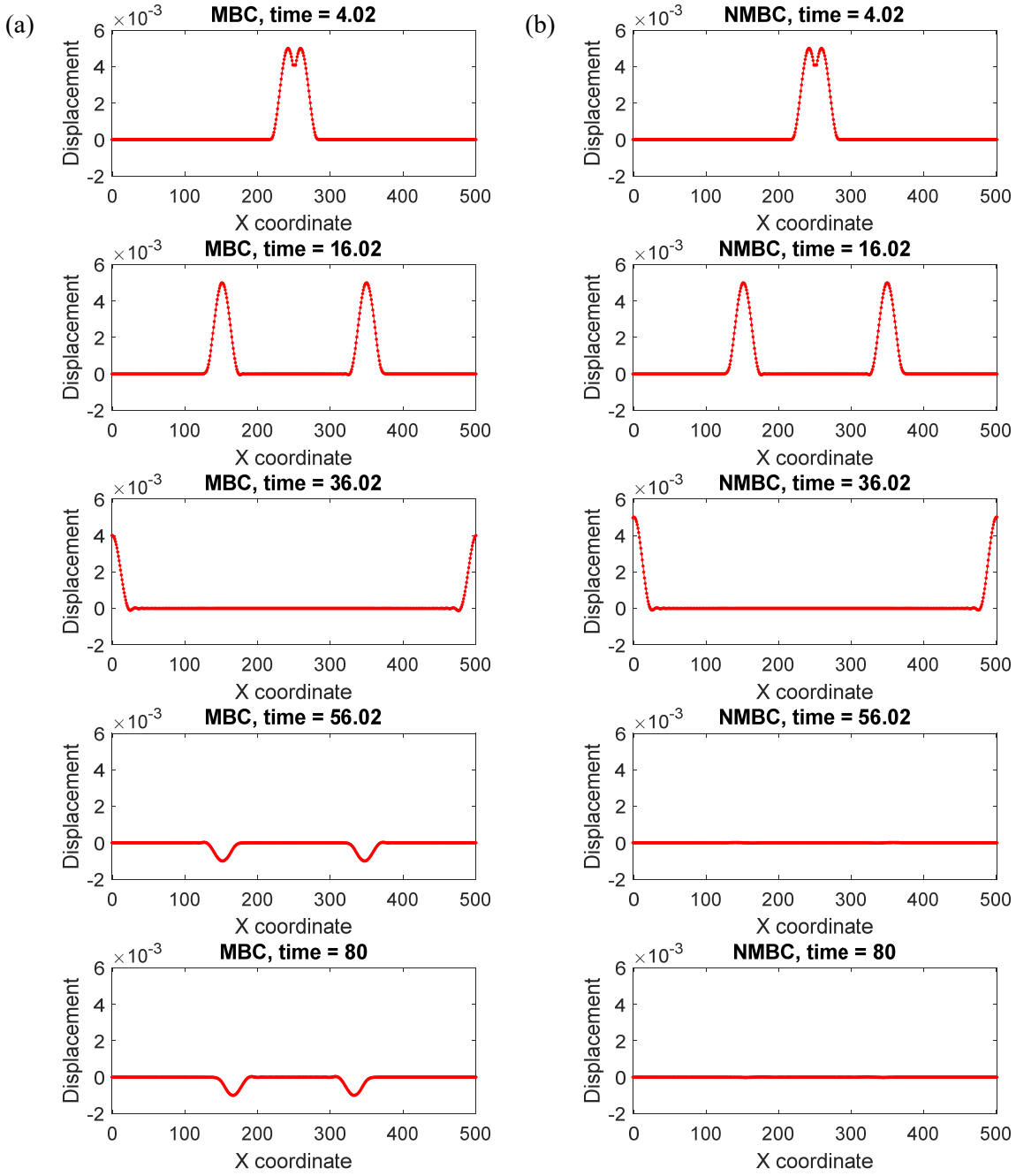


Figure 10.12. Timeframes of wave propagation in the 1D bar obtained by SNOPD with $n = 1$ and $G = 0.5$ and boundaries are treated by (a) MBC and (b) NMBC

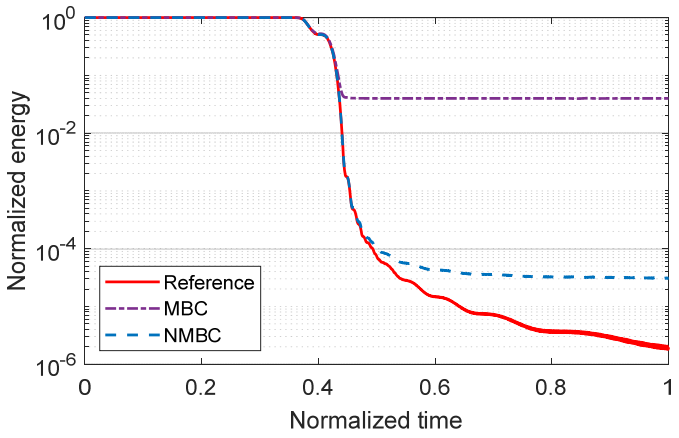


Figure 10.13. Total energy histories for wave propagation in the 1D bar by SNOPD with $n = 1$ and $G = 0.5$ and boundaries are treated by MBC and NMBC

In this 1D example, we systematically studied the performance of MBC and NMBC in various PD formulations. With the nonlocal extension to account for the edge effect, NMBC always shows a better wave absorption efficiency compared to MBC cases. In addition, NMBC works well for both BBPD and SNOPD. Its efficiency is inversely proportional to the horizon radius and directly proportional to the stabilization parameter G in SNOPD. Even with large neighborhood sizes or small G values, NMBC still yields satisfactory wave absorption efficiency. Finally, neither MBC nor NMBC is efficient in NOPD due to the spurious zero-energy modes.

10.3 Longitudinal wave propagation in a 1D bar with finite length

In this example, we consider a similar problem to the previous one except that the 1D bar has a finite length $l = 4,000$ in this case. The bar also has a unit cross-sectional area and the material parameters are the same as in the previous example. The middle segment of the bar with a length $L = 1,000$ is simulated by the fine-scale PD with a uniform nodal spacing $\Delta X = 1$, while the entire bar is modeled by the coarse-scale FEM with 2-node linear bar elements and a constant element length $\Delta l = 20$. This multiscale configuration

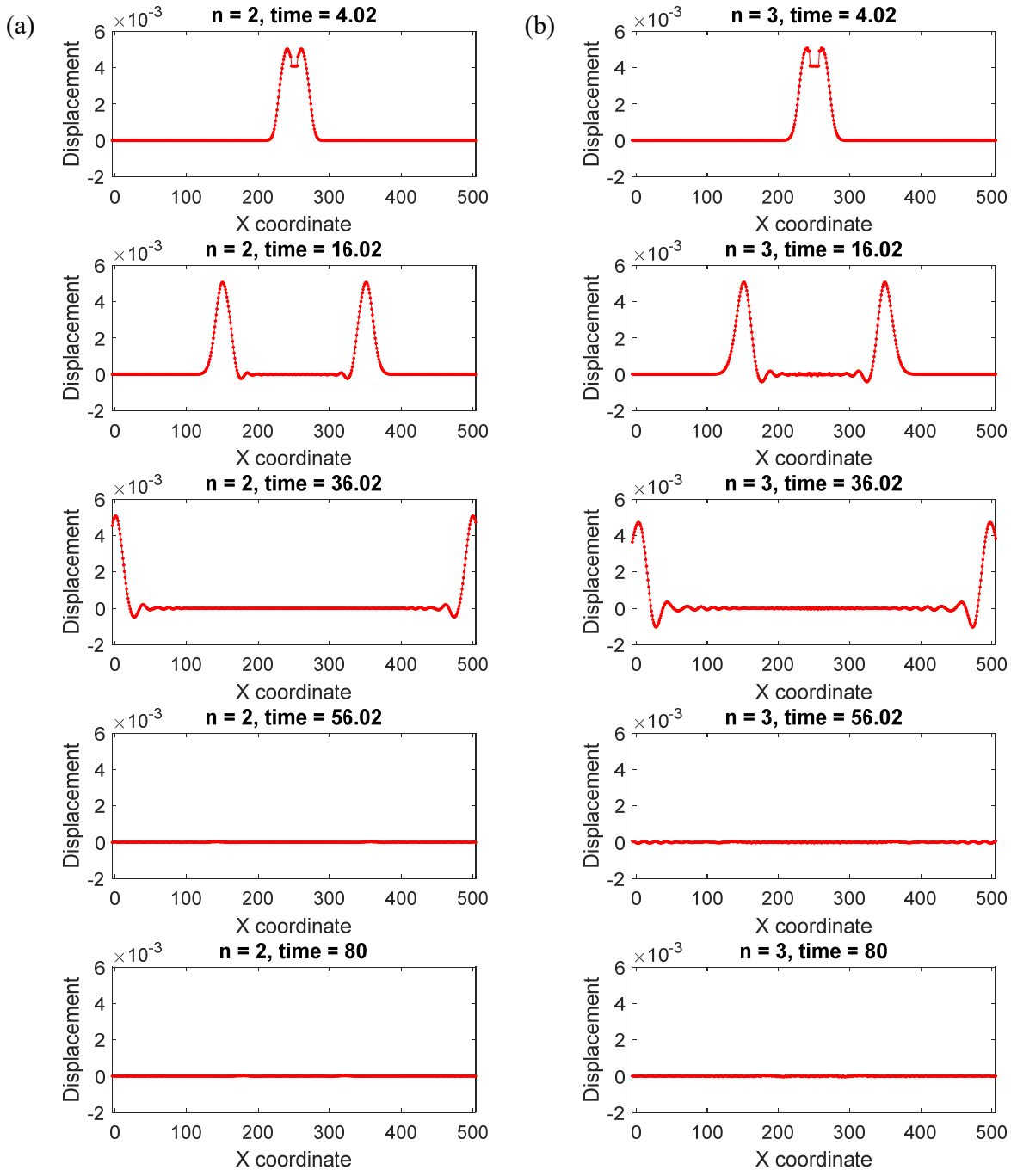


Figure 10.14. Timeframes of wave propagation in the 1D bar obtained by SNOPD with $G = 0.5$ and NMBC and various horizons: (a) $n = 2$ and (b) $n = 3$

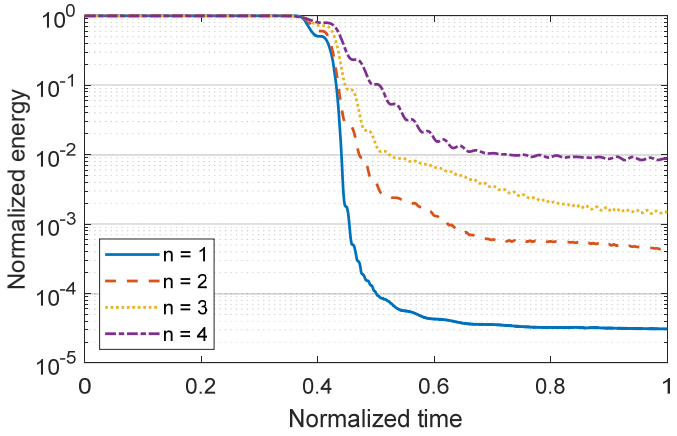


Figure 10.15. Total energy histories for wave propagation in the 1D bar by SNOPD with $G = 0.5$ and NMBC and various horizons: $n = 1 \sim 4$

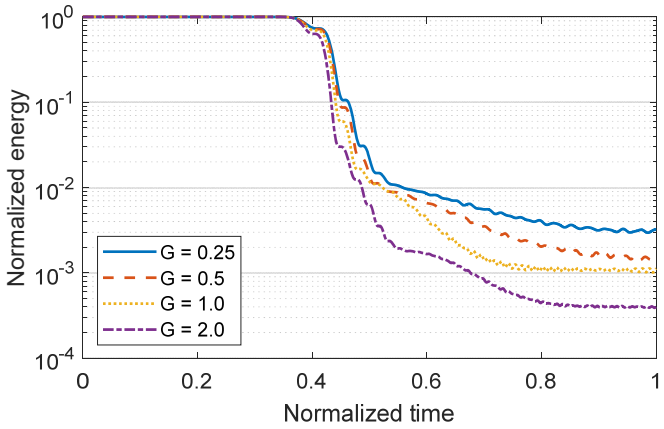


Figure 10.16. Total energy histories for wave propagation in the 1D bar by SNOPD with $n = 1$ and NMBC and $G = 0.25 \sim 2.0$

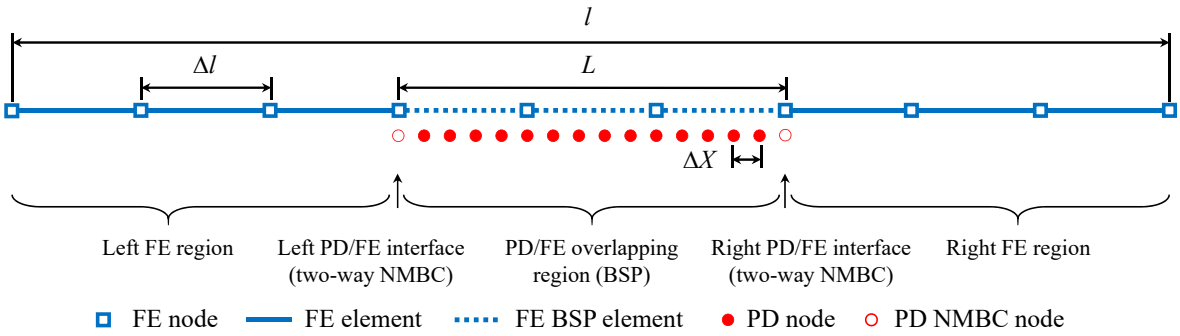


Figure 10.17. Schematic illustration of computational setup for concurrent multiscale simulation on wave propagation in a 1D bar

is schematically sketched in Figure 10.17. A horizon radius $\delta = 3\Delta X$ is adopted for PD simulations. For SNOPD, the stabilization parameter is $G = 0.5$. The explicit central difference time integration algorithm with a constant time step size $\Delta t = 0.02$ is employed for both the PD and FE simulations. We investigate the performance of the proposed concurrent multiscale framework with both BBPD and SNOPD formulations in this 1D wave propagation example. Note that the coupling between these two scales are realized by NMBC and bridging-scale projection (BSP) as introduced in previous sections.

10.3.1 Coupled BBPD/FEM in 1D

10.3.1.1 Performance of the two-way NMBC

First, we test the performance of the two-way NMBC, which transmits both incoming and outgoing waves for the PD region. Note that outgoing waves are essentially absorbed by NMBC at the PD/FE numerical interfaces. As illustrated in Figure 10.18 (a), a right-traveling harmonic wave profile with a long wavelength $\lambda = 1,000$ is imposed initially to the left end of the FE domain. Since we are mainly focused on testing the two-way NMBC, the coupling through BSP is disabled in this test. Therefore, the coarse-scale FE nodal values in the overlapping region is obtained by solving FE equations of motion instead of BSP with

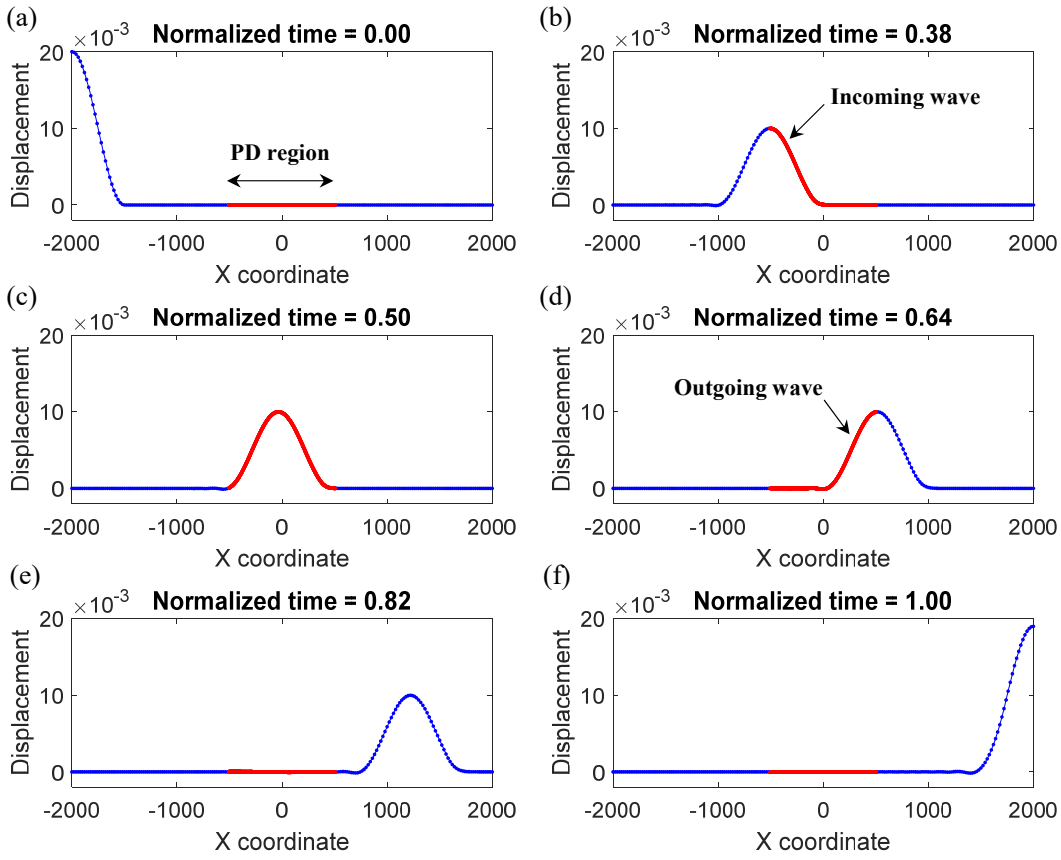


Figure 10.18. Wave propagation in the 1D bar simulated by BBPD/FEM with two-way NMBC and disabled BSPWave propagation in the 1D bar simulated by BBPD/FEM with two-way NMBC and disabled BSP

fine-scale PD solutions. Figure 10.18 (b) shows that incoming wave from the FE region on the left is successfully transmitted into the PD region through the left NMBC nodes. Then the wave continues traveling towards the right direction and is fully absorbed at the right NMBC nodes as shown in Figure 10.18 (c) $\sim$ (f). Therefore, the two-way NMBC in this test has served its purpose of eliminating the outgoing wave and transmitting the incoming wave. It is worth noting again that the coupling from PD to FE in the overlapping region through BSP is disabled. However, the FE solution matches well with the PD solution in this region such that the outgoing wave appears to be transmitted to the FE region on the

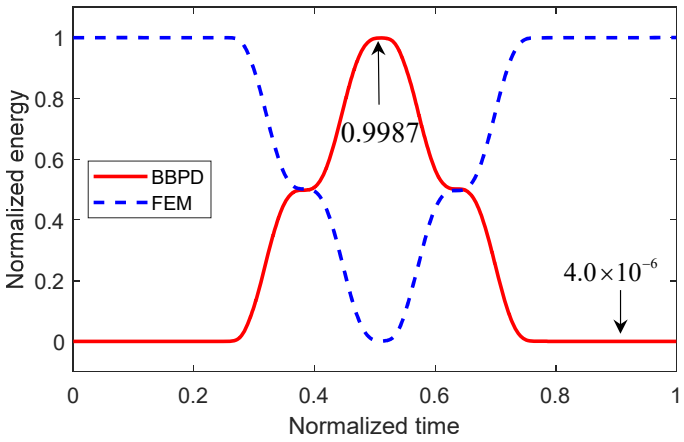


Figure 10.19. Total energy histories in BBPD and FE regions for wave propagation in the 1D bar with two-way NMBC and disabled BSP

right. If the wave is initiated from the PD region, then the FE solution will remain zero due to the disabled BSP.

To quantitatively investigate the efficiency of the two-way NMBC, we measured the total energies in PD model and FE model as a function of time. Note that the FE energy measurement excludes the overlapping region. The normalized total energy histories are shown in Figure 10.19. The stages that wave passing through the numerical interfaces can be clearly identified in this figure. It shows that the NMBC on the left interface transmitted 99.87% incoming wave energy into the PD region, while the remaining energy in PD reduced to only 4.0×10^{-6} by the NMBC on the right artificial interface, which absorbed the outgoing wave. Therefore, the two-way NMBC has demonstrated a remarkable efficiency in this benchmark problem.

10.3.1.2 Performance of the bridging scale projection

Next, we study the performance of the fully coupled concurrent multiscale framework by enabling the BSP, which projects the fine-scale PD solutions onto the coarse-scale FE nodes in the overlapping region. To begin with, we impose a monochromatic long wave profile with

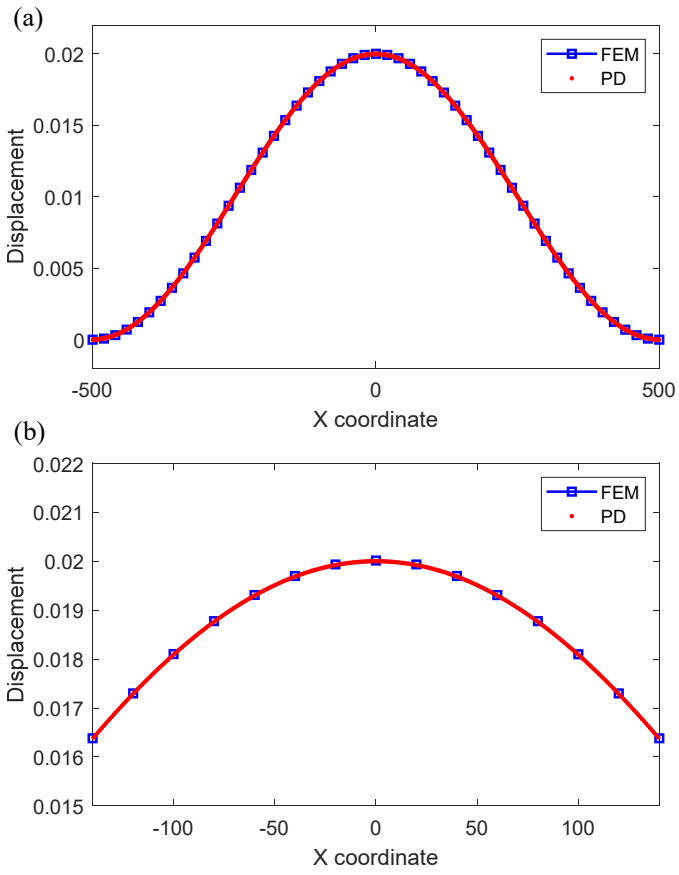


Figure 10.20. Initially imposed long wave for the study on BSP: (a) a view in the PD/FE overlapping region $x \in [-500, 500]$ and (b) an enlarged view $x \in [-140, 140]$

$\lambda = 1,000$ initially to the PD model, which is illustrated in Figure 10.20. The imposed wave is also resolvable at the coarse scale since its wavelength is long enough compared to FE element size. As shown in Figure 10.20 (a) and (b), BSP effectively projects the imposed PD nodal displacements onto the coarse-scale FE elements within the overlapping region. The simulated wave propagation in this case is shown in Figure 10.21 and note that the time in this figure is normalized. It can be seen that the wave propagates smoothly in the bar as if there is no numerical interface.

To check the efficiency of the coupling scheme, we plotted total energy histories in PD and FE models as shown in Figure 10.22. Similarly, calculation of the FE total energy excludes

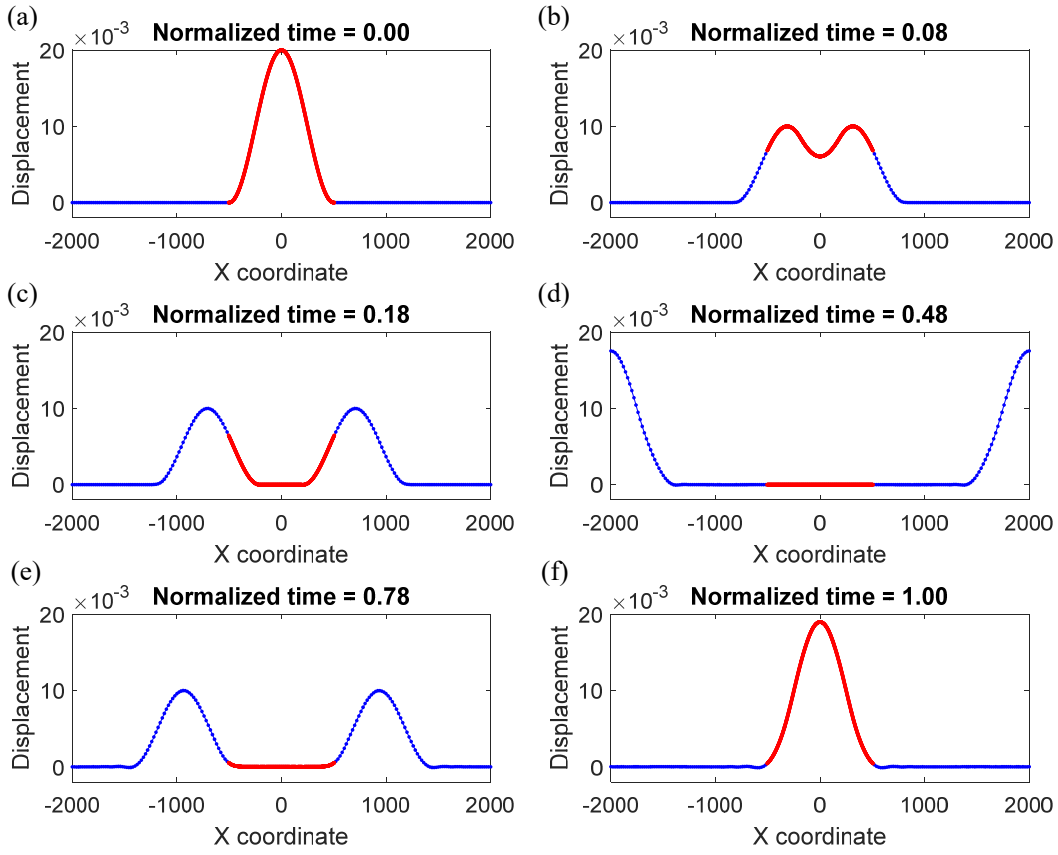


Figure 10.21. Long wave propagation in the 1D bar simulated by BBPD/FEM with two-way NMBC and BSP

the part in the overlapping region. It shows that the coupling scheme efficiently transmits the wave between PD and FEM models. For example, around 99.74% of the initial wave energy is transmitted to FE model through the BSP. In the meantime, NMBC reduces the remaining energy in PD model to a negligible level of 5.0×10^{-7} by absorbing the outgoing waves.

Robustness of the proposed coupling scheme is further demonstrated by extending the total simulation time such that waves can travel in the bar back and forth and pass through the numerical interfaces a couple of times. The normalized energy histories in this case are shown in Figure 10.23. It shows that the total energy transferred into the FE model reduces

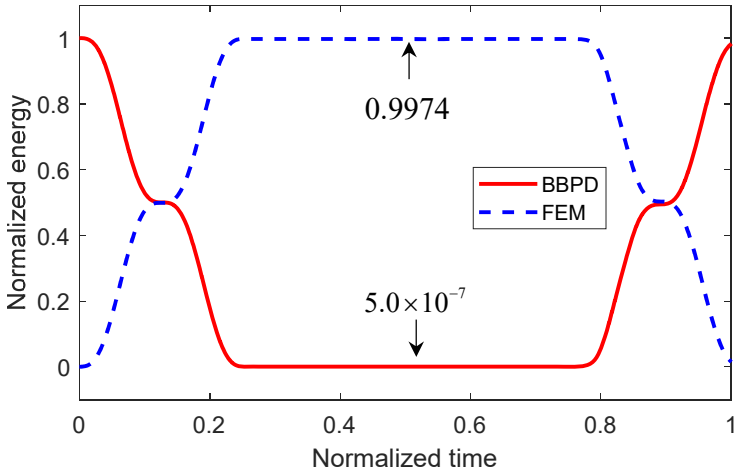


Figure 10.22. Total energy histories in BBPD and FE regions for long wave propagation in the 1D bar with two-way NMBC and BSP

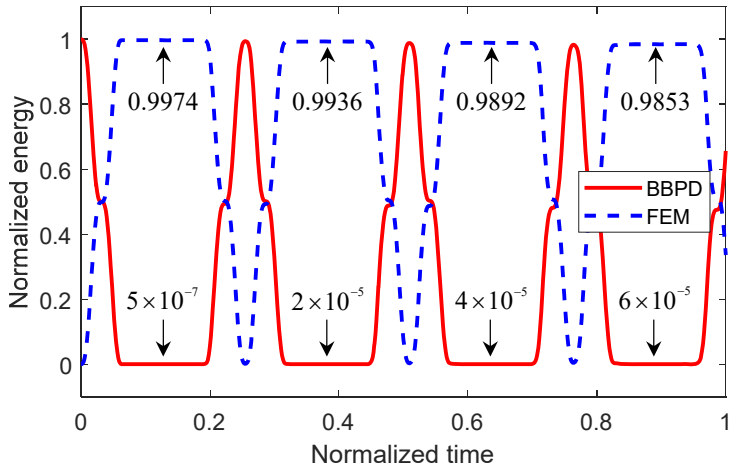


Figure 10.23. Extended total energy histories in BBPD and FE regions for long wave propagation in the 1D bar with two-way NMBC and BSP

by a tiny amount each time the waves passing through the artificial interfaces. Also, the normalized remaining energy in PD model increases by an even smaller amount. Overall, the exchange of energy between the PD and FE models remains stable and robust during the entire extended simulation.

10.3.1.3 Initial wave with both long and short wavelengths

To further demonstrate the capability of the coupled concurrent multiscale framework, we impose an initial wave that contains two components: one with a long wavelength of $\lambda = 1,000$ and the other with a short wavelength of $\lambda = 20$, which cannot be resolved by the coarse-scale FE model but can be fully captured by the fine-scale PD model. Figure 10.24 illustrates this mixed long/short initial wave profile. Note that this initial wave profile is imposed only on the PD model. Figure 10.24 (b) clearly shows that only the long wave component is projected onto the FE model in the overlapping region. From another point of view, the BSP acts effectively as a low-pass filter that only allows the wave components that can be resolved at coarse scale to be projected onto the FE model while filtering out the finer wave components.

Profiles of the mixed wave propagation simulation at different time step are illustrated in Figure 10.25 (a) $\sim$ (f). It shows that only the long wave passes through the numerical interfaces. The short waves are absorbed by the NMBC and therefore lost in the simulation. A comparison between Figure 10.25 (f) and Figure 10.25 (a) clearly demonstrates this loss of information. In this work, we mainly focus on long waves since they travel faster in general and get reflected back earlier at the physical boundaries. Hence, it is adequate to ignore reflections of short waves into the fine-scale region. However, for cases that short-wave reflections cannot be ignored, we can enrich the coarse-scale FE formulation to implicitly store the unresolvable information and transmit them back into the fine scale when it is necessary. Such a scheme has been proposed in an atomistic to continuum coupling approach, please refer to Chirputkar and Qian (Chirputkar and Qian, 2008) for details.

The normalized total energy histories with extended simulation time for the mixed wave propagation case are plotted in Figure 10.26. It shows that the energy associated with short wave components is lost in the system when the initial wave passing through the numerical interfaces for the first time. In this work, we assume that these high-frequency

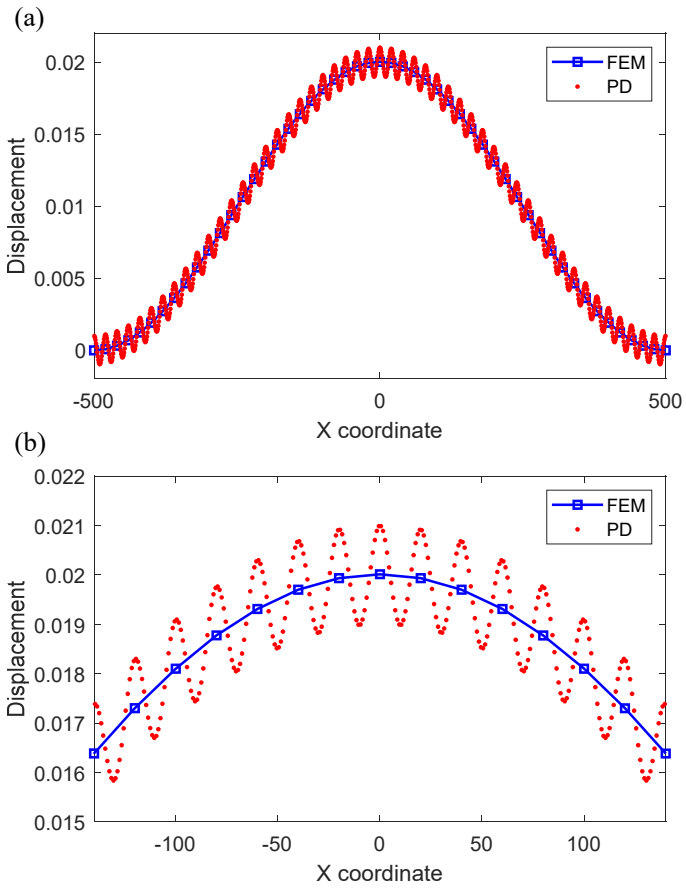


Figure 10.24. Initially imposed mixed long/short waves for the study on BSP: (a) a view in the PD/FE overlapping region $x \in [-500, 500]$ and (b) an enlarged view $x \in [-140, 140]$

wave components are dissipated during propagation due to material damping. After that, the total energy in the system, which is associated to the long wave component, as well as its exchange between PD and FE models remain stable for the rest of the simulation.

10.3.2 Coupled SNOPD/FEM in 1D

Finally, we proceed to the concurrent multiscale framework by coupling SNOPD/FEM. As a reminder, we set the SNOPD stabilization parameter $G = 0.5$ in this example. Since the coupling scheme with BBPD formulation is already demonstrated extensively, we directly study the fully coupled SNOPD/FEM in this case to avoid repetition.

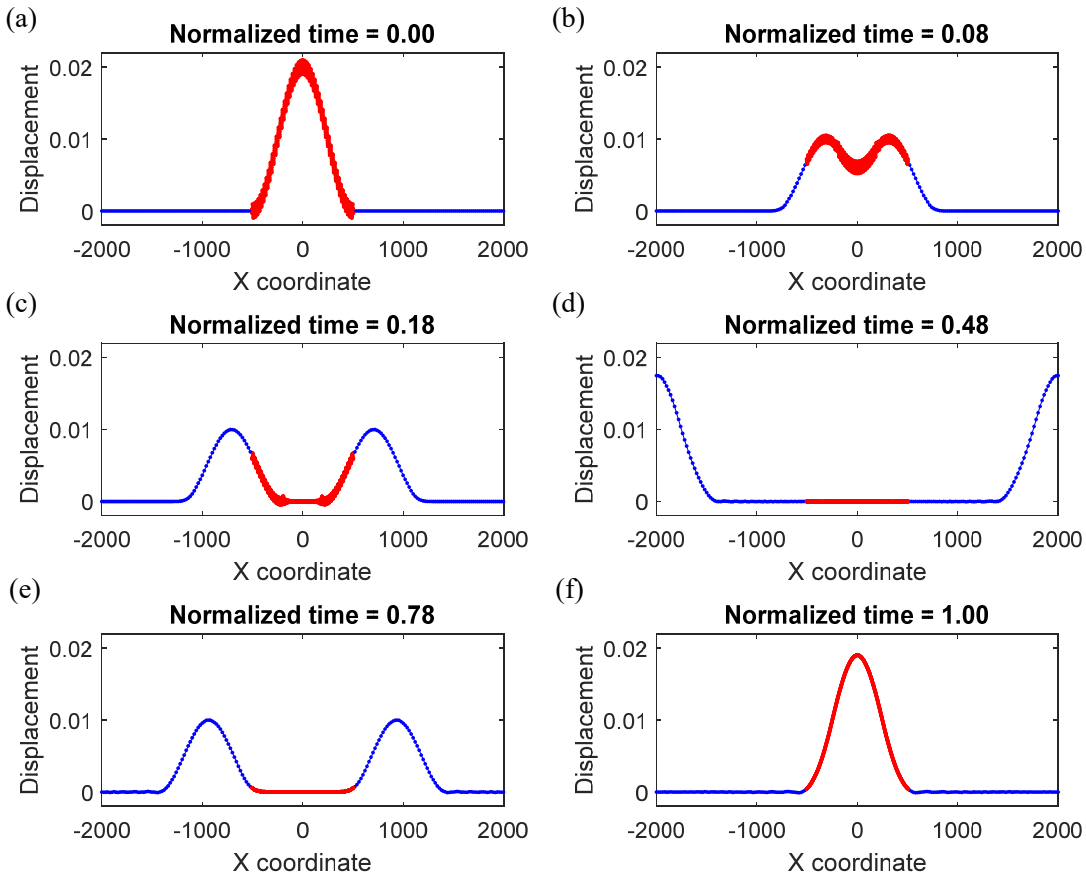


Figure 10.25. Mixed long/short waves propagation in the 1D bar simulated by BBPD/FEM with two-way NMBC and BSP

As shown in Figure 10.27 (a), we imposed a right-traveling initial wave profile to the left end of the FE model. The wavelength is $\lambda = 1,000$. This initial condition is the exactly same with the one shown in Figure 10.18. However, in this case the SNOPD and FEM are fully coupled through the two-way NMBC and BSP. Figure 10.27 (b) $\sim$ (f) show that the prescribed wave travels smoothly in the FE and PD models despite the presence of artificial interfaces.

Efficiency of the numerical interfaces is illustrated in Figure 10.28. Note that we extended the simulation time to test the stability of the coupling scheme with SNOPD and the FE energy does not account for the overlapping region. The peak normalized total energies

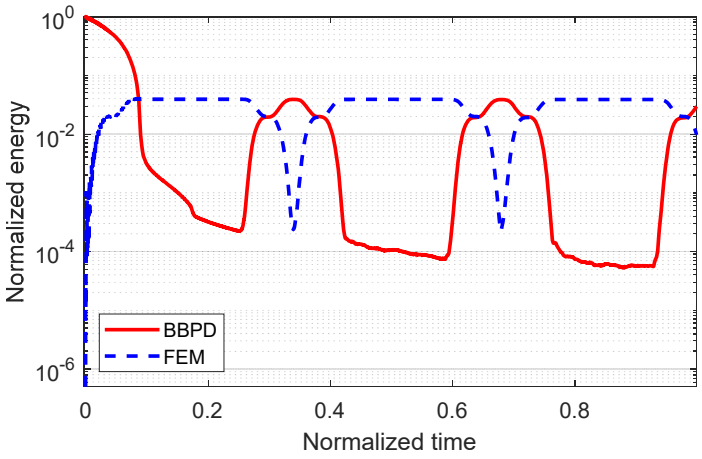


Figure 10.26. Extended total energy histories in BBPD and FE regions for mixed long/short waves propagation in the 1D bar with two-way NMBC and BSP

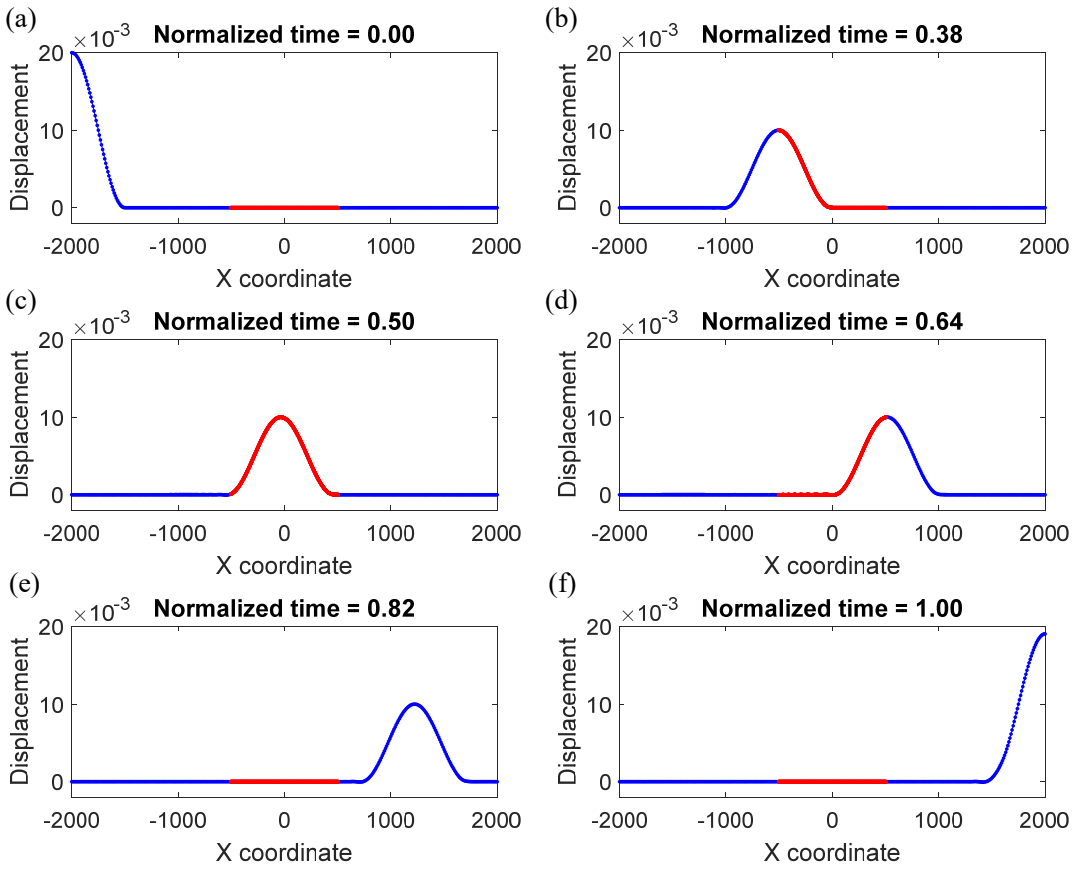


Figure 10.27. Wave propagation in the 1D bar simulated by the concurrent multiscale framework with coupled SNOPD/FEM

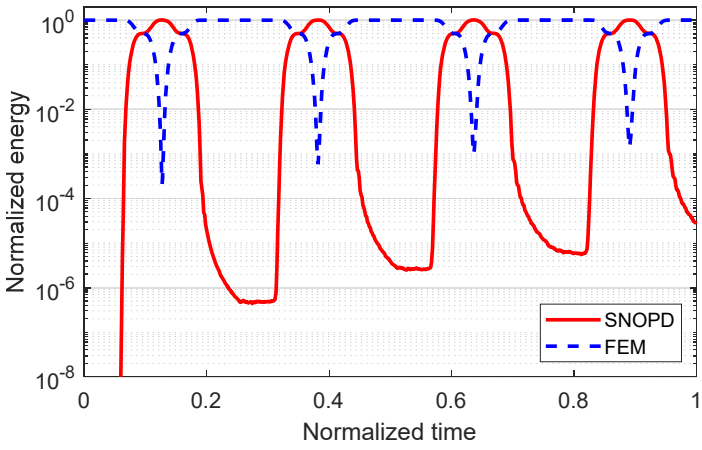


Figure 10.28. Extended total energy histories in SNOPD and FE regions for long wave propagation in the 1D bar

in both FE model and PD model remain a constant very close to 1 during this extended simulation. The remaining energy in PD model is below 1.0×10^{-5} when the wave is fully transmitted into the non-overlapping region FE model.

With the results presented in this test, we have shown that the proposed concurrent multiscale framework performs well in the 1D wave propagation problems with both the BBPD and SNOPD formulations. The developed coupling scheme based on the two-way NMBC and BSP demonstrated a high efficiency as well as exceptional stability and robustness.

10.4 Plane wave propagation in a 2D plate

10.4.1 Unidirectional wave propagation

In this 2D example, we first study unidirectional plane wave propagation. The geometrical dimensions and multiscale computational configuration are illustrated in Figure 10.29. The plate has a dimension of $800 \times 40 \times 1$ and is modeled by the 2D plane stress formulation. The material properties are the same as the previous 1D examples except that the Poisson's ratio is $\nu = 1/3$.

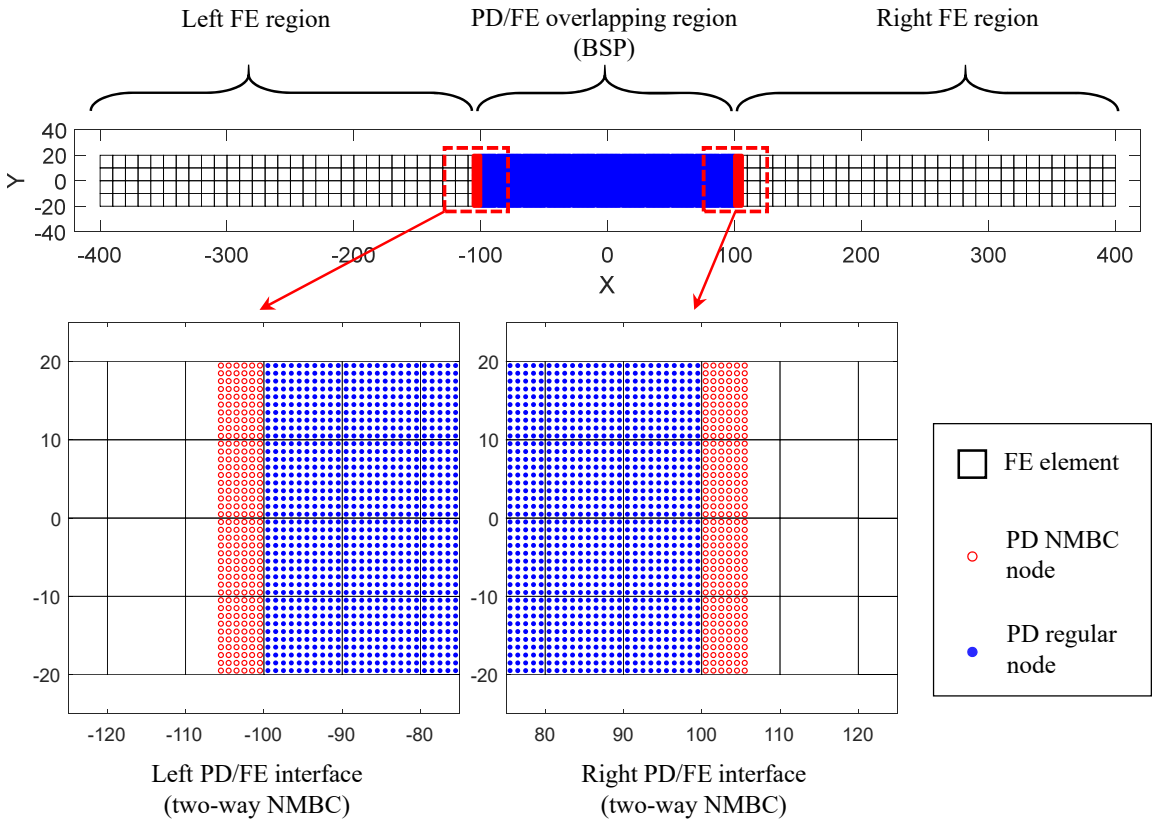


Figure 10.29. Geometrical dimensions and multiscale computational configuration for the unidirectional plane wave propagation example

In the coarse-scale FE model, the entire plate is discretized by uniform 4-node bilinear quadrilateral (Q4) elements with an element edge length $\Delta l = 10$ in both directions. As shown in Figure 10.29, the region within $x \in [-100, 100]$ and $y \in [-20, 20]$ is modeled by the fine-scale BBPD with a uniform nodal spacing $\Delta X = 1$. The PD horizon radius is $\delta = 3\Delta X$ in this case. For both FEM and PD simulations, we employ a constant time step size of $\Delta t = 0.02$.

As illustrated in Figure 10.30 (a), we first prescribe a unidirectional right-traveling plane wave profile with a wavelength of $\lambda = 200$ in the x direction to the left end of FE model. For each timeframe shown in Figure 10.30, both contours of the x-component displacement field and the x-displacement profile along the x-axis ($y = 0$) are plotted. In this simula-

tion, the initial wave passes through the PD region twice. The smooth wave propagation shown in Figure 10.30 (b) $\sim$ (e) indicates that both the two-way NMBC and the BSP work properly. To validate the result obtained by the coupled simulation, we extracted FE nodal displacement history at node $x = 200, y = 0$ and compared to the solution obtained by a FE-only model with a finer mesh. The comparison is shown in Figure 10.31 and an excellent agreement between the two solutions is observed.

Next, we impose a unidirectional plane wave profile with two wavelength components to the PD model. The long and short wavelengths are $\lambda = 200$ and $\lambda = 10$, respectively. The results are shown in Figure 10.32 (a) $\sim$ (e). Similar to the 1D case, only the long wave passes through the numerical interfaces and transmits into the FE regions. The short-wave components are absorbed by the two-way NMBC effectively.

10.4.2 Multidirectional wave propagation

Finally, we study the multidirectional plane wave propagation in 2D. The geometrical dimensions and computational configuration are illustrated in Figure 10.33. Dimensions of the plate are $800 \times 800 \times 1$. The same material parameters in the unidirectional wave case are employed. In the coarse-scale model, 1,600 Q4 elements with an edge length $\Delta l = 20$ is employed. The middle part of the plate within $x \in [-100, 100]$ and $y \in [-100, 100]$ is discretized by around 45,000 evenly distributed nodes in the fine-scale SNOPD model. The PD nodal spacing is $\Delta X = 1$ and the horizon radius is $\delta = 3\Delta X$. Figure 10.33 (b) $\sim$ (e) provide enlarged views showing the corner NMBCs at the PD/FE numerical interface. The stabilization parameter for SNOPD is $G = 1$. A constant time step size of $\Delta t = 0.02$ is employed for both FEM and PD simulations.

We impose an initial wave profile to the PD model, which is given by

$$u\left(r_I, t = 0\right) = \begin{cases} u_0 \left(\cos\left(\pi r_I / r_{\max}\right) + 1\right) & r_I \leqslant r_{\max} \\ 0 & r_I > r_{\max} \end{cases} \tag{10.2}$$

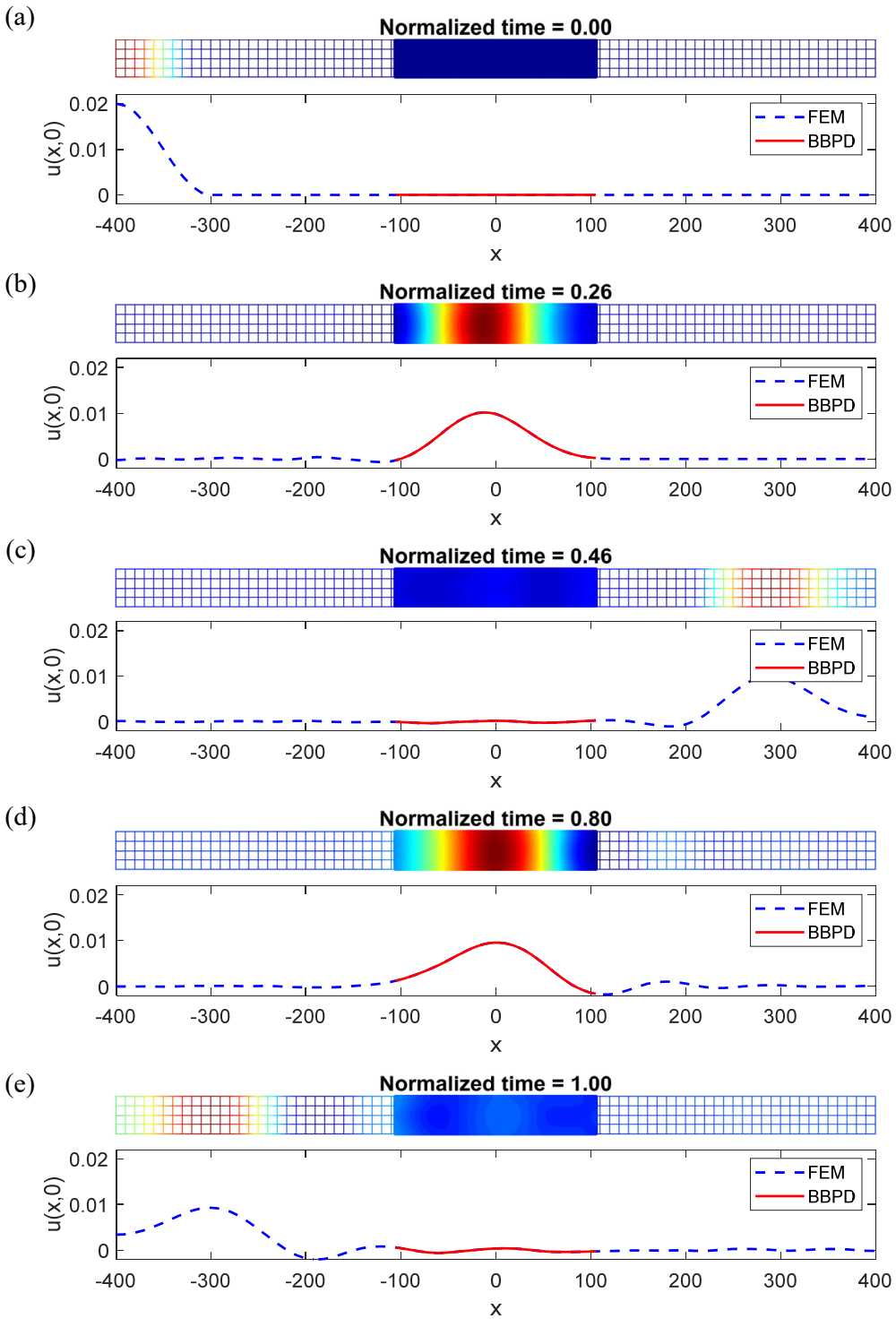


Figure 10.30. Timeframes for the unidirectional long plane wave propagation example

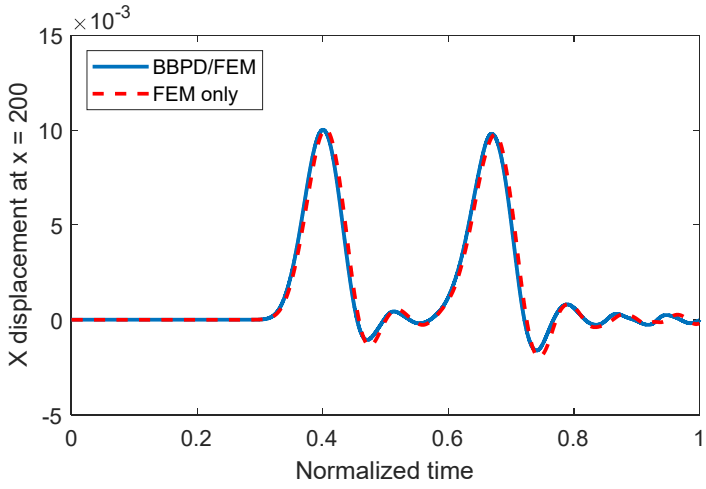


Figure 10.31. Histories of displacement in the x direction at FE node located at $x = 200, y = 0$

where $u_0 = 0.005, r_I = \sqrt{x_I^2 + y_I^2}$ and $r_{\max} = 100$.

The concurrent multiscale simulation results are shown in Figure 10.34. Displacement fields in the x direction at different time steps are plotted. In Figure 10.34, the FE displacement contours are plotted on the element edges while PD displacements are directly plotted on nodes. The initially imposed long wave propagates from the center of the plate towards the edges. Wave transmissions between the FE-only region and the PD/FE overlapping region are smooth. For the purpose of comparison, we also simulated this case by a refined FE-only model with 6,400 Q4 elements. The reference solutions are presented in Figure 10.35. The corresponding timeframes agree well with each other as shown in Figure 10.34 and Figure 10.35, which demonstrates a satisfactory performance of the coupling scheme in 2D for plane wave propagation in arbitrary direction.

Lastly, we prescribe a mixed long/short multidirectional wave to the PD model. The long wave component is the same as the previous. Both the wavelength and the amplitude of the short wave is 10 times smaller than the long wave. Displacement solutions in the x direction are illustrated in Figure 10.36 for different time steps. The short-wave components can be

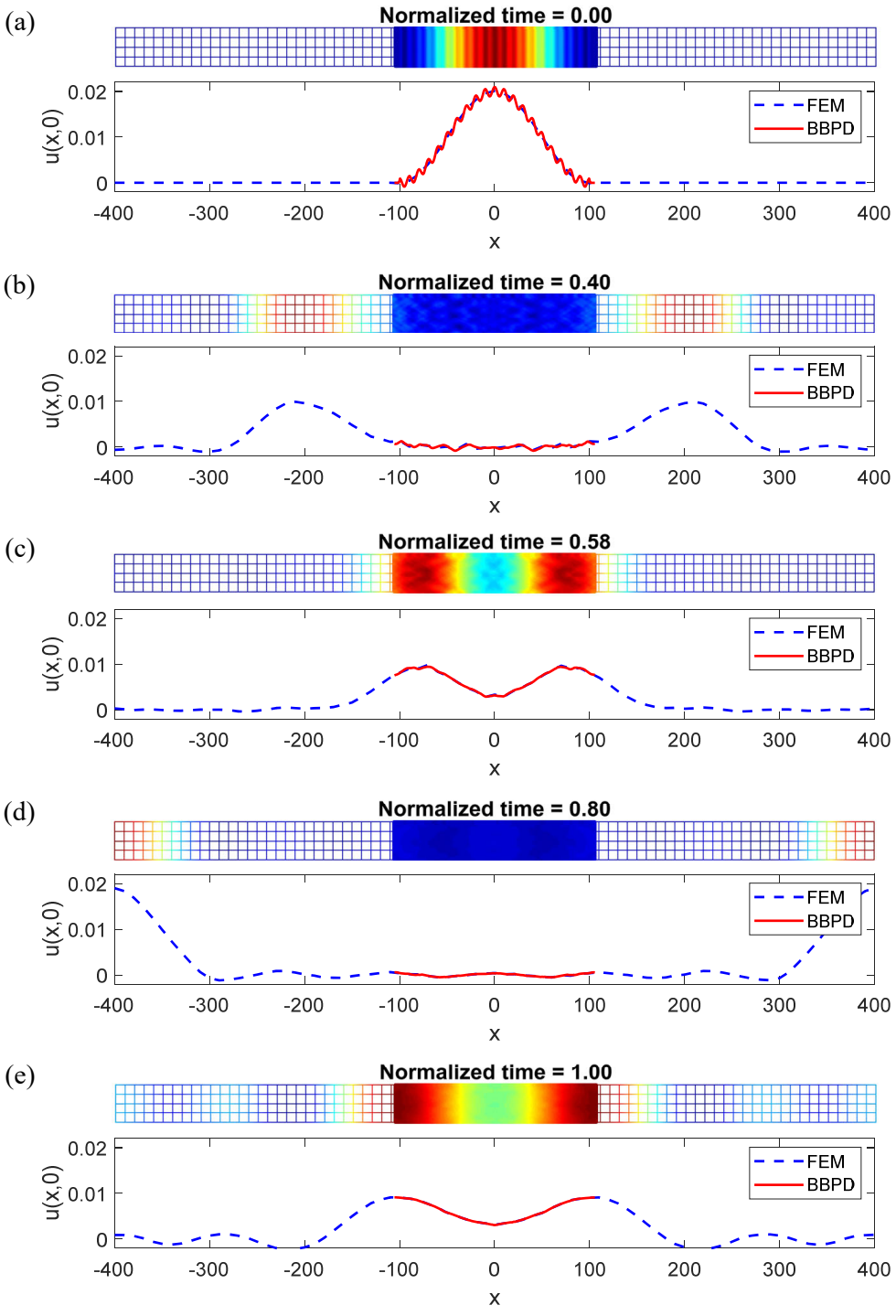


Figure 10.32. Timeframes for the unidirectional mixed long/short wave propagation example

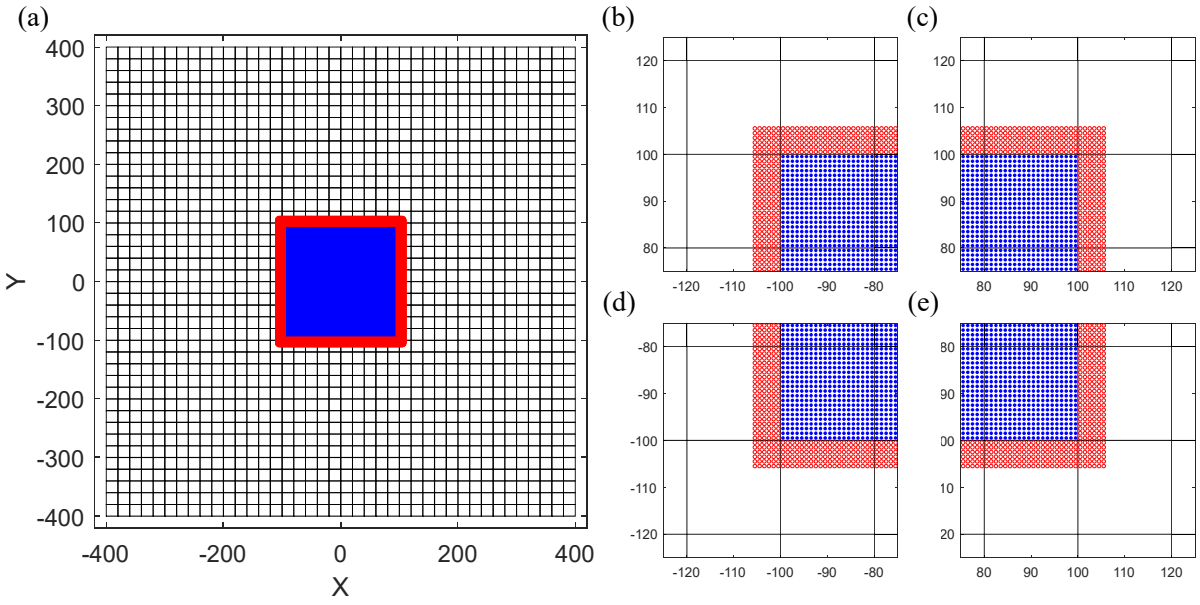


Figure 10.33. Geometrical dimensions and multiscale computational configuration for the multidirectional plane wave propagation example

clearly observed in Figure 10.36 (a) $\sim$ (c). After the waves passing through the numerical interface, differences between the corresponding timeframes shown in Figure 10.36 (d) $\sim$ (i) and Figure 10.34 (d) $\sim$ (i) are hardly noticeable, which indicates that the short waves are fully absorbed by the numerical interface. For dynamic fracture applications, short waves can be created by bond breaking. With the proposed concurrent multiscale framework, these short waves will not be reflected at the artificial interface and therefore cannot contaminate the fine-scale solution.

10.5 Dynamic fracture in a glass plate

To demonstrate the concurrent multiscale method in dynamic fracture applications, we first consider a thin square glass plate fracture benchmark problem studied in (Wada, 2017). For the purpose of comparison, this problem is simulated using both full BBPD and the coupled BBPD/XFEM.

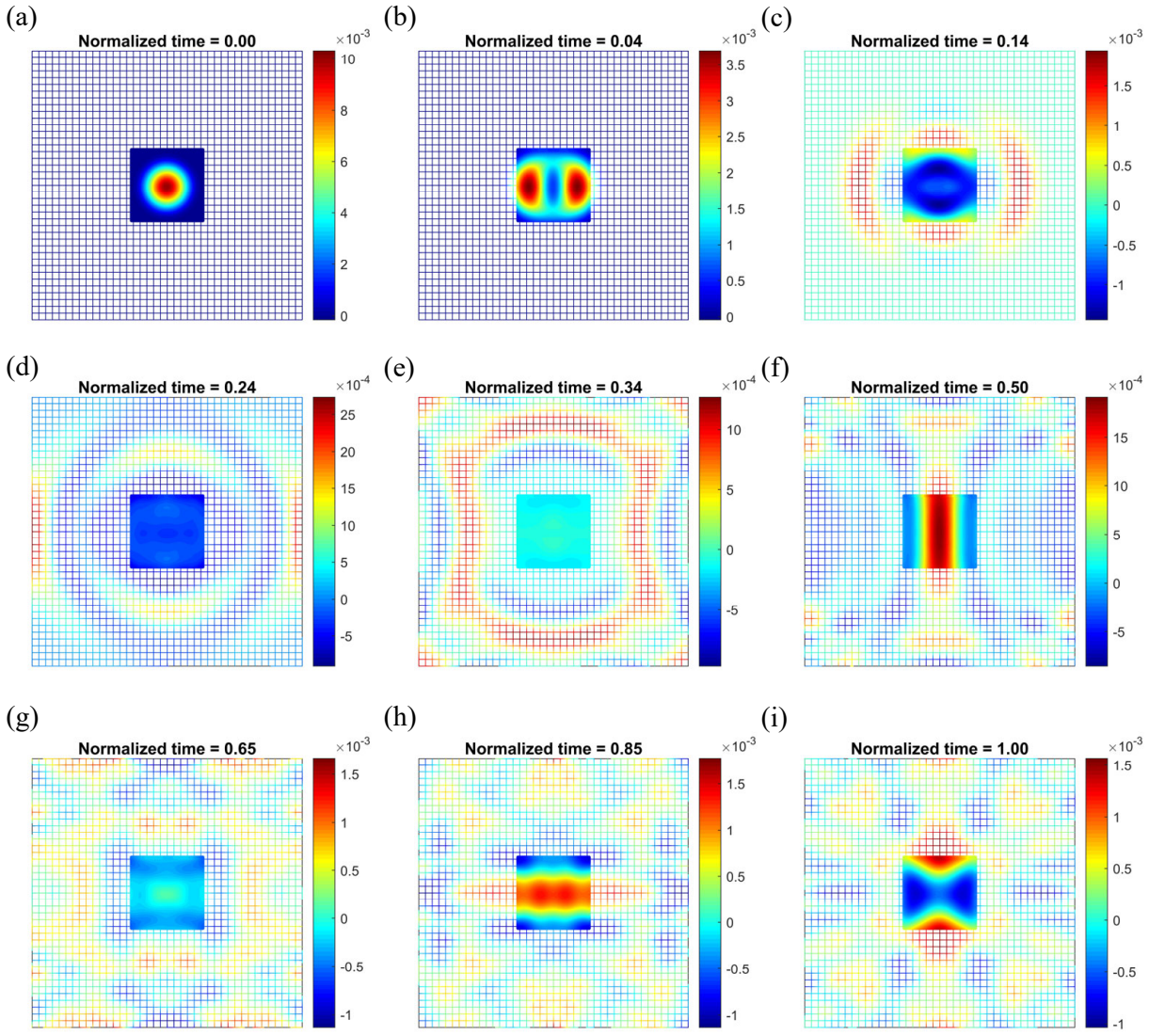


Figure 10.34. Timeframes for multidirectional plane wave propagation simulation

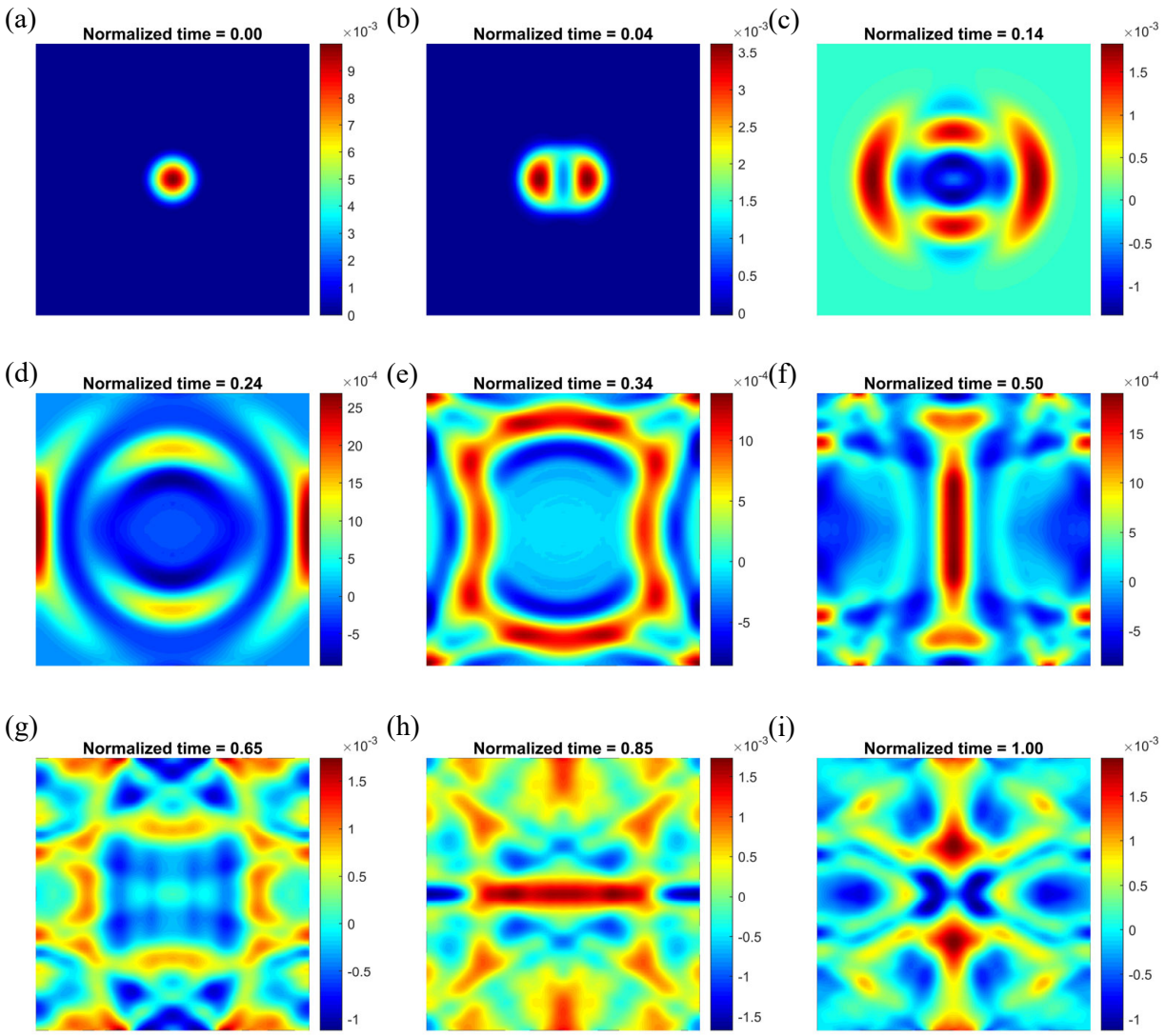


Figure 10.35. Timeframes for multidirectional plane wave propagation based on FEM simulation

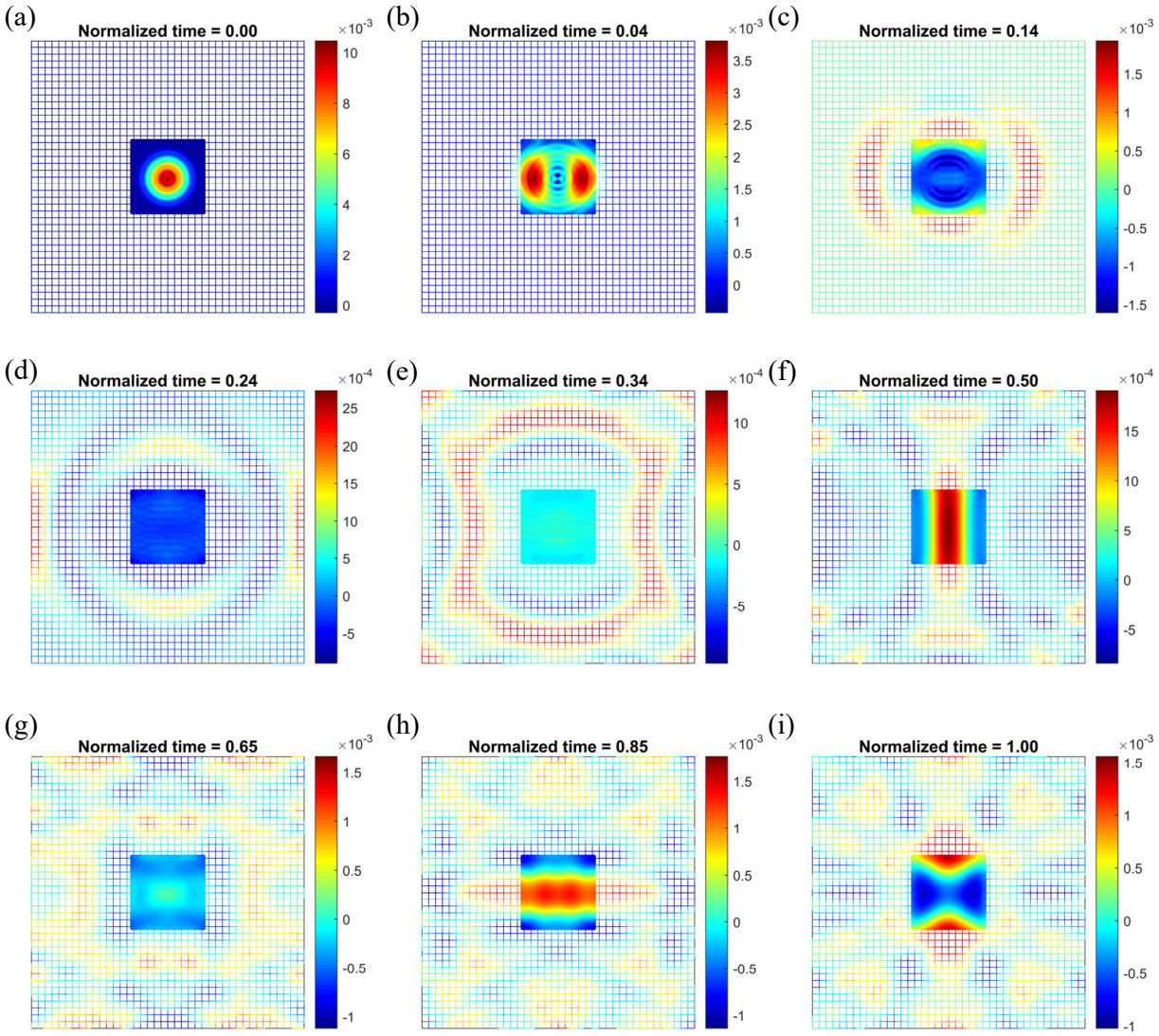


Figure 10.36. Timeframes for multidirectional plane wave propagation with mixed long/short wave components

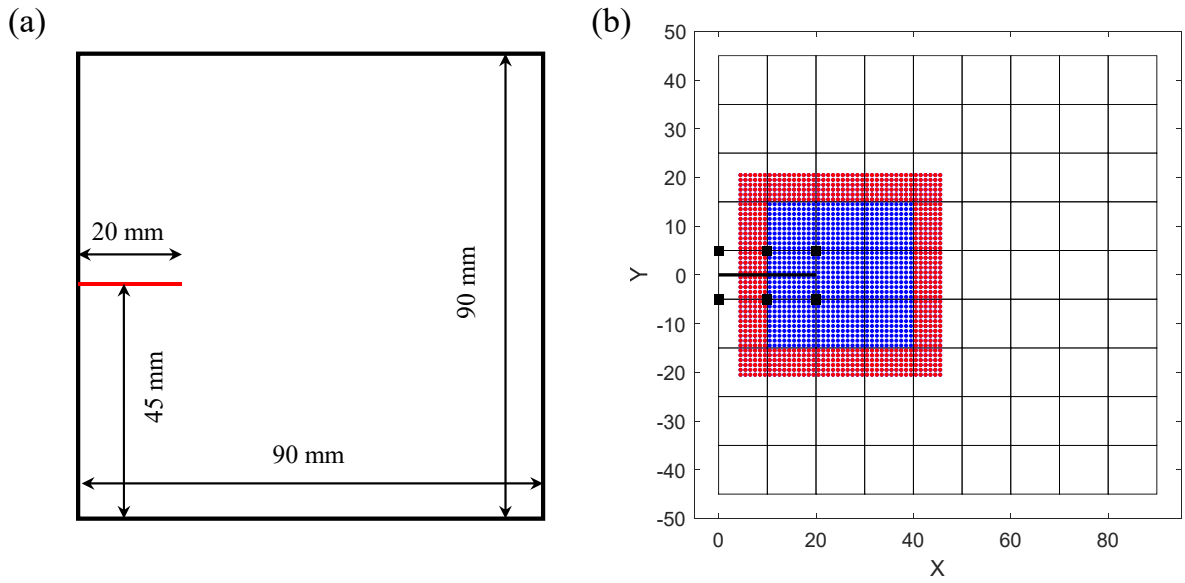


Figure 10.37. Thin glass plate fracture problem: (a) dimensions and (b) computational configuration

10.5.1 Problem statement

Figure 10.37 (a) shows the dimensions of the glass plate considered in this example. The plate is square-shaped with an edge length of 90 mm. Thickness of the plate is 1 mm and the plane stress formulation is employed in simulations. As indicated by the red line in Figure 10.37 (a), a horizontal initial crack with a length of 20 mm is placed at the middle of the left edge. The material properties of the glass plate are: Young's modulus $E = 74$ GPa, mass density $\rho = 2480$ kg/m^3, and critical strain $s_c = 0.0005$. Two types of loading conditions are considered in this example. In the Mode I fracture loading condition, both the bottom and top edge plate are subjected to a constant uniform traction rate of 5,000 MPa/s in the y direction. In the Mode II loading condition, the bottom edge is fixed while a constant velocity of 10 mm/s in $-x$ direction is applied to the top edge.

The initial computational configuration is illustrated in Figure 10.37 (b). The entire computational domain is discretized by a total of 81 four-node quadrilateral, fully integrated elements (Q4) with an uniform element size of 10 mm at the coarse scale. The initial fine-

scale PD region is placed at $x \in [10, 40]$ and $y \in [-15, 15]$ with an uniform nodal spacing of $\Delta X = 1$ mm. The horizon radius for PD is $\delta = 3$ mm. In Figure 10.37 (b), the red dots indicate the NMBC nodes and the initial crack path is represented by the solid line in black color. The initially enriched coarse-scale nodes and elements are mark by solid squares. For both full PD and coupled PD/XFEM simulations, a constant time step of 1.0×10^{-8} s is employed. The total numbers of steps are 150,000 and 426,000 for Mode I and Mode II cases, respectively. Note that in full PD simulation, the entire computational domain is discretized by evenly distributed PD nodes.

10.5.2 Mode I crack results

In the first loading condition, the expected crack path is a horizontal opening crack passing through the middle of the plate, which is also know as the Mode I crack. Figure 10.38 shows the crack propagation process in full PD simulation and the coupled PD/XFEM simulation. For each case, three images are captured at the beginning, the middle, and the end of the simulation. In both Figure 10.38 (a) and (b), the PD nodes are colored by the damage variable such that the crack path can be clearly observed. In addition to that, the deformation is scaled up 100 times in visualization. The results presented in this figure demonstrate a very good agreement between the coupled PD/XFEM simulation and the reference full PD simulation.

Figure 10.39 plots the crack length histories obtained by the coupled PD/XFEM simulation and the reference full PD simulation. In Figure 10.39 (a), the crack length is plotted against the physical time. It shows that the crack propagation starts slightly earlier in the coupled PD/XFEM simulation. However, the crack growth speeds in PD/XFEM and full PD simulations are almost identical as shown in Figure 10.39 (b), where the crack length is plotted against the normalized time. This quantitative comparison shows that the proposed concurrent multiscale coupling method is accurate.

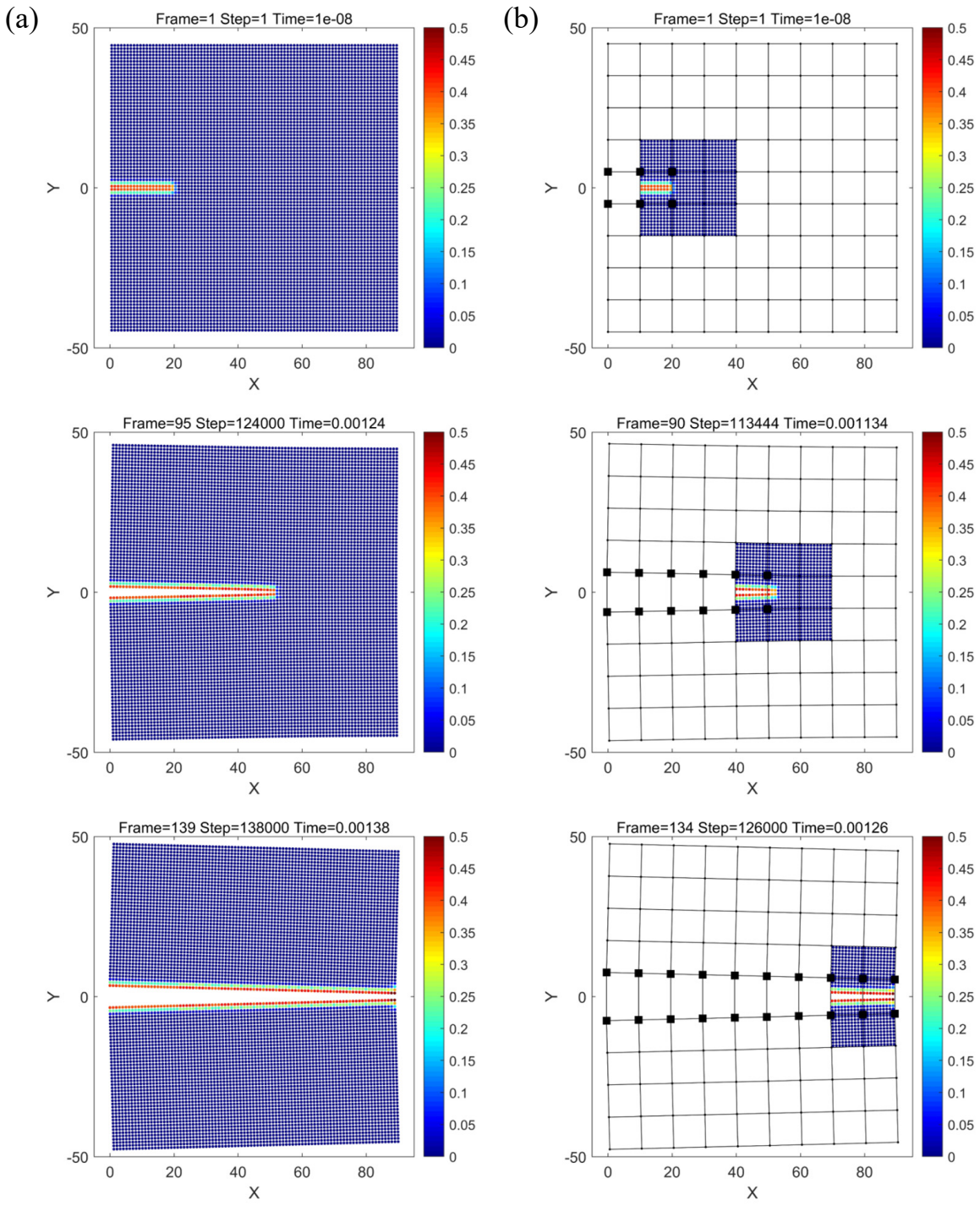


Figure 10.38. Mode I crack propagation in (a) full PD simulation and (b) coupled PD/XFEM simulation

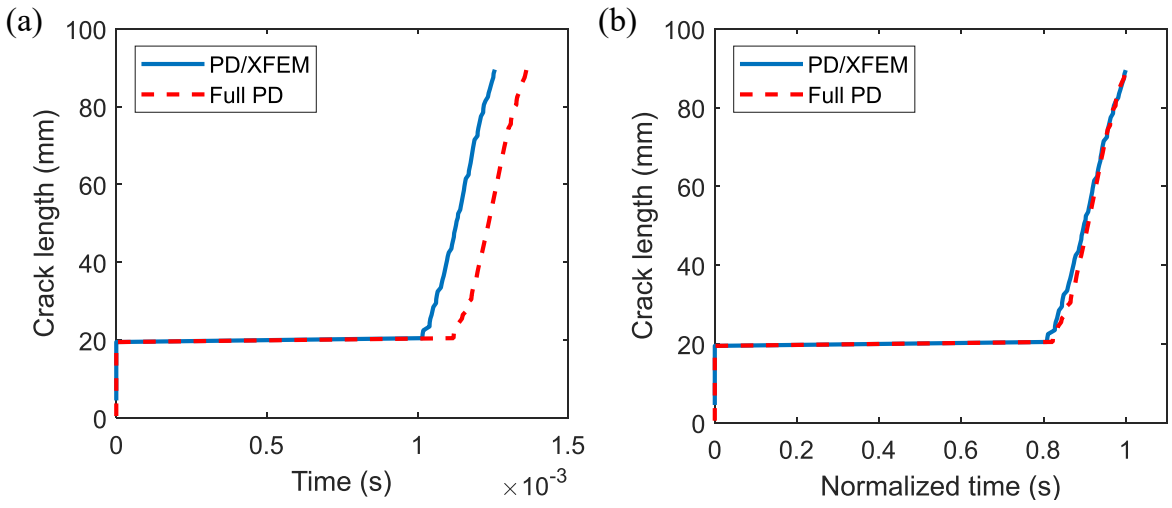


Figure 10.39. Mode I crack length vs. (a) physical time and (b) normalized time

10.5.3 Mode II crack results

Next, we study the second loading condition, which leads to the crack sliding mode or the Mode II fracture. Because of the shear loading condition, the crack is expected to propagate in a diagonal direction.

Similar to the Mode I case, the crack growth process in Mode II is illustrated in Figure 10.40. The full PD simulation results are shown in Figure 10.40 (a), in which a diagonal crack path is observed. The PD region relocation process in the coupled PD/XFEM simulation is demonstrated in Figure 10.40 (b).

A quantitative comparison for Mode II case is presented in Figure 10.41. First, the crack paths are compared between the PD/XFEM and full PD simulations. Almost exactly matched crack paths are shown in Figure 10.41 (a). Finally, the crack lengths are compared in Figure 10.41 (b), which also demonstrated an excellent agreement.

10.5.4 Computational performance

So far we have demonstrated that the proposed concurrent multiscale approach is indeed accurate in fracture simulations. In Table 10.1, the computational performances of both the

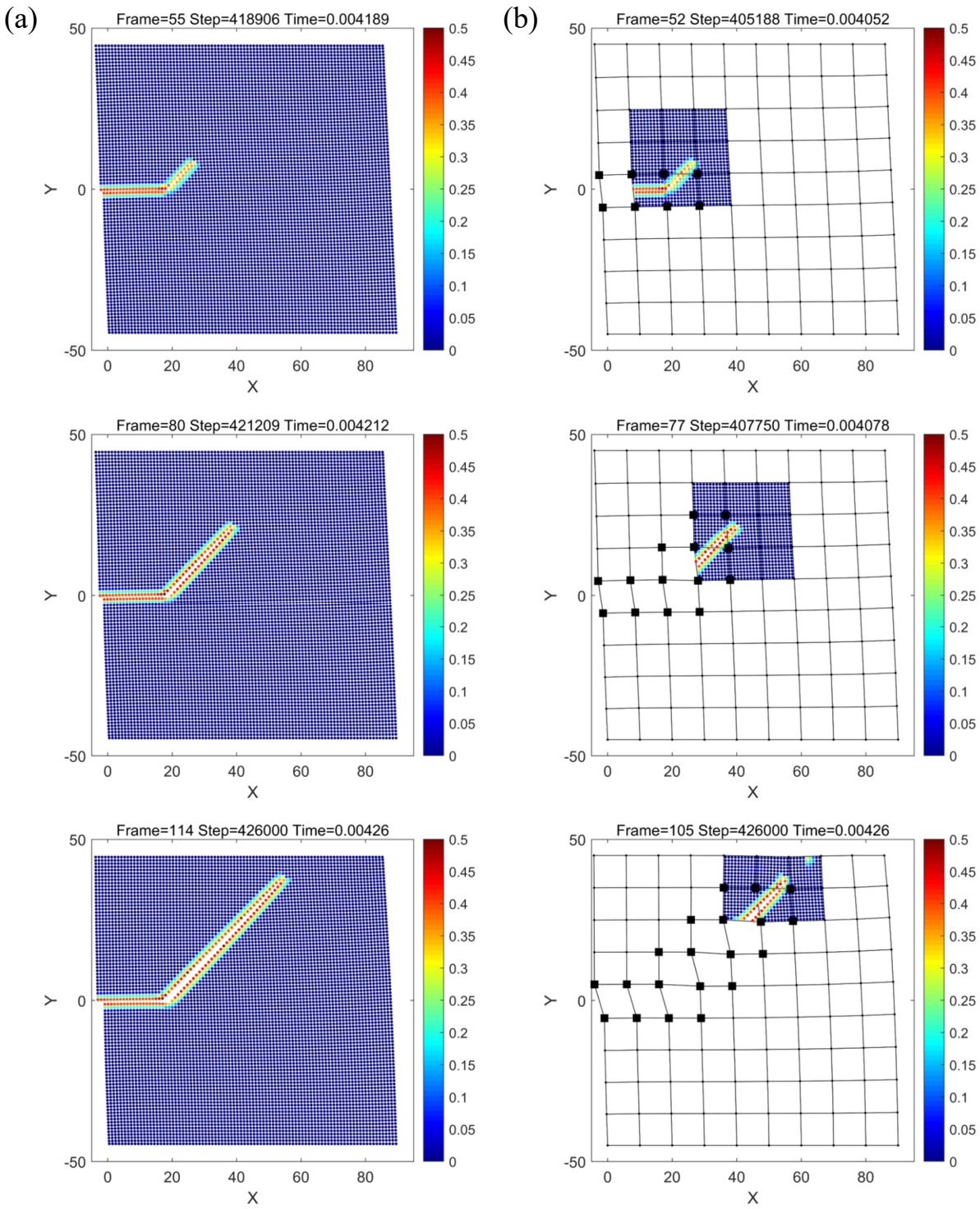


Figure 10.40. Mode II crack propagation in (a) full PD simulation and (b) coupled PD/XFEM simulation

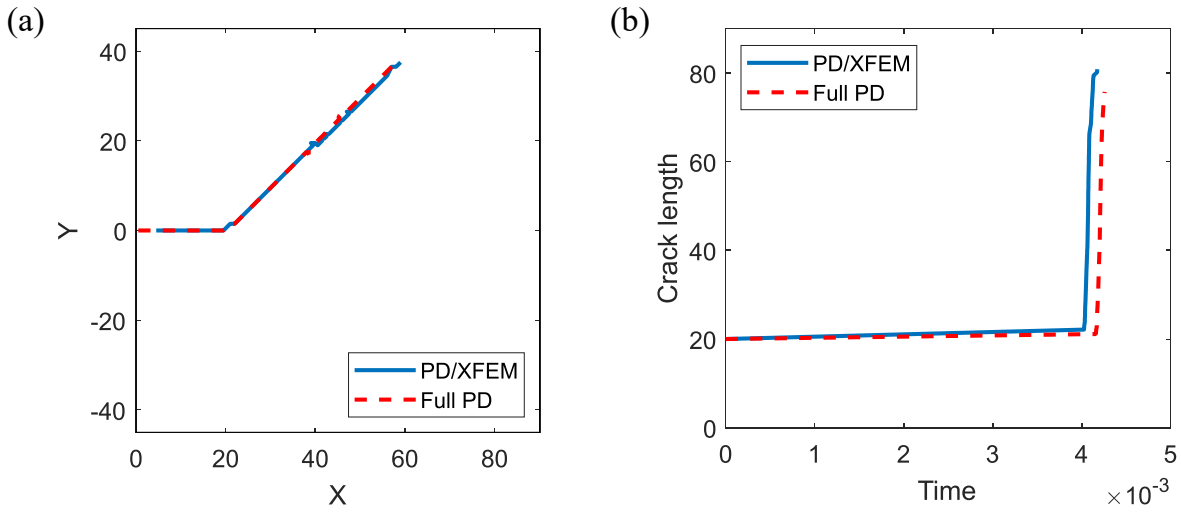


Figure 10.41. Mode II fracture: (a) crack path and (b) crack length vs. time

Table 10.1. Computational performances of coupled PD/XFEM and full PD simulations

Method	**# nodes**	**# DOFs**	**Time**
Coupled PD/XFEM	1,764 + 100	3,928	24.8s
Full PD	8,100	16,200	67.6s
Ratio	1 : 4	1 : 4	1 : 3

coupled PD/XFEM and full PD are presented and compared for the Mode I case. Note that 10 cores of the Intel Xeon CPU E5-2698 v4 2.20 GHz with OpenMP are employed in this testing. It shows the total computing time is linearly proportional to the total number of DOFs. In this particular case, the full PD simulation has 4 times DOFs compared to the coupled PD/XFEM simulation. The computing time in the full PD case is 3 times longer than the coupled PD/XFEM case. Therefore, the concurrent multiscale coupling method also demonstrated a high computational efficiency.

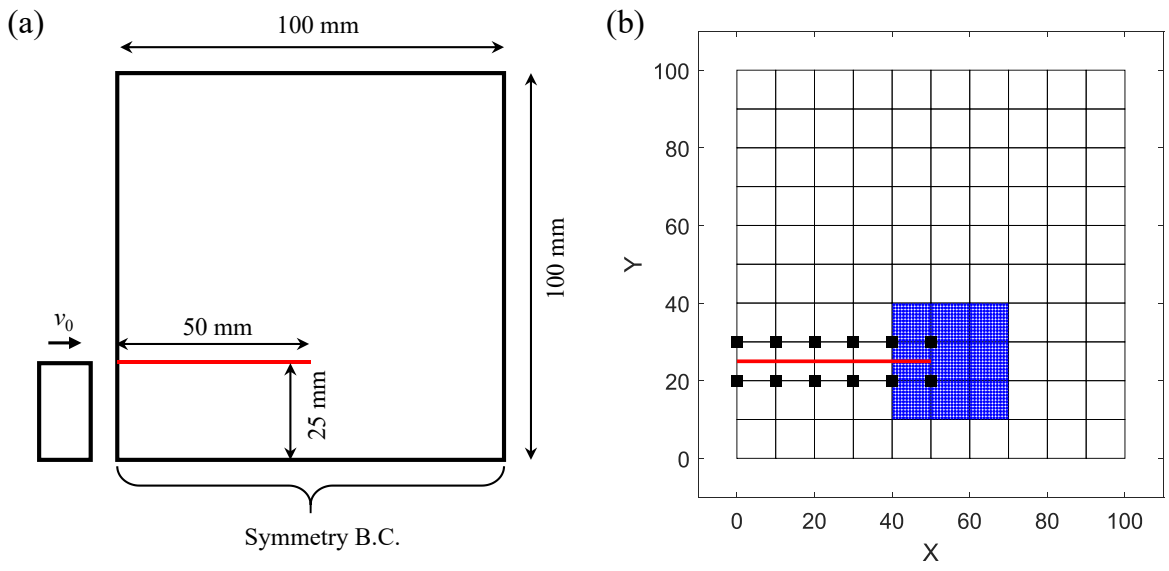


Figure 10.42. Kalthoff-Winkler experiment: a half model (a) dimensions and B.C.s and (b) computational configuration

10.6 Kalthoff-Winkler experiment: a half model

In the last example, we employ the concurrent multiscale method to simulate the Kalthoff-Winkler experiment (Kalthoff and Winkler, 1988), which is also simulated by using full BBPD in Section 7.4.3. Here we study only a half of the original model by considering the symmetry of the problem. The problem statement and computational configuration are illustrated in Figure 10.42. Detailed parameters of this problem can be found in Section 7.4.3. In this example, we employ a structured mesh of 100 Q4 elements of size 10×10 mm^2 in the coarse-scale simulation. For the fine-sale simulation, a uniform nodal spacing $\Delta X = 0.83$ mm is employed and the PD horizon radius is $\delta = 3.001\Delta X$. A time step size of 8.7×10^{-8} s is used in this simulation. To apply the symmetry boundary condition, the bottom edge ($y = 0$) as shown in Figure 10.42 (b) is constrained such that the vertical displacement is zero. The impact loading is approximated by applying a constant velocity in the x direction on the left edge where $x = 0$ and $y \in [0, 25]$.

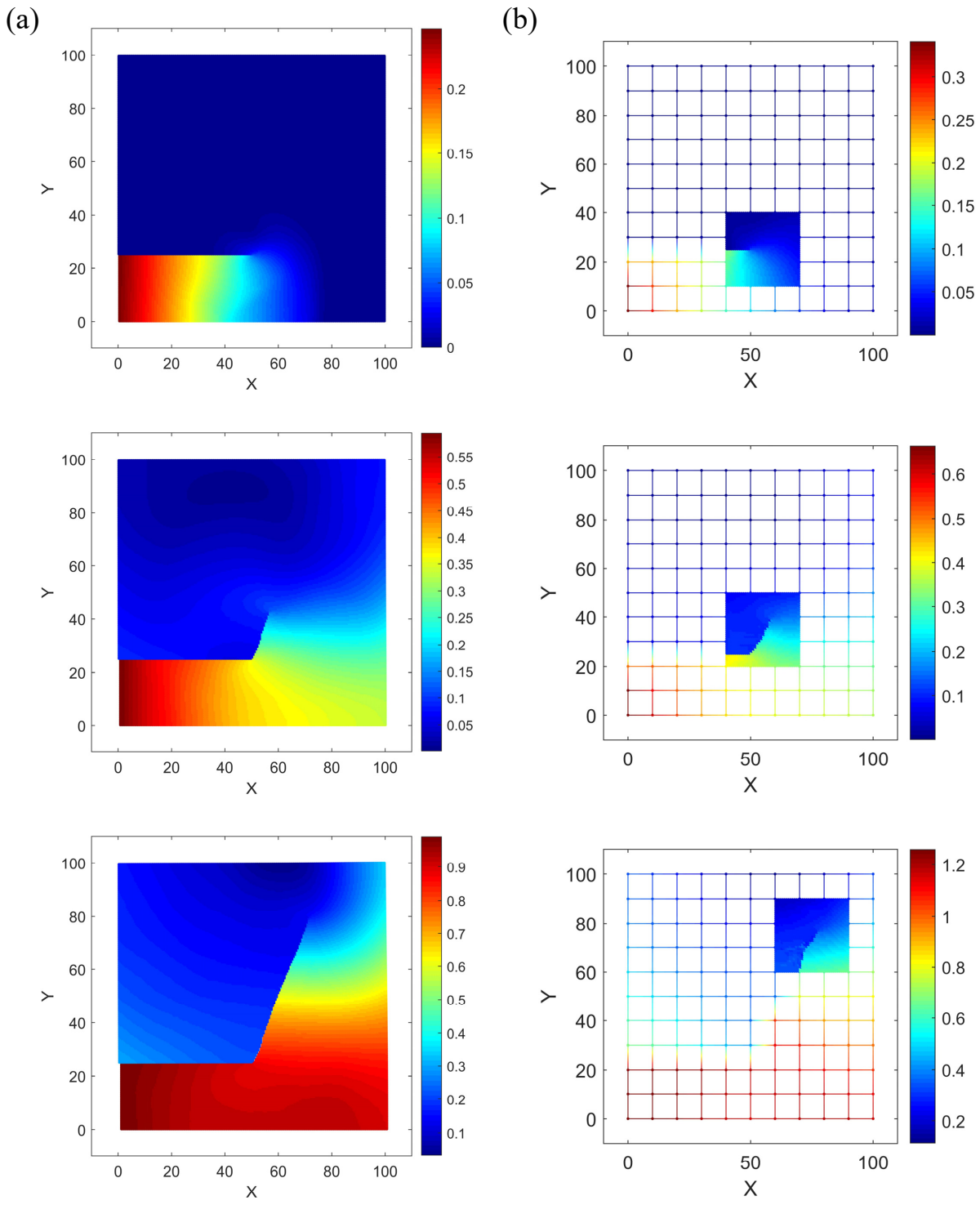


Figure 10.43. Kalthoff-Winkler crack propagation in (a) full PD simulation and (b) coupled PD/XFEM simulation

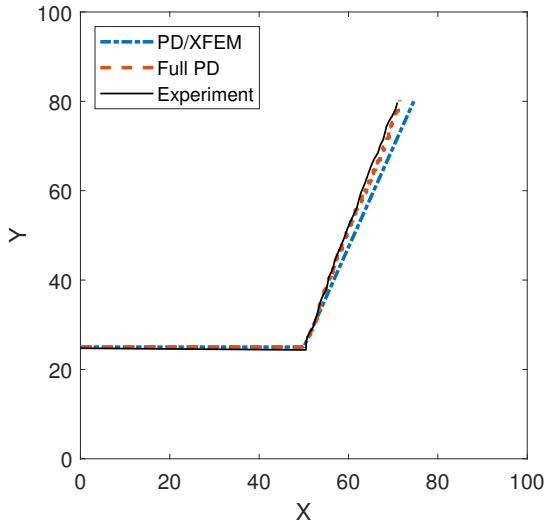


Figure 10.44. Kalthoff-Winkler experiment: comparsion between crack paths

The simulation results are shown in Figure 10.43. For the purpose of comparison, a full PD simulation of the half model is also performed. Figure 10.43 (a) shows the crack propagation obtained by the full PD simulation at three time frames. The nodes are colored by displacement magnitude and the crack path is clearly visible from the high contrast in color. Simulation results from the concurrent multiscale simulation are shown in Figure 10.43 (b). Similarly, snapshots at three different time steps are captured and both the coarse-scale element edges and the fine-scale nodes are colored by displacement magnitude. It can be seen that the fine-scale region is updated adaptively to track the crack propagation. Compared to the full PD simulation, the multiscale simulation well captured the crack propagation in Kalthoff-Winkler experiment but at a much reduced computational cost.

Finally, we present a quantitative comparison on the crack path, which is shown in Figure 10.44. Note that the experimentally observed crack path is extracted from (Song et al., 2008). The full PD simulation shows an excellent agreement with the experiment data. The crack path obtained from the concurrent multiscale simulation is only slightly deviated from the experiment path towards the very end but still matches well. The crack propagation angles are 66° in the concurrent multiscale simulation, 69° in the full PD simulation, and

70° in the experiment, respectively. The simulation results agree well with the experiment and the accuracy of the concurrent multiscale method is demonstrated.

10.7 Summary

In this chapter, we first presented several elastodynamics wave propagation examples in both 1D and 2D to study the performance of the proposed concurrent multiscale framework. The coupling scheme between coarse-scale FE model and fine-scale PD model is systematically investigated and demonstrates a very good performance. Subsequently, dynamic fracture simulations of a thin glass plate under Mode I and II loading cases as well as the Kalthoff-Winkler experiment are carried out to further testing the robustness the coupling scheme as well as the adaptive PD region relocation algorithms. The simulation results show that the proposed concurrent multiscale approach is both accurate and efficient compared to single scale methods. Therefore, we conclude that the main objective of this study (see Section 1.3.4) is achieved.

CHAPTER 11

CONCLUSION AND FUTURE WORK

In conclusion, two key contributions are made in this dissertation to the field of multiscale material failure predictions. The first contribution is the development of a high performance multiscale computational framework for direct numerical simulation on 3D high cycle fatigue life predictions. The second is a concurrent multiscale approach based on coupled Peridynamics and Finite Element Method for dynamic fracture simulations. These two parts are summarized and discussed in the following sections.

11.1 Multiscale high cycle fatigue life prediction

In the first part, a high performance multiscale computational framework is established for direct numerical simulation on 3D HCF applications. 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 $1 \sim 2$ 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 demonstrated the capabilities

The following article was reused in this chapter with permission from the publisher:

1. Zhang, R., S. Naboulsi, T. Eason, and D. Qian (2019). A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing. *Finite Elements in Analysis and Design 166*, 103320. Reuse with permission from *Elsevier*.

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.

Furthermore, it is well known that HCF failures are sensitive to material microstructures. In this work, we conducted a preliminary study on data-driven microstructure-based concurrent multiscale material modeling method. The proposed method is based on the recently developed Self-consistent Clustering Analysis, which is a novel reduced-order multiscale material model derived from Machine Learning techniques. For HCF applications, an efficient solution algorithm is developed to further accelerate the proposed method. Direct concurrent multiscale modeling of complex material microstructures is enabled by the proposed method for HCF simulations at a much reduced computational cost. For future work, one direction is to further study efficient algorithms for both offline and online stages of SCA regarding the computational performance. In addition, it is crucial to develop high performance computing implementations using many core GPUs as well as other parallel computing techniques. Another important direction is to employ the proposed method to study HCF life and failure mechanisms with microstructures of various material systems ranging from metals, ceramics, polymers to composites. It would be very helpful to provide insights in designing high performance materials that can bear longer service life in many applications.

11.2 Current multiscale dynamic fracture prediction

Motivated by the limitations in single scale methods, we established a concurrent multiscale approach to dynamic fracture failure predictions in the second part of this dissertation. The major goal of this study is to preserve accuracy at critical regions while achieving a high computational efficiency. The concurrent multiscale approach is developed based on coupled Peridynamics and Finite Element Method. In the proposed method, coarse-scale FE simulation is conducted over the entire domain of the problem, and coexists with a local fine-scale PD region where crack pre-exists or is expected to initiate. The coupling scheme is accomplished by a bridging-scale projection of the fine-scale PD solution onto the overlapping coarse-scale FE basis functions, and a class of two-way nonlocal matching boundary conditions that transmits long waves from coarse-scale FE domain to fine-scale PD region and eliminates spurious wave reflections at artificial numerical interfaces. NMBC is expressed in a parameterized form that involves interfacial PD displacements and velocities. The non-reflection and wave transmitting conditions are realized by minimizing the associated residual and its higher order derivatives, which are functions of PD wave dispersion relation at the wavelength of interest. To further accommodate the evolving nature of dynamic fracture, an adaptive scheme is established so that PD region is dynamically prescribed to track propagating crack. Accuracy and efficiency of the proposed concurrent multiscale approach are systemically investigated by elastodynamic wave propagation examples. Effectiveness and robustness are further demonstrated by brittle fracture benchmark examples.

For future work, several directions are recommended. First, it is beneficial to develop more robust and versatile fine-scale PD region relocation strategies for dynamic fracture applications. Second, the current work can be further extended to study more complex problems such as crack branching and coalescence, interactions between multiple existing cracks, and cracks initiation at multiple locations. Although we only studied 1D and 2D cases here, the proposed coupling scheme can be extended to 3D problems. In that case, high

performance computing implementations, e.g. using GPUs, are recommended to handle the computational cost due to the extra dimension. Finally, coupling between PD and meshfree particle methods can be established based on the proposed method to avoid generating the mesh, which is difficult and expensive for many industrial applications. In addition, many advantages of meshfree methods can be incorporated into this concurrent multiscale approach.

APPENDIX A

TWO-SCALE DAMAGE MODEL: THE EXACT SOLUTION

For uniaxial proportional fatigue loading condition, the above set of differential equations can be integrated in a closed-form in two steps. The first step is to perform time integrations over only one cycle under the assumption of constant damage to obtain both plastic strain and damage increments per single cycle. Then perform the integration over the whole loading cycles with the assumption that the microscale stress triaxiality is a constant and equals to its maximum during loading. Detailed derivation of the closed-form solution can be found in (Desmorat et al., 2007) and here we only summarize the results.

When the micro-defects closure parameter $h = 1$, the number of cycles to damage initiation ($D\,(N < N_D) = 0$) is given by

$$N_D = \frac{1}{4}\varepsilon_{pD}\frac{\boldsymbol{\mathcal{G}}^2}{C_y}\frac{\sigma_u - \sigma_f^\infty}{\left(\Delta\sigma/2 - \sigma_f^\infty\right)^2} \tag{A.1}$$

where $\boldsymbol{\mathcal{G}} = 3G(1-b)$ with G the shear modulus and b the Eshelby parameter. $\Delta\sigma = \sigma_{\max} - \sigma_{\min}$ is the stress amplitude at mesoscale.

The number of cycles to rupture ($D = D_c$) is given by

$$N_R = N_D + \frac{(2ES)^s \boldsymbol{\mathcal{G}} D_c}{\left(\sigma_f^\infty\right)^{2s}\left(\Delta\sigma - 2\sigma_f^\infty\right)\left(R_{v\,\min}^s + R_{v\,\max}^s\right)} \tag{A.2}$$

where the stress triaxiality functions are defined by

$$R_{v\,\min} = \frac{2}{3}(1+\nu) + \frac{1}{3}(1-2\nu)\left(\frac{\sigma_{\min}}{\sigma_f^\infty}\right)^2 \tag{A.3}$$

$$R_{v\,\max} = \frac{2}{3}(1+\nu) + \frac{1}{3}(1-2\nu)\left(\frac{\sigma_{\max}}{\sigma_f^\infty}\right)^2 \tag{A.4}$$

For a stress ratio of $R = \sigma\min/\sigma\max = 1$ and zero damage threshold $\varepsilon_{pD} = 0$, Eq. (A.2) can be further simplified as

$$N_R \approx \frac{(2ES)^s \boldsymbol{\mathcal{G}} D_c}{4\left(\sigma_f^\infty\right)^{2s}\left(\sigma_{\max} - 2\sigma_f^\infty\right) R_{v\,\max}^s} \tag{A.5}$$

Similarly, for shear loading it can be shown that the damage initiation life and fatigue life are

$$N_D^{\text{shear}} = \frac{1}{4}\varepsilon_{pD}\frac{\boldsymbol{\mathcal{G}}^2}{C_y}\frac{\sigma_u - \sigma_f^\infty}{\left(\sqrt{3}\Delta\tau/2 - \sigma_f^\infty\right)^2} \tag{A.6}$$

and

$$N_R^{\text{shear}} = N_D^{\text{shear}} + \left[\frac{3ES}{\left(\sigma_f^\infty\right)^2(1+\nu)}\right]^s \frac{\boldsymbol{\mathcal{G}}D_c}{2\left(\sqrt{3}\Delta\tau - 2\sigma_f^\infty\right)} \tag{A.7}$$

APPENDIX B

TWO-SCALE DAMAGE MODEL: PARAMETER CALIBRATION

Material properties involved in the two-scale damage model are summarized in Table B.1. It can be seen that most material parameters can be obtained directly from simple mechanical tests. However, the damage strength S and damage exponent s need to be calibrated by fitting S-N curves with the other material parameters in a two-steps scheme as follows.

In the first step, we let the micro-defects closure parameter $h = 1.0$ and fit the damage strength S and damage exponent s with the experimental S-N curve under logarithmic scales using the closed-form solution given in Appendix A. This step is referred to as the initial estimation step and can be done in MATLAB using its nonlinear least-squares curve fitting function `lsqcurvefit`[1].

Once initial estimation step is completed, we let $h = 0.2$ and keep the damage exponent s constant, then fit the experimental S-N curve by adjusting the damage strength S in the numerical solution scheme presented in Section 4.3, which is programmed using C++ as a standalone code. The second step is referred to as the numerical fitting step.

With the above two steps, a complete set of material parameters is determined for the two-scale fatigue damage model.

An example for the two-step S-N curve fitting Here we illustrate the above two-step S-N curve fitting process by an example. The uniaxial fatigue test data in this example is extracted from Fig. 2 in the paper by Desmorat et al. (Desmorat et al., 2007). The fatigue experiment is conducted under constant-amplitude, fully-reversed ($R = -1$) proportional loading condition. Other material parameters are given by $E = 197$ GPa, $\nu = 0.3$, $C_y = 1740$ MPa, $\varepsilon_{pD} = 0$, and $\sigma_f^\infty = 88$ MPa. Note that the energetic damage threshold w_D is zero since $\varepsilon_{pD} = 0$. Therefore, ultimate tensile strength σ_u is not required in this example.

[1]See `https://www.mathworks.com/help/optim/ug/lsqcurvefit.html`

Table B.1. Material parameters in the two-scale damage model

Parameter	Physical meaning	Source
E	Young's modulus	Uniaxial tensile test
ν	Poisson's ratio	Uniaxial tensile test
C_y	Plastic modulus	Uniaxial tensile test
σ_u	Ultimate tensile strength	Uniaxial tensile test
ε_{pD}	Plastic strain at failure	Uniaxial tensile test
h	Micro-defects closure parameter	≈ 0.2 for metals [a]
D_c	Crack initiation condition	≈ 0.3 for metals [b]
σ_f^∞	Asymptotic (endurance) fatigue limit	Uniaxial fatigue test
S	Damage strength	Uniaxial fatigue test
s	Damage exponent	Uniaxial fatigue test

Note: [a,b] See (Lemaitre, 1996; Lemaitre and Desmorat, 2005)

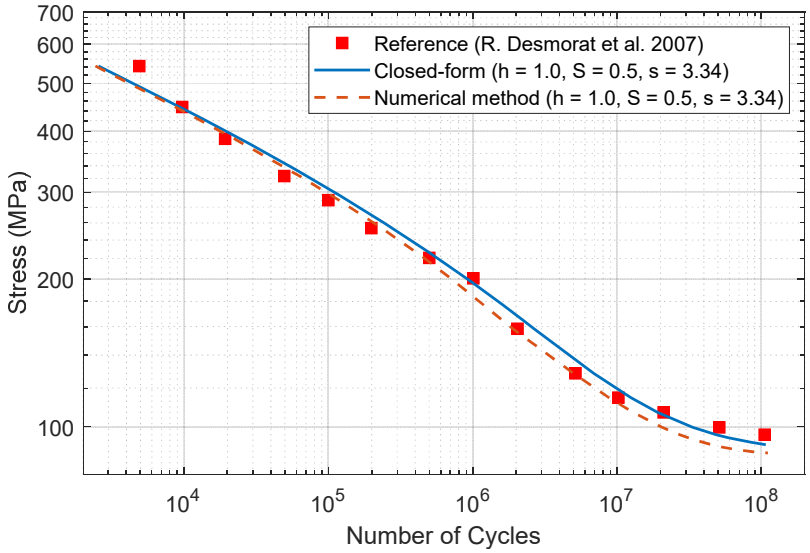


Figure B.1. Initial estimation of the damage strength S and damage exponent s

The result of initial estimation step is shown in Figure B.1 and compared with the test S-N data. The parameters are obtained as $S = 0.5$ and $s = 3.34$. For the purpose of validation, we also plotted the S-N curve obtained by substituting these parameters with $h = 1.0$ into the numerical standalone damage code and a good agreement among these curves is observed from Figure B.1.

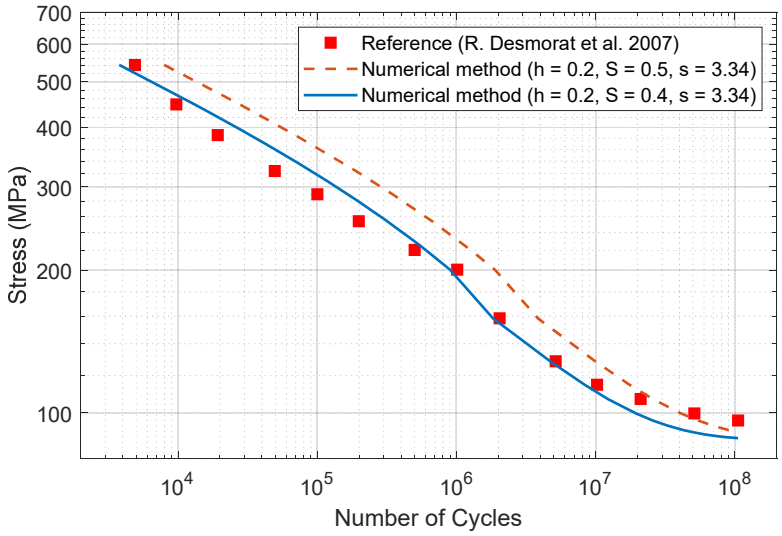


Figure B.2. Numerical fitting of the damage strength S

Next, by letting $h = 0.2$ and directly substituting $S = 0.5$ and $s = 3.34$ together with other material parameters into the numerical standalone damage code, we obtained a S-N curve that is slightly deviated from the experimental data as shown in Figure B.2. A good curve fitting is obtained by adjusting the damage strength S from 0.5 to 0.4.

Finally, we further integrated the standalone C++ code with the MATLAB curve fitting function `lsqcurvefit` to replace the trial and error approach employed in the above numerical fitting step. In this method, the MATLAB function `lsqcurvefit` directly calls the C++ code to compute the fatigue life. Parameters adjusting and the convergence are both controlled by the curve fitting algorithms in MATLAB. The fitting result is plotted in Figure B.3. As it can be seen, a good fit is obtained and both the damage strength S and the damage exponent s are fitted with the closure parameter $h = 0.2$. This method provides a more accurate option to fit these parameters in the two-scale damage model. However, its computational cost might be higher than the previous two-step method. Hence, it is recommended to use the parameters obtained from the above initial estimation step as the initial guess to achieve a faster convergence.

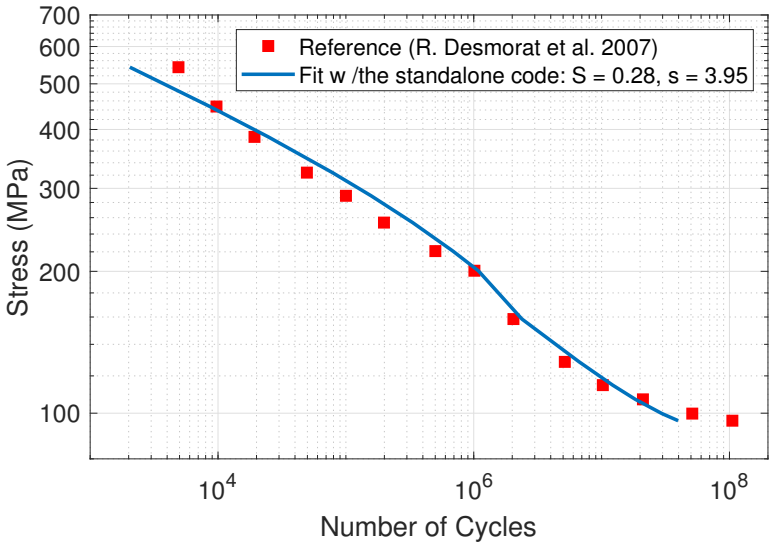


Figure B.3. Nonlinear least-squares fitting with the standalone code

APPENDIX C

CLUSTER-BASED LIPPMANN-SCHWINGER EQUATION: NEWTON'S ITERATION METHOD

The system of the cluster-based Lippmann-Schwinger equations are given by

$$\Delta\boldsymbol{\varepsilon}^I + \sum_{J=1}^{k} \boldsymbol{D}^{IJ} : \left[\Delta\boldsymbol{\sigma}^J - \boldsymbol{C}^0 : \Delta\boldsymbol{\varepsilon}^J\right] - \Delta\boldsymbol{\varepsilon}^0 = \boldsymbol{0} \qquad I = 1, 2, 3, .., k \tag{C.1}$$

in which k is the number of clusters and $\Delta\boldsymbol{\varepsilon}^0$ is the given macroscopic strain increment.

Eq. (C.1) is generally nonlinear and needs to be solved using the Newton's iteration method. The Newton's solution algorithm is summarized as follows:

1. For each cluster, given the local incremental strain tensor $\Delta\boldsymbol{\varepsilon}^I$, evaluate the local material constitutive model to find the local incremental stress tensor $\Delta\boldsymbol{\sigma}^I$ and tangent stiffness tensor $\boldsymbol{C}^I_{\text{alg}}$;
2. Calculate the residual $\boldsymbol{r} = \{\boldsymbol{r}^1, \boldsymbol{r}^2, ..., \boldsymbol{r}^k\}$ by

$$\boldsymbol{r}^I = \Delta\boldsymbol{\varepsilon}^I + \sum_{J=1}^{k} \boldsymbol{D}^{IJ} : \left[\Delta\boldsymbol{\sigma}^J - \boldsymbol{C}^0 : \Delta\boldsymbol{\varepsilon}^J\right] - \Delta\boldsymbol{\varepsilon}^0 \tag{C.2}$$

3. Calculate and assembly the system Jacobian matrix $\boldsymbol{M}$ by

$$\boldsymbol{M}^{IJ} = \frac{\partial \boldsymbol{r}^I}{\partial \Delta\boldsymbol{\varepsilon}^J} = \delta_{IJ}\boldsymbol{I}_4 + \boldsymbol{D}^{IJ} : (\boldsymbol{C}^J_{\text{alg}} - \boldsymbol{C}^0) \tag{C.3}$$

4. Solve the incremental strain tensor corrections $\delta\Delta\boldsymbol{\varepsilon} = \{\delta\Delta\boldsymbol{\varepsilon}^1, \delta\Delta\boldsymbol{\varepsilon}^2, ..., \delta\Delta\boldsymbol{\varepsilon}^k\}$ by

$$\delta\Delta\boldsymbol{\varepsilon} = -\boldsymbol{M}^{-1}\boldsymbol{r} \qquad \text{and} \qquad \Delta\boldsymbol{\varepsilon} \leftarrow \Delta\boldsymbol{\varepsilon} + \delta\Delta\boldsymbol{\varepsilon} \tag{C.4}$$

5. Check the convergence criteria, e.g. $|\delta\Delta\boldsymbol{\varepsilon}| < \epsilon$, in which ϵ is a threshold tolerance. If not converged then go back to Step 1, otherwise stop the iteration.

Note that the above algorithm is for macro-strain constrained Lippmann-Schwinger equation. For far-field stress loading cases where the macroscopic incremental stress tensor $\Delta\boldsymbol{\sigma}^0$

is given, the macroscopic strain increment $\Delta\boldsymbol{\varepsilon}^0$ is also unknown and needs to be solved for. In that case, an extra term is added to the residual vector

$$\boldsymbol{r}^{k+1} = \sum_{I=1}^{k} c^I \Delta\boldsymbol{\sigma}^I - \Delta\boldsymbol{\sigma}^0 \tag{C.5}$$

and correspondingly the Jacobian matrix is expanded by

$$\boldsymbol{M}^{I(k+1)} = -\boldsymbol{I}_4, \quad \boldsymbol{M}^{(k+1)J} = c^J \boldsymbol{C}^J_{\text{alg}}, \quad \boldsymbol{M}^{(k+1)(k+1)} = \boldsymbol{0} \tag{C.6}$$

The effective macroscopic tangent stiffness tensor is computed by

$$\bar{\boldsymbol{C}} = \frac{\partial\Delta\boldsymbol{\sigma}^0}{\partial\Delta\boldsymbol{\varepsilon}^0} = \sum_{I=1}^{k} c^I \frac{\partial\Delta\boldsymbol{\sigma}^I}{\partial\Delta\boldsymbol{\varepsilon}^0} = \sum_{I=1}^{k} c^I \frac{\partial\Delta\boldsymbol{\sigma}^I}{\partial\Delta\boldsymbol{\varepsilon}^I} : \frac{\partial\Delta\boldsymbol{\varepsilon}^I}{\partial\Delta\boldsymbol{\varepsilon}^0} = \sum_{I=1}^{k} c^I \boldsymbol{C}^I_{\text{alg}} : \boldsymbol{A}^I \tag{C.7}$$

where the local tangent stiffness tensors are readily available from the converged Newton iteration step.

To compute the strain concentration tensor $\boldsymbol{A}^I = \frac{\partial\Delta\boldsymbol{\varepsilon}^I}{\partial\Delta\boldsymbol{\varepsilon}^0}$, we first substitute its definition into Eq. (C.1), which yields

$$\boldsymbol{A}^I + \sum_{J=1}^{k} \boldsymbol{D}^{IJ} : (\boldsymbol{C}^J_{\text{alg}} - \boldsymbol{C}^0) : \boldsymbol{A}^J - \boldsymbol{I}_4 = \boldsymbol{0} \qquad I = 1, 2, 3, .., k \tag{C.8}$$

Then, by considering the system Jacobian in Eq. (C.3) we have

$$\{\boldsymbol{M}\}\{\boldsymbol{A}\} = \{\boldsymbol{I}\} \tag{C.9}$$

Let $\boldsymbol{B} = \boldsymbol{M}^{-1}$ in which $\boldsymbol{M}$ is the system Jacobian at the last Newton iteration, we can find the strain interaction tensors by $\boldsymbol{A}^I = \sum_{J=1}^{k} \boldsymbol{B}^{IJ}$. Subsequently, the effective macroscopic tangent stiffness tensor can computed by Eq. (C.7).

APPENDIX D

PERIDYNAMIC STATES

Peridynamic (PD) *state* is a mathematical object similar to tensor in classical continuum mechanics (CCM). It is introduced to derive the state-based PD. A PD state is a function or mapping defined on the bonds associated to a material point in a body. To better understand its mathematical concepts, we briefly summary the following definitions and conventions on PD states introduced by Silling et al. (Silling et al., 2007).

First, a state of order m is a mapping $\underline{\boldsymbol{A}}\langle\boldsymbol{\xi}\rangle : \mathcal{H} \to \mathcal{L}_m$ such that the image of $\boldsymbol{\xi} \in \mathcal{H}$ under the state is a tensor of order m in $\mathcal{L}_m$, which is the set of all m-order tensors. For example, a *scalar state* is a state of order 0 and a state of order 1 is called a *vector state*. States are generally written in uppercase, bold font with an underscore. A special case is for scalar states, which are denoted by lowercase, non-bold font with an underscore.

Second, the sum and subtraction of states of the same order are defined by

$$(\underline{\boldsymbol{A}} + \underline{\boldsymbol{B}})\langle\boldsymbol{\xi}\rangle = \underline{\boldsymbol{A}}\langle\boldsymbol{\xi}\rangle + \underline{\boldsymbol{B}}\langle\boldsymbol{\xi}\rangle \qquad \forall\boldsymbol{\xi} \in \mathcal{H} \tag{D.1}$$

$$(\underline{\boldsymbol{A}} - \underline{\boldsymbol{B}})\langle\boldsymbol{\xi}\rangle = \underline{\boldsymbol{A}}\langle\boldsymbol{\xi}\rangle - \underline{\boldsymbol{B}}\langle\boldsymbol{\xi}\rangle \qquad \forall\boldsymbol{\xi} \in \mathcal{H} \tag{D.2}$$

Point product of a state $\underline{\boldsymbol{A}}$ of order $m+p$ and a state $\underline{\boldsymbol{B}}$ of order p is a state of order m defined by

$$(\underline{\boldsymbol{AB}})_{i_1 i_2 \ldots i_m}\langle\boldsymbol{\xi}\rangle = \underline{A}_{i_1 i_2 \ldots i_m j_1 j_2 \ldots j_p}\langle\boldsymbol{\xi}\rangle\, \underline{B}_{j_1 j_2 \ldots j_p}\langle\boldsymbol{\xi}\rangle \qquad \forall\boldsymbol{\xi} \in \mathcal{H} \tag{D.3}$$

$$(\underline{\boldsymbol{BA}})_{i_1 i_2 \ldots i_m}\langle\boldsymbol{\xi}\rangle = \underline{B}_{j_1 j_2 \ldots j_p}\langle\boldsymbol{\xi}\rangle\, \underline{A}_{j_1 j_2 \ldots j_p i_1 i_2 \ldots i_m}\langle\boldsymbol{\xi}\rangle \qquad \forall\boldsymbol{\xi} \in \mathcal{H} \tag{D.4}$$

where summation occurs over repeated indices. The point product is commutative if any of the states is a scalar state or the two states are the same order. For example, the point product between scalar states $\underline{a}$ and $\underline{b}$ leads to

$$(\underline{ab})\langle\boldsymbol{\xi}\rangle = (\underline{ba})\langle\boldsymbol{\xi}\rangle = \underline{a}\langle\boldsymbol{\xi}\rangle\, \underline{b}\langle\boldsymbol{\xi}\rangle \qquad \forall\boldsymbol{\xi} \in \mathcal{H} \tag{D.5}$$

Similarly, for scalar state $\underline{a}$ and a vector state $\underline{\boldsymbol{B}}$, the product leads to

$$(\underline{a\boldsymbol{B}})\langle\boldsymbol{\xi}\rangle = (\underline{\boldsymbol{B}a})\langle\boldsymbol{\xi}\rangle = \underline{a}\langle\boldsymbol{\xi}\rangle\,\underline{\boldsymbol{B}}\langle\boldsymbol{\xi}\rangle \qquad \forall\boldsymbol{\xi}\in\mathcal{H} \tag{D.6}$$

Dot product of states $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$ is defined by

$$\underline{\boldsymbol{A}}\cdot\underline{\boldsymbol{B}} = \int_{\mathcal{H}} (\underline{\boldsymbol{A}\boldsymbol{B}})\langle\boldsymbol{\xi}\rangle\, dV \tag{D.7}$$

By definition, the dot product is also commutative if any of the states is a scalar state or the two states are the same order.

Magnitude state of $\underline{\boldsymbol{A}}$ is a scalar state defined by

$$\underline{a}\langle\boldsymbol{\xi}\rangle = \sqrt{(\underline{\boldsymbol{A}\boldsymbol{A}})\langle\boldsymbol{\xi}\rangle} \qquad \forall\boldsymbol{\xi}\in\mathcal{H} \tag{D.8}$$

Norm of $\underline{\boldsymbol{A}}$ is a scalar defined by

$$\|\underline{\boldsymbol{A}}\| = \sqrt{\underline{\boldsymbol{A}}\cdot\underline{\boldsymbol{A}}} \tag{D.9}$$

Finally, if both $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$ are vector states, tensor product of them leads to a second-order tensor defined by

$$\underline{\boldsymbol{A}}\otimes\underline{\boldsymbol{B}} = \int_{\mathcal{H}} \underline{\omega}\langle\boldsymbol{\xi}\rangle\,\underline{\boldsymbol{A}}\langle\boldsymbol{\xi}\rangle\otimes\underline{\boldsymbol{B}}\langle\boldsymbol{\xi}\rangle\, dV \tag{D.10}$$

where $\underline{\omega}$ is a scalar state termed as the *influence function*. If $\underline{\omega}$ only depends on the magnitude of bond vector, then it is said to be spherical and radially symmetric. The role of influence function was studied by Seleson and Parks (Seleson and Parks, 2011) and numerical experiments on the choice of influence functions were conducted by Queiruga and Moridis (Queiruga and Moridis, 2017). In this work, we chose the influence function to be $\underline{1}$ for simplicity.

APPENDIX E

A TIME HISTORY KERNEL APPROACH

Recall that our goal is to eliminate the fine-scale degrees of freedom (DOFs) in the coarse-scale subdomain and that the Peridynamics (PD) equation of motion essentially describes the total motion of the system such that

$$\boldsymbol{M}_p\ddot{\boldsymbol{q}} = \boldsymbol{f}\left(\boldsymbol{q}\right) \Leftrightarrow \boldsymbol{M}_p\ddot{\boldsymbol{u}} = \boldsymbol{f}\left(\boldsymbol{u}\right) \tag{E.1}$$

Now we introduce a partition that

$$\boldsymbol{u}' = \begin{Bmatrix} \boldsymbol{u}'_1 \\ \boldsymbol{u}'_2 \end{Bmatrix} \tag{E.2}$$

Note that $\boldsymbol{u}'_1$ needs to be solved explicitly while $\boldsymbol{u}'_2$ can be approximated through linearization.

Linearizing the force in $\boldsymbol{u}'_2$ yields

$$\boldsymbol{f}\left(\boldsymbol{u}\right) \doteq \boldsymbol{f}^*\left(\bar{\boldsymbol{u}} + \boldsymbol{u}'_1\right) - \boldsymbol{K}_2\boldsymbol{u}'_2 \tag{E.3}$$

in which $\boldsymbol{f}^*$ is the nonlinear internal force evaluated by setting $\boldsymbol{u}'_2 = \boldsymbol{0}$, and stiffness matrix $\boldsymbol{K}_2$ is given by

$$\boldsymbol{K}_{2,\alpha\beta} = -\left.\frac{\partial \boldsymbol{f}_\alpha}{\partial \boldsymbol{u}'_{2,\beta}}\right|_{\boldsymbol{u}'_2=0} \tag{E.4}$$

Note that comma in subscripts of Eq. (E.4) does not indicate partial derivative. The subscript α ranges over PD nodes in entire domain (assuming that PD nodes also exists in the coarse scale and we are trying to eliminate their DOFs from the governing equations), and subscript β ranges over the PD nodes in the coarse-scale subdomain.

Using a similar partition to Eq. (E.2), we can rewrite Eq. (E.1) as

$$\boldsymbol{M}_{p1}\ddot{\boldsymbol{q}}_1 = \boldsymbol{f}_1^*\left(\bar{\boldsymbol{u}} + \boldsymbol{u}'_1\right) - \boldsymbol{K}_{12}\boldsymbol{u}'_2 \tag{E.5}$$

and

$$\boldsymbol{M}_{p2}\ddot{\boldsymbol{q}}_2 = \boldsymbol{M}_{p2}\left(\ddot{\bar{\boldsymbol{u}}}_2 + \ddot{\boldsymbol{u}}'_2\right) = \boldsymbol{f}_2^*\left(\bar{\boldsymbol{u}} + \boldsymbol{u}'_1\right) - \boldsymbol{K}_{22}\boldsymbol{u}'_2 \tag{E.6}$$

Rearranging Eq. (E.6) leads to

$$\ddot{\boldsymbol{u}}'_2 + \boldsymbol{M}_{p2}^{-1}\boldsymbol{K}_{22}\boldsymbol{u}'_2 = \boldsymbol{M}_{p2}^{-1}\boldsymbol{f}_2^*\left(\bar{\boldsymbol{u}} + \boldsymbol{u}'_1\right) - \ddot{\bar{\boldsymbol{u}}}_2 \tag{E.7}$$

Thus, the fine-scale displacement field in the coarse-scale domain, i.e. $\boldsymbol{u}'_2$, can be eliminated by solving for them explicitly from Eq. (E.7) and substituting back to Eq. (E.5). The Eq. (E.7) can be solved using Laplace transform and finally leads to

$$\boldsymbol{M}_{p1}\ddot{\boldsymbol{q}}_1 = \boldsymbol{f}_1^*\left(\bar{\boldsymbol{u}} + \boldsymbol{u}'_1\right) - \int_0^t \boldsymbol{\theta}(t-\tau)\tilde{\boldsymbol{a}}_2 d\tau + \boldsymbol{R}(t) \tag{E.8}$$

in which

$$\boldsymbol{\theta}(t) = \mathscr{L}^{-1}\left\{\boldsymbol{K}_{12}\left(s^2\boldsymbol{I} + \boldsymbol{M}_{p2}^{-1}\boldsymbol{K}_{22}\right)^{-1}\right\} \tag{E.9}$$

$$\tilde{\boldsymbol{a}}_2 = \boldsymbol{M}_{p2}^{-1}\boldsymbol{f}_2^*\left(\bar{\boldsymbol{u}}(t) + \boldsymbol{u}'_1(t)\right) - \ddot{\bar{\boldsymbol{u}}}_2(t) \tag{E.10}$$

$$\boldsymbol{R}(t) = \dot{\boldsymbol{\theta}}(t)\boldsymbol{u}'_2(0) + \boldsymbol{\theta}(t)\dot{\boldsymbol{u}}'_2(0) \tag{E.11}$$

Note that $\mathscr{L}^{-1}$ in Eq. (E.9) indicates inverse Laplace transform.

The matrix $\boldsymbol{\theta}(t)$, which accounts for the effects of the removed DOFs, is a *time history kernel* (THK) and can be obtained analytically for simple cases or numerically for other cases (Wagner and Liu, 2003). In general, computational implementation of THK-based interfacial boundary condition is not an easy task.

REFERENCES

Aduloju, S. C. and T. J. Truster (2019). A variational multiscale discontinuous Galerkin formulation for both implicit and explicit dynamic modeling of interfacial fracture. *Computer Methods in Applied Mechanics and Engineering 343*, 602–630.

Alpert, D. N. (2009). Dynamic analysis of solid structures based on space-time finite element analysis. Master's thesis, University of Cincinnati, Cincinnati, OH.

Alpert, D. N. (2013). *Enriched Space-Time Finite Element Methods for Structural Dynamics Applications*. Ph. D. thesis, University of Cincinnati, Cincinnati, OH.

Amestoy, P. R., T. A. Davis, and I. S. Duff (1996). An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications 17*(4), 886–905.

Amestoy, P. R., I. S. Duff, J.-Y. L'Excellent, and J. Koster (2001). A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications 23*(1), 15–41.

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

Amirmaleki, M., J. Samei, D. E. Green, I. van Riemsdijk, and L. Stewart (2016). 3D micromechanical modeling of dual phase steels using the representative volume element method. *Mechanics of Materials 101*, 27–39.

Anahid, M., M. K. Samal, and S. Ghosh (2011). Dwell fatigue crack nucleation model based on crystal plasticity finite element simulations of polycrystalline titanium alloys. *Journal of the Mechanics and Physics of Solids 59*(10), 2157–2176.

Anderson, T. L. (2017). *Fracture mechanics: fundamentals and applications*. CRC press.

Andreev, K. and H. Räcke (2004). Balanced graph partitioning. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, New York, NY, USA, pp. 120–124. Association for Computing Machinery.

Aoyagi, T., F. Sawa, T. Shoji, H. Fukunaga, J.-i. Takimoto, and M. Doi (2002). A general-purpose coarse-grained molecular dynamics program. *Computer Physics Communications 145*(2), 267–279.

Argyris, J. H. and D. W. Scharpf (1969). Finite elements in time and space. *Nuclear Engineering and Design 10*(4), 456–464.

Ashurst, W. T. and W. G. Hoover (1976). Microscopic fracture studies in the two-dimensional triangular lattice. *Physical Review B 14*(4), 1465.

Askari, E., F. Bobaru, R. B. Lehoucq, M. L. Parks, S. A. Silling, and O. Weckner (2008). Peridynamics for multiscale materials modeling. *Journal of Physics: Conference Series 125*, 012078.

Babuška, I. (1976). Homogenization approach in engineering. In R. Glowinski and J. L. Lions (Eds.), *Computing Methods in Applied Sciences and Engineering*, Berlin, Heidelberg, pp. 137–153. Springer Berlin Heidelberg.

Babuška, I. and J. M. Melenk (1997). The partition of unity method. *International Journal for Numerical Methods in Engineering 40*(4), 727–758.

Barbu, L. G., S. Oller, X. Martinez, and A. Barbat (2015). High cycle fatigue simulation: A new stepwise load-advancing strategy. *Engineering Structures 97*, 118–129.

Barenblatt, G. I. (1962). The mathematical theory of equilibrium cracks in brittle fracture. *Advances in Applied Mechanics 7*(1), 55–129.

Bargmann, S., B. Klusemann, J. Markmann, J. E. Schnabel, K. Schneider, C. Soyarslan, and J. Wilmers (2018). Generation of 3D representative volume elements for heterogeneous materials: A review. *Progress in Materials Science 96*, 322–384.

Barnard, S. T. and H. D. Simon (1994). Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience 6*(2), 101–117.

Barrett, R., M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. Philadelphia, PA: SIAM.

Basquin, O. H. (1910). The exponential law of endurance tests. *American Society for Testing and Materials Proceedings 10*, 625–630.

Bažant, Z. P. (1994). Nonlocal damage theory based on micromechanics of crack interactions. *Journal of Engineering Mechanics 120*(3), 593–617.

Bažant, Z. P., T. B. Belytschko, and T.-P. Chang (1984). Continuum theory for strain-softening. *Journal of Engineering Mechanics 110*(12), 1666–1692.

Bažant, Z. P. and F. Lin (1988). Non-local yield limit degradation. *International Journal for Numerical Methods in Engineering 26*(8), 1805–1823.

Bažant, Z. P., W. Luo, V. T. Chau, and M. A. Bessa (2016). Wave dispersion and basic concepts of peridynamics compared to classical nonlocal damage models. *Journal of Applied Mechanics 83*(11).

Bažant, Z. P. and G. Pijaudier-Cabot (1988). Nonlocal continuum damage, localization instability and convergence. *Journal of Applied Mechanics 55*(2), 287–293.

Becker, R. and A. Needleman (1986). Effect of yield surface curvature on necking and failure in porous plastic solids. *Journal of Applied Mechanics 53*(3), 491–499.

Bednarek, T. and W. Sosnowski (2010). Practical fatigue analysis of hydraulic cylinders – Part II, damage mechanics approach. *International Journal of Fatigue 32*(10), 1591–1599.

Belytschko, T. and T. Black (1999). Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering 45*(5), 601–620.

Belytschko, T., Y. Guo, W. Kam Liu, and S. Ping Xiao (2000). A unified stability analysis of meshless particle methods. *International Journal for Numerical Methods in Engineering 48*(9), 1359–1400.

Belytschko, T. and J. I. Lin (1987). A three-dimensional impact-penetration algorithm with erosion. *International Journal of Impact Engineering 5*(1-4), 111–127.

Belytschko, T., W. K. Liu, B. Moran, and K. I. Elkhodary (2014). *Nonlinear finite elements for continua and structures* (2 ed.). Wiley.

Belytschko, T. and M. Tabbara (1996). Dynamic fracture using element-free Galerkin methods. *International Journal for Numerical Methods in Engineering 39*(6), 923–938.

Belytschko, T. and S. P. Xiao (2003). Coupling methods for continuum model with molecular model. *International Journal for Multiscale Computational Engineering 1*(1), 12.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*(9), 509–517.

Bessa, M. A., J. T. Foster, T. Belytschko, and W. K. Liu (2014). A meshfree unification: reproducing kernel peridynamics. *Computational Mechanics 53*(6), 1251–1264.

Bhamare, S. (2012). *High Cycle Fatigue Simulation using Extended Space-Time Finite Element Method Coupled with Continuum Damage Mechanics*. Ph. D. thesis, University of Cincinnati, Cincinnati, OH.

Bhamare, S., T. Eason, S. Spottswood, S. R. Mannava, V. K. Vasudevan, and D. Qian (2014). A multi-temporal scale approach to high cycle fatigue simulation. *Computational Mechanics 53*(2), 387–400.

Bhamare, S., G. Ramakrishnan, S. R. Mannava, K. Langer, V. K. Vasudevan, and D. Qian (2013). Simulation-based optimization of laser shock peening process for improved bending fatigue life of Ti–6Al–2Sn–4Zr–2Mo alloy. *Surface and Coatings Technology 232*, 464–474.

Bie, Y., X. Cui, and Z. Li (2018). A coupling approach of state-based peridynamics with node-based smoothed finite element method. *Computer Methods in Applied Mechanics and Engineering 331*, 675–700.

Bitzek, E., J. R. Kermode, and P. Gumbsch (2015). Atomistic aspects of fracture. *International Journal of Fracture 191*(1-2), 13–30.

Bland, A. S., W. Joubert, R. A. Kendall, D. B. Kothe, J. H. Rogers, and G. M. Shipman (2010). Jaguar: The world's most powerful computer system - an update. In *Cray User Group 2010*.

Bobaru, F., J. T. Foster, P. H. Geubelle, and S. A. Silling (2016). *Handbook of Peridynamic Modeling*. Advances in Applied Mathematics. New York: CRC Press.

Bobaru, F. and W. Hu (2012). The meaning, selection, and use of the peridynamic horizon and its relation to crack branching in brittle materials. *International Journal of Fracture 176*(2), 215–222.

Bobaru, F., Z. Xu, and Y. Wang (2016). Peridynamic modeling of impact and fragmentation. In *Handbook of Peridynamic Modeling*, pp. 417–442. Chapman and Hall/CRC.

Bobaru, F. and G. Zhang (2015). Why do cracks branch? A peridynamic investigation of dynamic brittle fracture. *International Journal of Fracture 196*(1), 59–98.

Bond, P. J., J. Holyoake, A. Ivetac, S. Khalid, and M. S. Sansom (2007). Coarse-grained molecular dynamics simulations of membrane proteins and peptides. *Journal of Structural Biology 157*(3), 593–605.

Bonora, N. and G. M. Newaz (1998). Low cycle fatigue life estimation for ductile metals using a nonlinear continuum damage mechanics model. *International Journal of Solids and Structures 35*(16), 1881–1894.

Breitenfeld, M. S., P. H. Geubelle, O. Weckner, and S. A. Silling (2014). Non-ordinary state-based peridynamic analysis of stationary crack problems. *Computer Methods in Applied Mechanics and Engineering 272*, 233–250.

Breitzman, T. and K. Dayal (2018). Bond-level deformation gradients and energy averaging in peridynamics. *Journal of the Mechanics and Physics of Solids 110*, 192–204.

Butt, S. N., J. J. Timothy, and G. Meschke (2017). Wave dispersion and propagation in state-based peridynamics. *Computational Mechanics 60*(5), 725–738.

Cai, W., M. de Koning, V. V. Bulatov, and S. Yip (2000). Minimizing boundary reflections in coupled-domain simulations. *Physical Review Letters 85*(15), 3213–3216.

Cantournet, S., R. Desmorat, and J. Besson (2009). Mullins effect and cyclic stress softening of filled elastomers by internal sliding and friction thermodynamics model. *International Journal of Solids and Structures 46*(11), 2255–2264.

Cedergren, J., S. Melin, and P. Lidström (2004). Numerical modelling of P/M steel bars subjected to fatigue loading using an extended gurson model. *European Journal of Mechanics - A/Solids 23*(6), 899–908.

Chan, K. S. (2010). Roles of microstructure in fatigue crack initiation. *International Journal of Fatigue 32*(9), 1428–1447.

Chan, W. M. and A. George (1980). A linear time implementation of the reverse Cuthill-McKee algorithm. *BIT Numerical Mathematics 20*(1), 8–14.

Cheng, S.-H. and C. Sun (2014). Size-dependent fracture toughness of nanoscale structures: Crack-tip stress approach in molecular dynamics. *Journal of Nanomechanics and Micromechanics 4*(4), A4014001.

Chessa, J. and T. Belytschko (2004). Arbitrary discontinuities in space-time finite elements by level sets and X-FEM. *International Journal for Numerical Methods in Engineering 61*(61), 2595–2614.

Chien, C. C. and T. Y. Wu (2000). An improved predictor/multi-corrector algorithm for a time-discontinuous Galerkin finite element method in structural dynamics. *Computational Mechanics 25*(5), 430–437.

Chien, C. C., C. S. Yang, and J. H. Tang (2003). Three-dimensional transient elastodynamic analysis by a space and time-discontinuous Galerkin finite element method. *Finite Elements in Analysis and Design 39*(7), 561–580.

Chirputkar, S. and D. Qian (2008). Coupled atomistic/continuum simulation based on extended space-time finite element method. *CMES - Computer Modeling in Engineering & Sciences 24*(2-3), 185–202.

Chowdhury, S. R., P. Roy, D. Roy, and J. N. Reddy (2019). A modified peridynamics correspondence principle: Removal of zero-energy deformation and other implications. *Computer Methods in Applied Mechanics and Engineering 346*, 530–549.

Chung, C.-S. and H.-K. Kim (2016). Fatigue strength of self-piercing riveted joints in lap-shear specimens of aluminium and steel sheets. *Fatigue & Fracture of Engineering Materials & Structures 39*(9), 1105–1114.

Clough, R. W. and J. Penzien (1993). *Dynamics of structures* (2nd ed.). New York: McGraw-Hill.

Cláudio, R. A., L. Reis, and M. Freitas (2014). Biaxial high-cycle fatigue life assessment of ductile aluminium cruciform specimens. *Theoretical and Applied Fracture Mechanics 73*, 82–90.

Coenen, E., V. Kouznetsova, and M. Geers (2012). Multi-scale continuous–discontinuous framework for computational-homogenization–localization. *Journal of the Mechanics and Physics of Solids 60*(8), 1486–1507.

Coffin, J. L. F. (1960). The stability of metals under cyclic plastic strain. *Journal of Basic Engineering 82*(3), 671–682.

Committee on Integrated Computational Materials Engineering (2008). *Integrated computational materials engineering: a transformational discipline for improved competitiveness and national security*. National Academies Press Washington, DC.

Cui, W. (2002). A state-of-the-art review on fatigue life prediction methods for metal structures. *Journal of Marine Science and Technology 7*(1), 43–56.

Curtin, W. A. and R. E. Miller (2003). Atomistic/continuum coupling in computational materials science. *Modelling and Simulation in Materials Science and Engineering 11*(3), R33.

Dagum, L. and R. Menon (1998). OpenMP: an industry standard api for shared-memory programming. *IEEE Computational Science and Engineering 5*(1), 46–55.

Davis, T. A. (2004). Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software 30*(2), 196–199.

Davis, T. A. (2006). *Direct Methods for Sparse Linear Systems*. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics.

Dayal, K. (2017). Leading-order nonlocal kinetic energy in peridynamics for consistent energetics and wave dispersion. *Journal of the Mechanics and Physics of Solids 105*, 235–253.

Delfour, M., W. Hager, and F. Trochu (1981). Discontinuous Galerkin methods for ordinary differential equations. *Mathematics of Computation 36*(154), 455–473.

Desmorat, R., A. Kane, M. Seyedi, and J. P. Sermage (2007). Two scale damage model and related numerical issues for thermo-mechanical high cycle fatigue. *European Journal of Mechanics A-Solids 26*(6), 909–935.

Dewapriya, M., R. Rajapakse, and A. Phani (2014). Atomistic and continuum modelling of temperature-dependent fracture of graphene. *International Journal of Fracture 187*(2), 199–212.

Diyaroglu, C., E. Madenci, R. J. Stewart, and S. S. Zoubi (2019). Combined peridynamic and finite element analyses for failure prediction in periodic and partially periodic perforated structures. *Composite Structures 229*, 111481.

Drugan, W. and J. Willis (1996). A micromechanics-based nonlocal constitutive equation and estimates of representative volume element size for elastic composites. *Journal of the Mechanics and Physics of Solids 44*(4), 497–524.

Dugdale, D. S. (1960). Yielding of steel sheets containing slits. *Journal of the Mechanics and Physics of Solids 8*(2), 100–104.

Dvorak, G. J. (1992). Transformation field analysis of inelastic composite materials. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences 437*(1900), 311–327.

E, W. and Z. Huang (2001). Matching conditions in atomistic-continuum modeling of materials. *Physical Review Letters 87*(13), 135501.

E, W. and Z. Huang (2002). A dynamic atomistic–continuum method for the simulation of crystalline materials. *Journal of Computational Physics 182*(1), 234–261.

Elices, M., G. V. Guinea, J. Gómez, and J. Planas (2002). The cohesive zone model: advantages, limitations and challenges. *Engineering Fracture Mechanics 69*(2), 137–163.

Eshelby, J. D. (1957). The determination of the elastic field of an ellipsoidal inclusion, and related problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences 241*(1226), 376–396.

Feyel, F. (1999). Multiscale FE2 elastoviscoplastic analysis of composite structures. *Computation Materials Science 16*(1-4), 344–354.

Feyel, F. (2003). A multilevel finite element method (FE2) to describe the response of highly non-linear structures using generalized continua. *Computer Methods in Applied Mechanics and Engineering 192*(28-30), 3233–3244.

Feyel, F. and J.-L. Chaboche (2000). FE2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre SiC/Ti composite materials. *Computer Methods in Applied Mechanics and Engineering 183*(3-4), 309–330.

Fish, J. (2011). Multiscale modeling and simulation of composite materials and structures. In R. de Borst and E. Ramm (Eds.), *Multiscale Methods in Computational Mechanics: Progress and Accomplishments*, Dordrecht, pp. 215–231. Springer Netherlands.

Fish, J., M. Bailakanavar, L. Powers, and T. Cook (2012). Multiscale fatigue life prediction model for heterogeneous materials. *International Journal for Numerical Methods in Engineering 91*(10), 1087–1104.

Fish, J., V. Filonova, and Z. Yuan (2012). Reduced order computational continua. *Computer Methods in Applied Mechanics and Engineering 221*, 104–116.

Fried, I. (1969). Finite-element analysis of time-dependent phenomena. *AIAA Journal 7*(6), 1170–1173.

Galvanetto, U., T. Mudric, A. Shojaei, and M. Zaccariotto (2016). An effective way to couple FEM meshes and Peridynamics grids for the solution of static equilibrium problems. *Mechanics Research Communications 76*, 41–47.

Ganzenmüller, G. C., S. Hiermaier, and M. May (2015). On the similarity of meshless discretizations of Peridynamics and Smooth-Particle Hydrodynamics. *Computers & Structures 150*, 71–78.

Gao, J. (2020). *Multiscale Modeling of Composite Laminate Structures: a Pathway Towards Predictive Science and Engineering*. Ph. D. thesis, Northwestern University, Evanston, IL.

Garikipati, K. and T. J. R. Hughes (2000). A variational multiscale approach to strain localization–formulation for multidimensional problems. *Computer Methods in Applied Mechanics and Engineering 188*(1-3), 39–60.

George, A., J. Liu, and E. Ng (1994). *Computer Solution of Sparse Linear Systems*. Oak Ridge National Laboratory.

Ghosh, S. and P. Chakraborty (2013). Microstructure and load sensitive fatigue crack nucleation in Ti-6242 using accelerated crystal plasticity fem simulations. *International Journal of Fatigue 48*, 231–246.

Giannakeas, I. N., T. K. Papathanasiou, and H. Bahai (2019). Wave reflection and cut-off frequencies in coupled FE-peridynamic grids. *International Journal for Numerical Methods in Engineering 120*(1), 29–55.

Giannakeas, I. N., T. K. Papathanasiou, A. S. Fallah, and H. Bahai (2020a). Coupling XFEM and Peridynamics for brittle fracture simulation—Part I: feasibility and effectiveness. *Computational Mechanics 66*(1), 103–122.

Giannakeas, I. N., T. K. Papathanasiou, A. S. Fallah, and H. Bahai (2020b). Coupling XFEM and Peridynamics for brittle fracture simulation: Part II—adaptive relocation strategy. *Computational Mechanics 66*(3), 683–705.

Gillner, K. and S. Münstermann (2017). Numerically predicted high cycle fatigue properties through representative volume elements of the microstructure. *International Journal of Fatigue 105*, 219–234.

Gitman, I., H. Askes, and L. Sluys (2007). Representative volume: Existence and size determination. *Engineering Fracture Mechanics 74*(16), 2518–2534.

Gropp, W., E. Lusk, and A. Skjellum (1999). *Using MPI: portable parallel programming with the message-passing interface* (Second ed.). Cambridge, Massachusetts: The MIT Press.

Gu, C., J. Lian, Y. Bao, Q. Xie, and S. Münstermann (2019). Microstructure-based fatigue modelling with residual stresses: Prediction of the fatigue life for various inclusion sizes. *International Journal of Fatigue 129*, 105158.

Gu, X., Q. Zhang, D. Huang, and Y. Yv (2016). Wave dispersion analysis and simulation method for concrete SHPB test in peridynamics. *Engineering Fracture Mechanics 160*, 124–137.

Gur, S., M. R. Sadat, G. N. Frantziskonis, S. Bringuier, L. Zhang, and K. Muralidharan (2019). The effect of grain-size on fracture of polycrystalline silicon carbide: A multiscale analysis using a molecular dynamics-peridynamics framework. *Computation Materials Science 159*, 341–348.

Gurson, A. L. (1977). Continuum theory of ductile rupture by void nucleation and growth: Part I—yield criteria and flow rules for porous ductile media. *Journal of Engineering Materials and Technology 99*(1), 2–15.

Ha, Y. D. and F. Bobaru (2010). Studies of dynamic crack propagation and crack branching with peridynamics. *International Journal of Fracture 162*(1-2), 229–244.

Han, F. and G. Lubineau (2012). Coupling of nonlocal and local continuum models by the Arlequin approach. *International Journal for Numerical Methods in Engineering 89*(6), 671–685.

Han, X., J. Gao, M. Fleming, C. Xu, W. Xie, S. Meng, and W. K. Liu (2020). Efficient multi-scale modeling for woven composites based on self-consistent clustering analysis. *Computer Methods in Applied Mechanics and Engineering 364*, 112929.

Hashin, Z. (1960). The elastic moduli of heterogeneous materials. Report, Harvard University, Cambridge, MA.

Hendrickson, B. and R. Leland (1995). An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing 16*(2), 452–469.

Hill, R. (1965). A self-consistent mechanics of composite materials. *Journal of the Mechanics and Physics of Solids 13*(4), 213–222.

Hu, W., Y. D. Ha, and F. Bobaru (2012). Peridynamic model for dynamic fracture in unidirectional fiber-reinforced composites. *Computer Methods in Applied Mechanics and Engineering 217-220*, 247–261.

Hughes, T. J. R. (1995). Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering 127*(1-4), 387–401.

Hughes, T. J. R., G. R. Feijóo, L. Mazzei, and J.-B. Quincy (1998). The variational multiscale method—a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering 166*(1-2), 3–24.

Hughes, T. J. R., L. P. Franca, and M. Mallet (1987). A new finite element formulation for computational fluid dynamics: VI. Convergence analysis of the generalized SUPG formulation for linear time-dependent multidimensional advective-diffusive systems. *Computer Methods in Applied Mechanics and Engineering 63*(1), 97–112.

Hughes, T. J. R. and G. M. Hulbert (1988). Space-time finite element methods for elastodynamics: formulations and error estimates. *Computer Methods in Applied Mechanics and Engineering 66*(3), 339–363.

Hughes, T. J. R. and J. R. Stewart (1996). A space-time formulation for multiscale phenomena. *Journal of Computational and Applied Mathematics 74*(1–2), 217–229.

Hulbert, G. M. (1992). Time finite element methods for structural dynamics. *International Journal for Numerical Methods in Engineering 33*(2), 307–331.

Hulbert, G. M. and T. J. R. Hughes (1990). Space-time finite element methods for second-order hyperbolic equations. *Computer Methods in Applied Mechanics and Engineering 84*(3), 327–348.

Irwin, G. R. (1957). Analysis of stresses and strains near the end of a crack traversing a plate. *Journal of Applied Mechanics 24*, 361–364.

Iyer, K., S. J. Hu, F. L. Brittman, P. C. Wang, D. B. Hayden, and S. P. Marin (2005). Fatigue of single- and double-rivet self-piercing riveted lap joints. *Fatigue & Fracture of Engineering Materials & Structures 28*(11), 997–1007.

Javili, A., R. Morasata, E. Oterkus, and S. Oterkus (2019). Peridynamics review. *Mathematics and Mechanics of Solids 24*(11), 3714–3739.

Jiang, H., X. Gao, and T. S. Srivatsan (2009). Predicting the influence of overload and loading mode on fatigue crack growth: A numerical approach using irreversible cohesive elements. *Finite Elements in Analysis and Design 45*(10), 675–685.

Johnson, C. (1988). Error estimates and adaptive time-step control for a class of one-step methods for stiff ordinary differential equations. *SIAM Journal on Numerical Analysis 25*(4), 908–926.

Johnson, G. R. and R. A. Stryk (1987). Eroding interface and improved tetrahedral element algorithms for high-velocity impact computations in three dimensions. *International Journal of Impact Engineering 5*(1-4), 411–421.

Kafka, O. L., C. Yu, M. Shakoor, Z. Liu, G. J. Wagner, and W. K. Liu (2018). Data-driven mechanistic modeling of influence of microstructure on high-cycle fatigue life of nickel titanium. *JOM 70*(7), 1154–1158.

Kalamkarov, A. L., I. V. Andrianov, and V. V. Danishevs'kyy (2009). Asymptotic homogenization of composite materials and structures. *Applied Mechanics Reviews 62*(3).

Kalthoff, J. and S. Winkler (1988). Failure mode transition at high rates of shear loading. *DGM Informationsgesellschaft mbH, Impact Loading and Dynamic Behavior of Materials 1*, 185–195.

Kang, K. and W. Cai (2007). Brittle and ductile fracture of semiconductor nanowires - molecular dynamics simulations. *Philosophical Magazine 87*(14-15), 2169–2189.

Kanit, T., S. Forest, I. Galliet, V. Mounoury, and D. Jeulin (2003). Determination of the size of the representative volume element for random composites: statistical and numerical approach. *International Journal of Solids and Structures 40*(13), 3647–3679.

Kanouté, P., D. Boso, J. Chaboche, and B. Schrefler (2009). Multiscale methods for composites: a review. *Archives of Computational Methods in Engineering 16*(1), 31–75.

Karim, M. R., M. Kattoura, S. R. Mannava, V. K. Vasudevan, A. S. Malik, and D. Qian (2018). A computational study on the microstructural evolution in near-surface copper grain boundary structures due to femtosecond laser processing. *Computational Mechanics 61*(1), 105–117.

Karimi, M., T. Roarty, and T. Kaplan (2006). Molecular dynamics simulations of crack propagation in ni with defects. *Modelling and Simulation in Materials Science and Engineering 14*(8), 1409.

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

Karypis, G. and V. Kumar (1998b). Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing 48*(1), 96–129.

Kikuchi, H., R. K. Kalia, A. Nakano, P. Vashishta, P. S. Branicio, and F. Shimojo (2005). Brittle dynamic fracture of crystalline cubic silicon carbide (3c-sic) via molecular dynamics simulation. *Journal of Applied Physics 98*(10), 103524.

Kilic, B., A. Agwai, and E. Madenci (2009). Peridynamic theory for progressive damage prediction in center-cracked composite laminates. *Composite Structures 90*(2), 141–151.

Kilic, B. and E. Madenci (2010). Coupling of peridynamic theory and the finite element method. *Journal of Mechanics of Materials and Structures 5*(5), 707–733.

Kilic, S. A. (2012). Effect of ordering for iterative solvers in structural mechanics problems. In M. W. Berry, K. A. Gallivan, E. Gallopoulos, A. Grama, B. Philippe, Y. Saad, and F. Saied (Eds.), *High-Performance Scientific Computing: Algorithms and Applications*, London, pp. 251–260. Springer-Verlag.

Kouznetsova, V., W. Brekelmans, and F. Baaijens (2001). An approach to micro-macro modeling of heterogeneous materials. *Computational mechanics 27*(1), 37–48.

Kouznetsova, V., M. G. Geers, and W. Brekelmans (2004). Multi-scale second-order computational homogenization of multi-phase materials: a nested finite element solution strategy. *Computer Methods in Applied Mechanics and Engineering 193*(48-51), 5525–5550.

Kouznetsova, V., M. G. Geers, and W. M. Brekelmans (2002). Multi-scale constitutive modelling of heterogeneous materials with a gradient-enhanced computational homogenization scheme. *International Journal for Numerical Methods in Engineering 54*(8), 1235–1260.

Kröner, E. (1961). On the plastic deformation of polycrystals. *Acta Metallurgica 9*(2), 155–161.

Kulkarni, S. and A. Tabarraei (2018). An analytical study of wave propagation in a peridynamic bar with nonuniform discretization. *Engineering Fracture Mechanics 190*, 347–366.

Kumar, V., A. Grama, A. Gupta, and G. Karypis (1994). *Introduction to parallel computing: design and analysis of algorithms*. Benjamin-Cummings Publishing Co., Inc.

Kunthong, P. and L. L. Thompson (2005). An efficient solver for the high-order accurate time-discontinuous Galerkin (TDG) method for second-order hyperbolic systems. *Finite Elements in Analysis and Design 41*(7-8), 729–762.

Lautrou, N., D. Thevenet, and J. Y. Cognard (2009). Fatigue crack initiation life estimation in a steel welded joint by the use of a two-scale damage model. *Fatigue & Fracture of Engineering Materials & Structures 32*(5), 403–417.

Lee, J., S. E. Oh, and J.-W. Hong (2017). Parallel programming of a peridynamics code coupled with finite element method. *International Journal of Fracture 203*(1-2), 99–114.

Lemaitre, J. (1985). A continuous damage mechanics model for ductile fracture. *Journal of Engineering Materials and Technology 107*(1), 83–89.

Lemaitre, J. (1996). *A Course on Damage Mechanics* (2 ed.). Berlin: Springer-Verlag.

Lemaitre, J. and R. Desmorat (2005). *Engineering Damage Mechanics: Ductile, Creep, Fatigue and Brittle Failures*. Berlin: Springer-Verlag Berlin Heidelberg.

Lemaitre, J. and I. Doghri (1994). Damage 90: a post processor for crack initiation. *Computer Methods in Applied Mechanics and Engineering 115*(3–4), 197–232.

Lemaitre, J., J. P. Sermage, and R. Desmorat (1999). A two scale damage concept applied to fatigue. *International Journal of Fracture 97*(1-4), 67–81.

Lesaint, P. and P. Raviart (1974). 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*, New York, pp. 89–123. Academic press.

Lestriez, P., F. Bogard, J. L. Shan, and Y. Q. Guo (2007). Damage evolution on mechanical parts under cyclic loading. *AIP Conference Proceedings 908*(1), 1389–1394.

Li, D., A. Chrysanthou, I. Patel, and G. Williams (2017). Self-piercing riveting—a review. *The International Journal of Advanced Manufacturing Technology 92*(5), 1777–1824.

Li, P., Z. M. Hao, and W. Q. Zhen (2018). A stabilized non-ordinary state-based peridynamic model. *Computer Methods in Applied Mechanics and Engineering 339*, 262–280.

Li, S., X. Liu, A. Agrawal, and A. C. To (2006). Perfectly matched multiscale simulations for discrete lattice systems: Extension to multiple dimensions. *Physical Review B 74*(4), 045418.

Li, X. D. and N. E. Wiberg (1996). Structural dynamic analysis by a time-discontinuous Galerkin finite element method. *International Journal for Numerical Methods in Engineering 39*(12), 2131–2152.

Li, X. D. and N. E. Wiberg (1998). Implementation and adaptivity of a space-time finite element method for structural dynamics. *Computer Methods in Applied Mechanics and Engineering 156*(1–4), 211–229.

Liu, W. and J.-W. Hong (2012). A coupling approach of discretized peridynamics with finite element method. *Computer Methods in Applied Mechanics and Engineering 245*, 163–175.

Liu, W. K., D. Qian, S. Gonella, S. Li, W. Chen, and S. Chirputkar (2010). Multiscale methods for mechanical science of complex materials: Bridging from quantum to stochastic multiresolution continuum. *International Journal for Numerical Methods in Engineering 83*(8-9), 1039–1080.

Liu, Z., M. Bessa, and W. K. Liu (2016). Self-consistent clustering analysis: an efficient multi-scale scheme for inelastic heterogeneous materials. *Computer Methods in Applied Mechanics and Engineering 306*, 319–341.

Liu, Z., M. Fleming, and W. K. Liu (2018). Microstructural material database for self-consistent clustering analysis of elastoplastic strain softening materials. *Computer Methods in Applied Mechanics and Engineering 330*, 547–577.

Liu, Z., O. L. Kafka, C. Yu, and W. K. Liu (2018). Data-driven self-consistent clustering analysis of heterogeneous materials with crystal plasticity. In E. Oñate, D. Peric, E. de Souza Neto, and M. Chiumenti (Eds.), *Advances in Computational Plasticity: A Book in Honour of D. Roger J. Owen*, Cham, pp. 221–242. Springer International Publishing.

Luo, J. and V. Sundararaghavan (2018). Stress-point method for stabilizing zero-energy modes in non-ordinary state-based peridynamics. *International Journal of Solids and Structures 150*, 197–207.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Berkeley, Calif., pp. 281–297. University of California Press.

Madenci, E. and E. Oterkus (2014). *Peridynamic Theory and Its Applications*. New York, NY: Springer.

Manson, S. S. (1965). Fatigue: A complex subject—some simple approximations. *Experimental Mechanics 5*(4), 193–226.

Matouš, K., M. G. D. Geers, V. G. Kouznetsova, and A. Gillman (2017). A review of predictive nonlinear theories for multiscale modeling of heterogeneous materials. *Journal of Computational Physics 330*, 192–220.

McDowell, D. L. (2007). Simulation-based strategies for microstructure-sensitive fatigue modeling. *Materials Science and Engineering: A 468-470*, 4–14.

Mear, M. E. and J. W. Hutchinson (1985). Influence of yield surface curvature on flow localization in dilatant plasticity. *Mechanics of Materials 4*(3), 395–407.

Melenk, J. M. and I. Babuška (1996). The partition of unity finite element method: Basic theory and applications. *Computer Methods in Applied Mechanics and Engineering 139*(1-4), 289–314.

Molent, L., M. McDonald, S. Barter, and R. Jones (2008). Evaluation of spectrum fatigue crack growth using variable amplitude data. *International Journal of Fatigue 30*(1), 119–137.

Moore, J. A., D. Frankel, R. Prasannavenkatesan, A. G. Domel, G. B. Olson, and W. K. Liu (2016). A crystal plasticity-based study of the relationship between microstructure and ultra-high-cycle fatigue life in nickel titanium alloys. *International Journal of Fatigue 91*, 183–194.

Mori, T. and K. Tanaka (1973). Average stress in matrix and average elastic energy of materials with misfitting inclusions. *Acta Metallurgica 21*(5), 571–574.

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

National Science Technology Council (2011). *Materials genome initiative for global competitiveness*. Executive Office of the President, National Science and Technology Council.

Needleman, A. (1987). A continuum model for void nucleation by inclusion debonding. *Journal of Applied Mechanics 54*(3), 525–531.

Needleman, A. (2014). Some issues in cohesive surface modeling. *Procedia IUTAM 10*, 221–246.

Newmark, N. M. (1959). A method of computation for structural dynamics. *Proceedings of the American Society of Civil Engineers 85*, 67–94.

Ni, T., M. Zaccariotto, Q.-Z. Zhu, and U. Galvanetto (2019). Static solution of crack propagation problems in peridynamics. *Computer Methods in Applied Mechanics and Engineering 346*, 126–151.

Nicely, C. (2018). *Concurrently coupled Peridynamics/Finite Element simulation based on nonlocal matching boundary conditions*. Ph. D. thesis, The Univeristy of Texas at Dallas, Richardson, TX.

Nicely, C., S. Tang, and D. Qian (2018). Nonlocal matching boundary conditions for non-ordinary peridynamics with correspondence material model. *Computer Methods in Applied Mechanics and Engineering 338*, 463–490.

Nordmann, J., K. Naumenko, and H. Altenbach (2020). Cohesive zone models—theory, numerics and usage in high-temperature applications to describe cracking and delamination. In K. Naumenko and M. Krüger (Eds.), *Advances in Mechanics of High-Temperature Materials*, Cham, pp. 131–168. Springer International Publishing.

Oden, J. T. (1969). A general theory of finite elements. II. applications. *International Journal for Numerical Methods in Engineering 1*(3), 247–259.

Oden, J. T., T. Belytschko, J. Fish, T. J. R. Hughes, C. Johnson, D. Keyes, A. Laub, L. Petzold, D. Srolovitz, and S. Yip (2006). Simulation-based engineering science: Revolutionizing engineering science through simulation. *NSF Blue Ribbon Panel on SBES*.

Oliver, J., M. Caicedo, A. E. Huespe, J. Hernández, and E. Roubin (2017). Reduced order modeling strategies for computational multiscale fracture. *Computer Methods in Applied Mechanics and Engineering 313*, 560–595.

Oller, S., O. Salomón, and E. Oñate (2005). A continuum mechanics model for mechanical fatigue analysis. *Computation Materials Science 32*(2), 175–195.

ORNL (2018a). Summit user guide: system overview. *https://www.olcf.ornl.gov/for-users/system-user-guides/summit/system-overview Accessed Dec. 21, 2018.*

ORNL (2018b). Ttian user guide: system overview. *https://www.olcf.ornl.gov/for-users/system-user-guides/titan/system-overview Accessed Dec. 21, 2018.*

Oskay, C. and J. Fish (2007). Eigendeformation-based reduced order homogenization for failure analysis of heterogeneous materials. *Computer Methods in Applied Mechanics and Engineering 196*(7), 1216–1243.

Ostoja-Starzewski, M. (2006). Material spatial randomness: From statistical to representative volume element. *Probabilistic Engineering Mechanics 21*(2), 112–132.

Oterkus, E. and E. Madenci (2012). Peridynamic analysis of fiber-reinforced composite materials. *Journal of Mechanics of Materials and Structures 7*(1), 45–84.

Oterkus, E., E. Madenci, O. Weckner, S. Silling, P. Bogert, and A. Tessler (2012). Combined finite element and peridynamic analyses for predicting failure in a stiffened composite curved panel with a central slot. *Composite Structures 94*(3), 839–850.

Papanicolau, G., A. Bensoussan, and J.-L. Lions (1978). *Asymptotic analysis for periodic structures*. Elsevier.

Paris, P. and F. Erdogan (1963). A critical analysis of crack propagation laws. *Journal of Basic Engineering 85*(4), 528–533.

Park, K. and G. H. Paulino (2013). Cohesive zone models: A critical review of traction-separation relationships across fracture surfaces. *Applied Mechanics Reviews 64*(6).

Patil, S. P. and Y. Heider (2019). A review on brittle fracture nanomechanics by all-atom simulations. *Nanomaterials (Basel, Switzerland) 9*(7), 1050.

Paz, M. and W. Leigh (2004). *Structural Dynamics: Theory and Computation* (5th ed.). Springer US.

Pijaudier-Cabot, G. and Z. P. Bažant (1987). Nonlocal damage theory. *Journal of Engineering Mechanics 113*(10), 1512–1533.

Pijaudier-Cabot, G., K. Haidar, and J. Dubé (2004). Non-local damage model with evolving internal length. *International Journal for Numerical and Analytical Methods in Geomechanics 28*(7-8), 633–652.

Pirondi, A., N. Bonora, D. Steglich, W. Brocks, and D. Hellmann (2006). Simulation of failure under cyclic plastic loading by damage models. *International Journal of Plasticity 22*(11), 2146–2170.

Poncelet, M., G. Barbier, B. Raka, S. Courtin, R. Desmorat, J. C. Le-Roux, and L. Vincent (2010). Biaxial high cycle fatigue of a type 304L stainless steel: Cyclic strains and crack initiation detection by digital image correlation. *European Journal of Mechanics A-Solids 29*(5), 810–825.

Pothen, A., H. D. Simon, and K.-P. Liou (1990). Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications 11*(3), 430–452.

Pothen, A., H. D. Simon, L. Wang, and S. T. Barnard (1992). Towards a fast implementation of spectral nested dissection. In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, Supercomputing '92, Washington, DC, USA, pp. 42–51. IEEE Computer Society Press.

Przybyla, C., R. Prasannavenkatesan, N. Salajegheh, and D. L. McDowell (2010). Microstructure-sensitive modeling of high cycle fatigue. *International Journal of Fatigue 32*(3), 512–525.

Przybyla, C. P. and D. L. McDowell (2010). Microstructure-sensitive extreme value probabilities for high cycle fatigue of Ni-base superalloy IN100. *International Journal of Plasticity 26*(3), 372–394.

Przybyla, C. P. and D. L. McDowell (2011). Simulated microstructure-sensitive extreme value probabilities for high cycle fatigue of duplex Ti–6Al–4V. *International Journal of Plasticity 27*(12), 1871–1895.

Qian, D. and S. Chirputkar (2014). Bridging scale simulation of lattice fracture using enriched space-time finite element method. *International Journal for Numerical Methods in Engineering 97*(11), 819–850.

Qian, D., G. J. Wagner, and W. K. Liu (2004). A multiscale projection method for the analysis of carbon nanotubes. *Computer Methods in Applied Mechanics and Engineering 193*(17-20), 1603–1632.

Qian, D., Z. Zhou, and Q. Zheng (2015). Coarse-grained modeling and simulation of graphene sheets based on a discrete hyperelastic approach. *International Journal for Numerical Methods in Engineering 102*(3-4), 450–467.

Queiruga, A. and G. Moridis (2017). Numerical experiments on the convergence properties of state-based peridynamic laws and influence functions in two-dimensional problems. *Computer Methods in Applied Mechanics and Engineering 322*, 97–122.

Rabczuk, T. and T. Belytschko (2004). Cracking particles: a simplified meshfree method for arbitrary evolving cracks. *International Journal for Numerical Methods in Engineering 61*(13), 2316–2343.

Rafii-Tabar, H. (1998). Nanoscopic modelling of the adhesion, indentation and fracture characteristics of metallic systems via molecular dynamics simulations. In H. Kitagawa, T. Aihara, and Y. Kawazoe (Eds.), *Mesoscopic Dynamics of Fracture: Computational Materials Design*, Berlin, Heidelberg, pp. 36–48. Springer Berlin Heidelberg.

Raje, N., T. Slack, and F. Sadeghi (2009). A discrete damage mechanics model for high cycle fatigue in polycrystalline materials subject to rolling contact. *International Journal of Fatigue 31*(2), 346–360.

Rao, S. S. (2011). *Mechanical vibrations* (5th ed.). Upper Saddle River, N.J.: Prentice Hall.

Reed, W. and T. Hill (1973). Triangular mesh methods for the neutron transport equation. Report LA-UR-73-479, Los Alamos Scientific Laboratory.

RIKEN (2018). System configuration of the K computer. *https://www.r-ccs.riken.jp/en/wp-content/uploads/system_handout.pdf Accessed Dec. 21, 2018.*

Roe, K. L. and T. Siegmund (2003). An irreversible cohesive zone model for interface fatigue crack growth simulation. *Engineering Fracture Mechanics 70*(2), 209–232.

Rudd, R. E. and J. Q. Broughton (1998). Coarse-grained molecular dynamics and the atomic limit of finite elements. *Physical review B 58*(10), R5893.

Rudd, R. E. and J. Q. Broughton (2005). Coarse-grained molecular dynamics: Nonlinear finite elements and finite temperature. *Physical Review B 72*(14), 144104.

Réthoré, J., A. Gravouil, and A. Combescure (2005). A combined space–time extended finite element method. *International Journal for Numerical Methods in Engineering 64*(2), 260–284.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems* (Second ed.). Society for Industrial and Applied Mathematics.

Saad, Y. and M. H. Schultz (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing 7*(3), 856–869.

Schijve, J. (2003). Fatigue of structures and materials in the 20th century and the state of the art. *International Journal of Fatigue 25*(8), 679–702.

Schlinkman, R. T. (2019). Thermo-mechanical fatigue using the extended space-time finite element method. Master's thesis, The Univeristy of Texas at Dallas, Richardson, TX.

Schütz, W. (1996). A history of fatigue. *Engineering Fracture Mechanics 54*(2), 263–300.

Seleson, P., S. Beneddine, and S. Prudhomme (2013). A force-based coupling scheme for peridynamics and classical elasticity. *Computation Materials Science 66*, 34–49.

Seleson, P. and M. Parks (2011). On the role of the influence function in the peridynamic theory. *International Journal for Multiscale Computational Engineering 9*(6), 689–706.

Shenoy, V. B., R. Miller, E. b. Tadmor, D. Rodney, R. Phillips, and M. Ortiz (1999). An adaptive finite element approach to atomic-scale mechanics—the quasicontinuum method. *Journal of the Mechanics and Physics of Solids 47*(3), 611–642.

Siegmund, T. (2004). A numerical study of transient fatigue crack growth by use of an irreversible cohesive zone model. *International Journal of Fatigue 26*(9), 929–939.

Silling, S. A. (2000). Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids 48*(1), 175–209.

Silling, S. A. (2017). Stability of peridynamic correspondence material models and their particle discretizations. *Computer Methods in Applied Mechanics and Engineering 322*, 42–57.

Silling, S. A. and E. Askari (2005). A meshfree method based on the peridynamic model of solid mechanics. *Computers & Structures 83*(17), 1526–1535.

Silling, S. A., M. Epton, O. Weckner, J. Xu, and E. Askari (2007). Peridynamic states and constitutive modeling. *Journal of Elasticity 88*(2), 151–184.

Silling, S. A. and R. B. Lehoucq (2010). Peridynamic theory of solid mechanics. In H. Aref and E. v. d. Giessen (Eds.), *Advances in Applied Mechanics*, Volume 44, pp. 73–168. Elsevier.

Silling, S. A., D. J. Littlewood, and P. Seleson (2015). Variable horizon in a peridynamic medium. *Journal of Mechanics of Materials and Structures 10*(5), 591–612.

Simulia (2014a). 5.1.1 small-sliding interaction between bodies. In *Abaqus Theory Guide* (6.14 ed.)., Providence, RI, USA.

Simulia (2014b). 6.2.7 low-cycle fatigue analysis using the direct cyclic approach. In *Abaqus Analysis User's Guide* (6.14 ed.)., Providence, RI, USA.

Song, J.-H., H. Wang, and T. Belytschko (2008). A comparative study on finite element methods for dynamic fracture. *Computational Mechanics 42*(2), 239–250.

Song, S. H., G. H. Paulino, and W. G. Buttlar (2006). A bilinear cohesive zone model tailored for fracture of asphalt concrete considering viscoelastic bulk material. *Engineering Fracture Mechanics 73*(18), 2829–2848.

Strouboulis, T., I. Babuška, and K. Copps (2000). The design and analysis of the generalized finite element method. *Computer Methods in Applied Mechanics and Engineering 181*(1–3), 43–69.

Sun, W. and J. Fish (2019). Superposition-based coupling of peridynamics and finite element method. *Computational Mechanics 64*(1), 231–248.

Sun, X., E. V. Stephens, and M. A. Khaleel (2007). Fatigue behaviors of self-piercing rivets joining similar and dissimilar sheet metals. *International Journal of Fatigue 29*(2), 370–386.

Swadener, J. G., M. I. Baskes, and M. Nastasi (2002). Molecular dynamics simulation of brittle fracture in silicon. *Phys Rev Lett 89*(8), 085503.

Sánchez-Palencia, E. (1980). Non-homogeneous media and vibration theory. *Lecture Notes in Physics 127*.

Tadmor, E. B., M. Ortiz, and R. Phillips (1996). Quasicontinuum analysis of defects in solids. *Philosophical Magazine A 73*(6), 1529–1563.

Takagaki, M. and T. Nakamura (2007). Fatigue crack modeling and simulation based on continuum damage mechanics. *Journal of Pressure Vessel Technology 129*(1), 96–102.

Takahashi, T. and T. Hamada (2009). GPU-accelerated boundary element method for helmholtz' equation in three dimensions. *International Journal for Numerical Methods in Engineering 80*(10), 1295–1321.

Tang, S. (2008). A finite difference approach with velocity interfacial conditions for multiscale computations of crystalline solids. *Journal of Computational Physics 227*(8), 4038–4062.

TechPowerUp (2018). GPU Specs Database: Manufacturer - NVIDIA, Generation - Tesla. *https://www.techpowerup.com/gpu-specs/?mfgr=NVIDIA&generation=Tesla Accessed Dec. 21, 2018*.

Tian, R. (2014). Simulation at extreme-scale: Co-design thinking and practices. *Archives of Computational Methods in Engineering 21*(1), 39–58.

Tian, R., S. Chan, S. Tang, A. M. Kopacz, J.-S. Wang, H.-J. Jou, L. Siad, L.-E. Lindgren, G. B. Olson, and W. K. Liu (2010). A multiresolution continuum simulation of the ductile fracture process. *Journal of the Mechanics and Physics of Solids 58*(10), 1681–1700.

Timmermann, K., W. Zinn, and B. Scholtes (2014). Corrosion and fatigue of AL-Alloys AA359.0 and AA6060 in different surface treatment states. In *International Conference on Residual Stresses 9*, Volume 768 of *Materials Science Forum*, pp. 572–579. Trans Tech Publications Ltd.

To, A. C. and S. Li (2005). Perfectly matched multiscale simulations. *Physical Review B 72*(3), 035414.

Tong, Q. and S. Li (2016). Multiscale coupling of molecular dynamics and peridynamics. *Journal of the Mechanics and Physics of Solids 95*, 169–187.

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

Tvergaard, V. and A. Needleman (1984). Analysis of the cup-cone fracture in a round tensile bar. *Acta Metallurgica 32*(1), 157–169.

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

Vorst, H. A. v. d. (1992). Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing 13*(2), 631–644.

Wada, S. (2017). *Multiscale Simulation of Failure Based on Space-Time Finite Element Method and Non-Ordinary State-Based Peridynamics*. Ph. D. thesis, The Univeristy of Texas at Dallas, Richardson, TX.

Wada, S., R. Zhang, S. R. Mannava, V. K. Vasudevan, and D. Qian (2018). Simulation-based prediction of cyclic failure in rubbery materials using nonlinear space-time finite element method coupled with continuum damage mechanics. *Finite Elements in Analysis and Design 138*, 21–30.

Wagener, R. and T. Melz (2018). Fatigue life curve – a continuous wöhler curve from LCF to VHCF. *Materials Testing 60*(10), 924–930.

Wagner, G. J. and W. K. Liu (2003). Coupling of atomistic and continuum simulations using a bridging scale decomposition. *Journal of Computational Physics 190*(1), 249–274.

Wallace, E. J. and M. S. Sansom (2007). Carbon nanotube/detergent interactions via coarse-grained molecular dynamics. *Nano letters 7*(7), 1923–1928.

Wang, L., Y. Chen, J. Xu, and J. Wang (2017). Transmitting boundary conditions for 1D peridynamics. *International Journal for Numerical Methods in Engineering 110*(4), 379–400.

Wang, X., S. S. Kulkarni, and A. Tabarraei (2019). Concurrent coupling of peridynamics and classical elasticity for elastodynamic problems. *Computer Methods in Applied Mechanics and Engineering 344*, 251–275.

Wang, X. and S. Tang (2010). Matching boundary conditions for diatomic chains. *Computational Mechanics 46*(6), 813–826.

Wang, X. and S. Tang (2013). Matching boundary conditions for lattice dynamics. *International Journal for Numerical Methods in Engineering 93*(12), 1255–1285.

Wheeler, O. E. (1972). Spectrum loading and crack growth. *Journal of Basic Engineering 94*(1), 181–186.

Wiberg, N.-E. and X. Li (1999). Adaptive finite element procedures for linear and nonlinear dynamics. *International Journal for Numerical Methods in Engineering 46*(10), 1781–1802.

Xiao, S. P. and T. Belytschko (2004). A bridging domain method for coupling continua with molecular dynamics. *Computer Methods in Applied Mechanics and Engineering 193*(17), 1645–1669.

Xu, M. and T. Belytschko (2008). Conservation properties of the bridging domain method for coupled molecular/continuum dynamics. *International Journal for Numerical Methods in Engineering 76*(3), 278–294.

Xu, X. P. and A. Needleman (1994). Numerical simulations of fast crack growth in brittle solids. *Journal of the Mechanics and Physics of Solids 42*(9), 1397–1434.

Yaghoobi, A. and M. G. Chorzepa (2018). Formulation of symmetry boundary modeling in non-ordinary state-based peridynamics and coupling with finite element analysis. *Mathematics and Mechanics of Solids 23*(8), 1156–1176.

Yang, B., S. Mall, and K. Ravi-Chandar (2001). A cohesive zone model for fatigue crack growth in quasibrittle materials. *International Journal of Solids and Structures 38*(22-23), 3927–3944.

Yang, Y., S. Chirputkar, D. N. Alpert, T. Eason, S. Spottswood, and D. Qian (2012). Enriched space-time finite element method: a new paradigm for multiscaling from elastodynamics to molecular dynamics. *International Journal for Numerical Methods in Engineering 92*(2), 115–140.

Yu, C., O. L. Kafka, and W. K. Liu (2019). Self-consistent clustering analysis for multiscale modeling at finite strains. *Computer Methods in Applied Mechanics and Engineering 349*, 339–359.

Yu, Y., F. F. Bargos, H. You, M. L. Parks, M. L. Bittencourt, and G. E. Karniadakis (2018). A partitioned coupling framework for peridynamics and classical theory: analysis and simulations. *Computer Methods in Applied Mechanics and Engineering 340*, 905–931.

Yuan, Z. and J. Fish (2009). Hierarchical model reduction at multiple scales. *International Journal for Numerical Methods in Engineering 79*(3), 314–339.

Zaccariotto, M., T. Mudric, D. Tomasi, A. Shojaei, and U. Galvanetto (2018). Coupling of fem meshes with peridynamic grids. *Computer Methods in Applied Mechanics and Engineering 330*, 471–497.

Zhang, P., L. Ma, F. Fan, Z. Zeng, C. Peng, P. E. Loya, Z. Liu, Y. Gong, J. Zhang, X. Zhang, P. M. Ajayan, T. Zhu, and J. Lou (2014). Fracture toughness of graphene. *Nature Communications 5*, 3782.

Zhang, R., S. Naboulsi, T. Eason, and D. Qian (2019). A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing. *Finite Elements in Analysis and Design 166*, 103320.

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

Zhang, R., L. Wen, J. Xiao, and D. Qian (2019). An efficient solution algorithm for space–time finite element method. *Computational Mechanics 63*(3), 455–470.

Zhang, S. and C. Oskay (2016). Reduced order variational multiscale enrichment method for elasto-viscoplastic problems. *Computer Methods in Applied Mechanics and Engineering 300*, 199–224.

BIOGRAPHICAL SKETCH

Rui Zhang earned his Bachelor of Science in Engineering from Northwestern Polytechnical University in Xi'an, China in 2011. He continued his graduate study there with a concentration on Computational Mechanics and received his Master of Science in Engineering in 2014. Shortly after his graduation, he came to The University of Texas at Dallas as a visiting student in the Department of Mechanical Engineering to pursue his research interest in Multiscale Mechanics. In 2017, he received the prestigious Eugene McDermott Graduate Fellowship offered by The University of Texas at Dallas, which allowed him to continue his research on mechanics and to earn his Doctor of Philosophy in Mechanical Engineering.

CURRICULUM VITAE

Rui Zhang

October, 2020

Contact Information:

Department of Mechanical Engineering
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: rui.zhang@utdallas.edu

Educational History:

Ph.D., Mechanical Engineering, The University of Texas at Dallas, 2020
M.S., Flight Vehicle Design, Northwestern Polytechnical University, 2014
B.S., Flight Vehicle Design and Engineering, Northwestern Polytechnical University, 2011

Employment History:

Research Assistant, The University of Texas at Dallas, May 2017 – present

Honors and Awards:

Best Student Poster Competition on Computational Mechanics Award, Applied Mechanics Division, American Society of Mechanical Engineers, 2018
Eugene McDermott Graduate Fellowship, The University of Texas at Dallas, 2017–2021
Outstanding Thesis Award, Northwestern Polytechnical University, 2014
National Scholarship for Graduate Students, Ministry of Education, China, 2012
First Prize Winner of Postgraduate Mathematical Contest in Modeling, China Academic Degrees and Graduate Education Development Center of Ministry of Education, 2011
National Scholarship, Ministry of Education, China, 2010
Meritorious Winner of Mathematical Contest in Modeling, Consortium for Mathematics and Its Applications, 2010
National Second Prize Winner of Contemporary Undergraduate Mathematical Contest in Modeling, China Society for Industrial and Applied Mathematics, 2010
First Prize Scholarship, Northwestern Polytechnical University, 2008–2012

Professional Appointments and Memberships:

Ad Hoc Reviewer, Journal of Mechanics of Time-Dependent Materials, 2020
Chair for two technical sessions, IMECE, 2018
Co-organizer for one technical session, IMECE, 2017
Member, American Society of Mechanical Engineers, 2017
Member, American Association for the Advancement of Science, 2016

Research Interests:

Finite Element Method, Boundary Element Method, Meshfree Particle Methods, and Peridynamics; Multiscale and multiphysics mechanics of advanced functional materials, 2D materials, nanomaterials, and composites; Multiscale approaches to fatigue and fracture failures in materials and structures; High performance computing techniques and data-driven approaches in computational solid mechanics

Peer-reviewed Journal Publications:

1. He W., **R. Zhang**, Y. Cheng, C. Zhang, X. Zhou, Z. Liu, X. Hu, Z. Liu, J. Sun, G. Wan, Y. Wang, D. Qian, and Z. Liu (2020). Intrinsic elastic conductors with internal buckled electron pathway for flexible electromagnetic interference shielding and tumor ablation. *Science China Materials 63*(7), 1318–1329.
2. **Zhang, R.**, S. Naboulsi, T. Eason, and D. Qian (2019). A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing. *Finite Elements in Analysis and Design 166*, 103320.
3. **Zhang, R.**, L. Wen, J. Xiao, and D. Qian (2019). An efficient solution algorithm for space-time finite element method. *Computational Mechanics 63*(3), 455–470.
4. Wang X, T. Xu, **R. Zhang**, D. Qian, P. Kokkada, S. Roy, and H. Lu (2019). Modelling the compressive buckling load as a function of the interphase nanocomposite region thickness in a carbon nanotube sheet wrapped carbon fiber composite. *Journal of Applied Mechanics 86*(10), 101007.
5. Li J., L. Mou, **R. Zhang**, J. Sun, R. Wang, B. An, H. Chen, K. Inoue, R. Ovalle-Robles, and Z. Liu (2019). Multi-Responsive and Multi-Motion Bimorph Actuator Based on Super-Aligned Carbon Nanotube Sheets. *Carbon 148*, 487–495.
6. Li J., **R. Zhang**, L. Mou, M. Jung de Andrade, X. Hu, K. Yu, J. Sun, T. Jia, Y. Dou, H. Chen, S. Fang, D. Qian, and Z. Liu (2019). Photothermal Bimorph Actuators with in-Built Cooler for Light Mills, Frequency Switches, and Soft Robots. *Advanced Functional Materials 29*(27), 1808995.
7. Wang R., Z. Liu, G. Wan, T. Jia, C. Zhang, X. Wang, M. Zhang, D. Qian, M. Jung de Andrade, N. Jiang, S. Yin, **R. Zhang**, D. Feng, W. Wang, H. Zhang, H. Chen, Y. Wang, R. Ovalle-Robles, K. Inoue, H. Lu, S. Fang, R. H. Baughman, and Z. Liu (2019). Controllable Preparation of Ordered and Hierarchically Buckled Structures for Inflatable Tumor Ablation, Volume Strain Sensor, and Communication via Inflatable Antenna. *ACS Applied Materials and Interfaces 11*, 10862–10873.
8. Wada, S., **R. Zhang**, S. R. Mannava, V. K. Vasudevan, and D. Qian (2018). Simulation based prediction of cyclic failure in rubbery materials using nonlinear space-time finite element method coupled with continuum damage mechanics. *Finite Elements in Analysis and Design 138*, 21–30.
9. Falin A., Q. Cai, E.J.G. Santos, D. Scullion, D. Qian, **R. Zhang**, Z. Yang, S. Huang, K. Watanabe, T. Taniguchi, M. R. Barnett, Y. Chen, R. S. Ruoff, and L. H. Li (2017). Mechanical properties of atomically thin boron nitride and the role of interlayer interactions. *Nature Communications 8*, 15815.

10. **Zhang, R.**, L.Wen, S. Naboulsi, T. Eason, V. K. Vasudevan, and D. Qian (2016). Accelerated multiscale space-time finite element simulation and application to high cycle fatigue life prediction. *Computational Mechanics 58*(2), 329–349.
11. Liu Z., S. Fang, F. A. Moura, J. N. Ding, N. Jiang, J. Di, M. Zhang, X. Lepró, D. S. Galvão, C. S. Haines, N. Y. Yuan, S. G. Yin, D. W. Lee, R. Wang, H. Y. Wang, W. Lv, C. Dong, R. C. Zhang, M. J. Chen, Q. Yin, Y. T. Chong, **R. Zhang**, X. Wang, M. D. Lima, R. Ovalle-Robles, D. Qian, H. Lu, and R. H. Baughman (2015). Hierarchically buckled sheath-core fibers for superelastic electronics, sensors, and muscles. *Science 349*(6246), 400–404.
12. **Zhang R.**, L. Wen, and J. Xiao (2015). GPU-accelerated boundary element method for large-scale problems in Acoustics. *Chinese Journal of Computational Physics 32*(3), 299–309.
13. **Zhang R.**, L. Wen, L. Yang, and J. Xiao (2014). Realization of computational homogenization for thermal conductivity of composites. *Acta Materiae Compositae Sinica 31*(6), 1581–1587.

Patents:

1. **Zhang R**., L. Wen, Z. Tang, and Y. Lu. Method for multiscale calculation of equivalent stiffness matrices of complex composite material structures. C.N. Patent 2016103733893, filed May 31, 2016, and issued July 14, 2020.
2. **Zhang R**., L. Wen, Z. Tang, and Y. Lu. Multiscale calculation method for equivalent thermal expansion coefficient of complicated composite material structure. C.N. Patent 2016103738929, filed May 31, 2016, and issued July 3, 2020.
3. **Zhang R**., L. Wen, Z. Tang, and Y. Lu. Multiscale calculation method for equivalent heat conduction coefficient of complicated composite material structure. C.N. Patent 2016103738399, filed May 31, 2016, and issued June 16, 2020.
4. **Zhang R**., L. Wen, Z. Tang, and Y. Lu. Complex composite material structure equivalent material performance multi-scale calculating method. C.N. Patent 2016103738401, filed May 31, 2016, and issued June 21, 2019.

Conference Presentations (as the presenting author):

1. **Zhang R.**, Y. Liu, J. Gao, D. Suarez, S. Saha, W. K. Liu, and D. Qian (2020). A multiscale approach to high cycle fatigue life prediction based on extended space-time finite element method and self-consistent clustering analysis. *ASME International Mechanical Engineering Congress and Exposition*, Virtual Conference, Online, November 16–19.
2. **Zhang R.**, C. Nicely, and D. Qian (2020). Multiscale Dynamic Fracture Simulation Based on Concurrent Coupling of Peridynamics and Extended Finite Element Method. *ASME International Mechanical Engineering Congress and Exposition*, Virtual Conference, Online, November 16–19.

3. **Zhang R.**, S. Wada, C. Nicely, and D. Qian (2019). Concurrent Multiscale Coupling of Peridynamic with Finite Element Method for Fracture Simulations. *ASME International Mechanical Engineering Congress and Exposition*, Salt Lake City, UT, November 11–14.
4. **Zhang R.**, S. Wada, and D. Qian (2019). A multiscale simulation approach to high cycle fatigue failure prediction. *19th International Conference on New Trends in Fatigue and Fracture*, Tucson, Arizona, October 8–10.
5. **Zhang R.**, S. Wada, C. Nicely, and D. Qian (2019). Coupled Peridynamics/Finite Element Method for Multiscale Fracture Simulations. *15th US National Congress on Computational Mechanics*, Austin, TX, July 28–August 1.
6. **Zhang R.**, S. Wada, C. Nicely, and D. Qian (2018). Multiscale Fracture Simulation based on Coupled Space-Time Finite Element Method and Peridynamics. *ASME International Mechanical Engineering Congress and Exposition*, Pittsburgh, PA, November 9–15.
7. **Zhang R.**, L. Wen, and D. Qian (2018). A Multi-Temporal Scale Approach to Thermomechanical Fatigue Failure Prediction. *ASME International Mechanical Engineering Congress and Exposition*, Pittsburgh, PA, November 9–15.
8. **Zhang R.**, S. Wada, C. Nicely, and D. Qian (2018). A Multiscale Space-time Computational Framework for Fracture and Fatigue Failure Simulation. *13th World Congress on Computational Mechanics*, New York City, NY, July 22–27.
9. **Zhang R.**, S. Wada, C. Nicely, and D. Qian (2018). Structural and Material Failure Prediction Based on Multiscale Space-time Approach. *18th U.S. National Congress for Theoretical and Applied Mechanics*, Chicago, IL, June 4–9.
10. Nicely C., S. Wada, **R. Zhang**, S. Tang, and D. Qian (2018). Nonlocal Matching Boundary Conditions for Concurrently Coupled Peridynamic/FEM Simulation. *18th U.S. National Congress for Theoretical and Applied Mechanics*, Chicago, IL, June 4–9.
11. Wada S., C. Nicely, **R. Zhang**, and D. Qian (2017). A Multiscale Approach to Fracture Based on Coupled Peridynamic/FEM Simulation (Part I). *ASME International Mechanical Engineering Congress and Exposition*, Tampa, FL, November 3–9.
12. Wada, S, C. Nicely, **R. Zhang**, and D. Qian (2017). A Multiscale Approach to Fracture Based on Coupled Peridynamic/FEM Simulation (Part II). *ASME International Mechanical Engineering Congress and Exposition*, Tampa, FL, November 3–9.
13. **Zhang R.**, J. Xiao, L. Wen, and D. Qian (2017). A Multi-Temporal Scale Approach to Thermomechanical Failure and Response Prediction. *14th US National Congress on Computational Mechanics*, Montreal, QC, Canada, July 17–20.
14. **Zhang R.**, L. Wen, and D. Qian (2015). Accelerated simulation of structural/material dynamics based on space-time finite element method. *13th US National Congress on Computational Mechanics*, San Diego, CA, July 26–30.