Computational Mechanics (2019) 63:455–470
https://doi.org/10.1007/s00466-018-1603-8

ORIGINAL PAPER

An efficient solution algorithm for space–time finite element method

Rui Zhang[1,2] · Lihua Wen[1] · Jinyou Xiao[1] · Dong Qian[2]

Received: 11 March 2018 / Accepted: 3 July 2018 / Published online: 14 July 2018

Abstract
An efficient solution algorithm has been developed for space–time finite element method that is derived from time discontinuous Galerkin (TDG) formulation. The proposed algorithm features an iterative solver accelerated by a novel and efficient preconditioner. This preconditioner is constructed based on the block structure of coupled space–time system matrix, which is expressed as addition of Kronecker products of temporal and spatial submatrices. With this unique decomposition, the most computationally intensive operations in the iterative solver, i.e. matrix operations, are subsequently optimized and accelerated employing the inverse property of Kronecker product. Theoretical analysis and numerical examples both demonstrate that the proposed algorithm provides significantly better performance than the already developed implementations for TDG-based space–time FEM. It reduces the computational cost of solving space–time equations to the same order of solving stiffness equations associated with regular FEM, thereby enabling practical implementation of the space–time FEM for engineering applications.

Keywords Time-discontinuous Galerkin formulation · Space–time FEM · Kronecker product · Preconditioner

1 Introduction

In the development of computational approaches for resolving structural and material dynamical responses, space–time finite element (STFEM) is a unique method in its approach to a wide range of temporal scales. Compared with the traditional FEM based on semi-discrete schemes, the temporal domain in STFEM is discretized with mesh and approximated by the FEM shape functions. This concept of temporal discretization was first introduced in the time finite element method [1–3] derived based on Hamilton's principle for dynamics during the late 1960s—shortly after the establishment of regular FEM in the early 1960s. 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 [4]. 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 was first partitioned into smaller space–time slabs. A Galerkin formulation was then established within each space–time slab. The resulting formulation is called a time-discontinuous Galerkin (TDG) formulation [5, 6] 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 [5, 6]. It was then extended to second-order hyperbolic systems in [7–12]. Based on the choice of unknown fields, two different formulations, i.e., single-field and two-field formulations have been developed [7, 8]. It was shown that TDG method significantly reduces artificial oscillations that are associated with semi-discrete schemes in capturing sharp gradients or discontinuities [9]. In addition, the method is both higher order accurate and unconditionally stable [6–17]. Based on the key concepts introduced in generalized FEM [18], extended FEM [19] and partition of unity FEM [20, 21], the widely adopted polynomial based approximations in STFEM can be further enhanced using enrichment functions that represent the problem physics [22–28]. This enriched formulation is termed the extended space–time finite element method (XTFEM). Within the single-field TDG framework, enrichment func-

✉ Dong Qian
dong.qian@utdallas.edu

[1] School of Astronautics, Northwestern Polytechnical University, Xi'an 710072, People's Republic of China

[2] Department of Mechanical Engineering, The University of Texas at Dallas, Richardson, TX 75080, USA

tions such as harmonic function and Heaviside function have been introduced for coupled atomistic/continuum simulations of lattice fracture [23], dynamical responses with multi-temporal scales [24, 25] and direct numerical simulations of high-cycle fatigue failure in both metallic [26, 27] and rubbery [28] materials. Both STFEM and XTFEM provided stable and accurate predictions while employing time step sizes orders of magnitude greater than that of semi-discrete methods for solving either linear or nonlinear problems [26–28].

Although the robustness of the space–time method has been extensively demonstrated, a critical barrier for the extensive and practical application has been the large computational cost associated with the additional time dimension that is introduced [8]. As such, the extended predicative capability of the method is paid at the price of converting an n-dimensional spatial problem to an $n+1$ dimensional problem. Compared with the regular FEM, TDG method typically leads to larger system of coupled equations that are expressed in the form of $\mathcal{K}\mathbf{d} = \mathcal{F}$ in which $\mathcal{K}$ is space—time stiffness matrix of size $N \times N$ and non-symmetric. Assuming quadratic interpolation in time and n_s the number of spatial degrees of freedom (DOFs), then $N = 3n_s$ and $6n_s$ respectively for single-field STFEM and XTFEM. Obtaining solutions to the space–time stiffness equation requires $\mathrm{O}(N^3)$ operations and $\mathrm{O}(N^2)$ storage if direct solver is employed, which are several orders of magnitude higher comparing to solving the regular stiffness equation in FEM.

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

$$\mathcal{K} = \mathbf{\Phi} \otimes \mathbf{K} + \mathbf{\Psi} \otimes \mathbf{M} \tag{1}$$

in which $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are temporal submatrices, $\mathbf{K}$ and $\mathbf{M}$ are spatial stiffness and mass submatrices, and symbol $\otimes$ denotes the Kronecker product. It should be noted that $\mathbf{K}$ and $\mathbf{M}$ have exactly the same form as their counterparts in the regular FEM. In the case of two-field formulation, $\mathcal{K}$ is weakly coupled, i.e., there is no single block in $\mathcal{K}$ coupled to both $\mathbf{K}$ and $\mathbf{M}$. Based on this feature, a family of iterative predictor/multi-corrector algorithms have been developed [12, 16, 29]. In these algorithms, the original equations are first recast into a partially decoupled form. Subsequently stationary iterative method, such as Gauss-Jacobi method [12, 29] or Gauss—Seidel method [16], is applied to the multi-corrector phase. However, these algorithms are not directly applicable to the single-field formulation and XTFEM as the corresponding space–time stiffness matrix is fully coupled. Alternatively, a general preconditioned iterative solution algorithm was developed in [27]. It first constructs the preconditioner by incomplete factorization of space–time stiffness matrix $\mathcal{K}$, during which the computational cost is minimized using matrix reordering algorithms based on Graph theory. The preconditioned space–time linear system of 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 formulation and XTFEM on 2D problems. However, the approach did not take into account of the unique block structure of $\mathcal{K}$ as indicated by Eq. (1). Further benchmark tests yielded degraded performance for 3D problems with this algorithm, which was due to the fact that the condition number of $\mathcal{K}$ in 3D has significantly deteriorated when compared with the 2D cases.

Based on the prior efforts, the main objective of this work is to establish an efficient solution algorithm to significantly scale down the computational cost of TDG methods for 3D and large-scale problems. This goal is realized by further exploiting the unique block structure of coupled space–time matrix equations. The proposed algorithm has two key components. First, a novel and efficient 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 computationally intensive operations associated with iterative solvers, are optimized and accelerated employing the inverse property of Kronecker product. Computational cost of the resulting algorithm is analyzed theoretically first and then demonstrated in both 2D and 3D numerical examples using various versions of the TDG method. It is shown that performance of the proposed algorithm is at least 1–2 orders of magnitude better than that of either direct sparse solver or the previously developed iterative approach [27] for problems with relatively large number of unknowns (e.g., $N > 10^4$). Through this implementation, the computational cost of solving the space–time stiffness equations is reduced to the same order as solving the corresponding stiffness equations in regular FEM, thereby enabling practical application of STFEM.

The rest of this paper is organized as follows. In Sect. 2, we review the space–time FEM based on various TDG formulations and the resulting coupled space–time stiffness matrices. In Sect. 3, we introduce the proposed solution approach and analyze its computational cost. Numerical examples and discussions are provided in Sect. 4. Finally, conclusions are drawn in Sect. 5.

2 Space–time finite element method

We start by briefly reviewing the TDG formulations. More details can be found in [8, 9, 25, 26].

2.1 Governing equations

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

$$\rho\ddot{\mathbf{u}} = \nabla \cdot \boldsymbol{\sigma}(\nabla \mathbf{u}) + \mathbf{f} \quad \text{on} \quad Q \equiv \Omega \times]0, T[\tag{2}$$

$$\mathbf{u} = \bar{\mathbf{u}} \quad \text{on} \quad \Upsilon_u \equiv \Gamma_u \times]0, T[\tag{3}$$

$$\mathbf{n} \cdot \boldsymbol{\sigma}(\nabla \mathbf{u}) = \mathbf{t} \quad \text{on} \quad \Upsilon_t \equiv \Gamma_t \times]0, T[\tag{4}$$

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

$$\dot{\mathbf{u}}(\mathbf{x}, 0) = \mathbf{v}_0(\mathbf{x}) \quad \text{for} \quad \mathbf{x} \in \Omega \tag{6}$$

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

2.2 Space–time discretization

In TDG method, the space–time domain is 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 $I_n = \left]t_{n-1}, t_n\right[$. Displacement and traction boundary conditions are defined on $(\Upsilon_u)_n = \Gamma_u \times I_n$ and $(\Upsilon_t)_n = \Gamma_t \times I_n$ respectively. Space—time slab Q_n is further discretized into $(n_{el})_n$ space–time elements. The approximations established will be denoted with a superscript "h". The domain (interior) of the eth element defined as $Q_n^e \subset Q_n$ and its boundary as Υ_n^e. The domain and boundary of the interior of the slab are defined as $Q_n^\Sigma = \bigcup_{e=1}^{(n_{el})_n} Q_n^e$ and $\Upsilon_n^\Sigma = \bigcup_{e=1}^{(n_{el})_n} \Upsilon_n^e - \Upsilon_n$ respectively.

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

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

$$a\left(\mathbf{w}^h, \mathbf{u}^h\right)_\Omega = \int_\Omega \nabla \mathbf{w}^h \cdot \boldsymbol{\sigma}\left(\nabla \mathbf{u}^h\right) d\Omega \tag{8}$$

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

$$a\left(\mathbf{w}^h, \mathbf{u}^h\right)_{Q_n} = \int_{Q_n} \nabla \mathbf{w}^h \cdot \boldsymbol{\sigma}\left(\nabla \mathbf{u}^h\right) dQ \tag{10}$$

$$\left(\mathbf{w}^h, \mathbf{u}^h\right)_{Q_n^\Sigma} = \int_{Q_n^\Sigma} \mathbf{w}^h \cdot \mathbf{u}^h dQ \tag{11}$$

$$\left(\mathbf{w}^h, \mathbf{u}^h\right)_{\Upsilon_n^\Sigma} = \int_{\Upsilon_n^\Sigma} \mathbf{w}^h \cdot \mathbf{u}^h d\Upsilon \tag{12}$$

$$\left(\mathbf{w}^h, \mathbf{u}^h\right)_{(\Upsilon_t)_n} = \int_{(\Upsilon_t)_n} \mathbf{w}^h \cdot \mathbf{u}^h d\Upsilon \tag{13}$$

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

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

$$[\![\mathbf{u}(\mathbf{x})]\!] = \mathbf{u}\left(\mathbf{x}^+\right) - \mathbf{u}\left(\mathbf{x}^-\right) \tag{15}$$

in which

$$\mathbf{u}\left(t_n^\pm\right) = \lim_{\varepsilon \to 0^\pm} \mathbf{u}(t_n \pm \varepsilon) \tag{16}$$

$$\mathbf{u}\left(\mathbf{x}^\pm\right) = \lim_{\varepsilon \to 0^\pm} \mathbf{u}(\mathbf{x} + \varepsilon \mathbf{n}) \tag{17}$$

$$\mathbf{n} = \mathbf{n}^+ = \mathbf{n}^- \tag{18}$$

2.3 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 $\mathbf{u}^h(\mathbf{x}, t)$ and test functions $\delta\mathbf{u}^h(\mathbf{x}, t)$ 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

$$\mathbf{u}^{\mathrm{h}}(\mathbf{x}, t) \in U$$

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

$$\delta\mathbf{u}^{\mathrm{h}}(\mathbf{x}, t) \in U_0$$

$$U_0 = \left\{ \delta\mathbf{u}^h(\mathbf{x}, t) \middle| \delta\mathbf{u}^h \in C^0\left(\bigcup_{n=1}^{N} Q_n\right),\ \delta\mathbf{u}^h = 0 \text{ on } \Gamma_u \right\} \tag{20}$$

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

$$B_{DG}\left(\delta\mathbf{u}^{\mathrm{h}}, \mathbf{u}^{\mathrm{h}}\right)_n = L_{DG}\left(\delta\mathbf{u}^{\mathrm{h}}\right)_n \tag{21}$$

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

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

$$L_{DG}\left(\delta\mathbf{u}^{\mathrm{h}}\right)_n = \left(\delta\dot{\mathbf{u}}^{\mathrm{h}}, \mathbf{f}\right)_{Q_n} + \left(\delta\dot{\mathbf{u}}^{\mathrm{h}}, \mathbf{t}\right)_{(\Upsilon_t)_n} + \left(\delta\dot{\mathbf{u}}^{\mathrm{h}}\left(t_{n-1}^{+}\right), \rho\dot{\mathbf{u}}^{\mathrm{h}}\left(t_{n-1}^{-}\right)\right)_{\Omega} + a\left(\delta\mathbf{u}^{\mathrm{h}}\left(t_{n-1}^{+}\right), \mathbf{u}^{\mathrm{h}}\left(t_{n-1}^{-}\right)\right)_{\Omega} \tag{23}$$

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

$$\mathbf{N}(\mathbf{x}, t) = \mathbf{N}_t \otimes \mathbf{N}_{\mathbf{x}} = \left[\, N_{t_1}\mathbf{N}_{\mathbf{x}} \;\ldots\; N_{t_i}\mathbf{N}_{\mathbf{x}} \;\ldots\; N_{t_k}\mathbf{N}_{\mathbf{x}} \,\right] \tag{24}$$

where $\mathbf{N}_{\mathbf{x}}$ and $\mathbf{N}_t$ are the spatial and temporal shape functions respectively. The spatial shape functions are the same as these in the regular FEM. A 3-node quadratic shape function is employed for $\mathbf{N}_t$ in the single-field formulation:

$$\mathbf{N}_t = \left[\, \frac{2(t_n - t)(t_{n-1/2} - t)}{\Delta t^2} \;\; \frac{-4(t_n - t)(t_{n-1} - t)}{\Delta t^2} \;\; \frac{2(t_{n-1} - t)(t_{n-1/2} - t)}{\Delta t^2} \,\right] \tag{25}$$

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.

Based on Eq. (24), integrations over the spatial domain can be done independent of the ones over the temporal domain. Therefore, the 1st and 2nd terms on right-hand side of Eq. (22) are given as

$$\begin{aligned}\left(\delta\dot{\mathbf{u}}^{\mathrm{h}}, \rho\ddot{\mathbf{u}}^{\mathrm{h}}\right)_{Q_n} &= \delta\mathbf{d}^T \left(\int_{Q_n} \dot{\mathbf{N}}^T \cdot \rho\ddot{\mathbf{N}} dQ\right)\mathbf{d} \\ &= \delta\mathbf{d}^T \left[\left(\int_{I_n} \dot{\mathbf{N}}_t^T \ddot{\mathbf{N}}_t dt\right) \otimes \mathbf{M}\right]\mathbf{d}\end{aligned} \tag{26}$$

$$\begin{aligned}a\left(\delta\dot{\mathbf{u}}^{\mathrm{h}}, \mathbf{u}^{\mathrm{h}}\right)_{Q_n} &= \delta\mathbf{d}^T \left(\int_{Q_n} \dot{\mathbf{N}}_{,\mathbf{x}}^T \cdot \mathbf{C}\mathbf{N}_{,\mathbf{x}} dQ\right)\mathbf{d} \\ &= \delta\mathbf{d}^T \left[\left(\int_{I_n} \dot{\mathbf{N}}_t^T \mathbf{N}_t dt\right) \otimes \mathbf{K}\right]\mathbf{d}\end{aligned} \tag{27}$$

in which $\delta\mathbf{d}$ is arbitrary virtual displacement that can be dropped from both sides of Eq. (21), $\mathbf{d}$ is nodal displacement vector, $\mathbf{K}$ and $\mathbf{M}$ are spatial stiffness and mass matrices, respectively. Given the quadratic temporal shape functions [Eq. (25)], integrations over time in Eqs. (26) and (27) can be evaluated analytically and the space–time stiffness matrix of Eq. (21) is further expressed as

$$\mathcal{K} = \begin{bmatrix} \frac{5\mathbf{M}}{\Delta t^2} + \frac{\mathbf{K}}{2} & -\frac{4\mathbf{M}}{\Delta t^2} - \frac{2\mathbf{K}}{3} & -\frac{\mathbf{M}}{\Delta t^2} + \frac{\mathbf{K}}{6} \\ -\frac{12\mathbf{M}}{\Delta t^2} + \frac{2\mathbf{K}}{3} & \frac{16\mathbf{M}}{\Delta t^2} & -\frac{4\mathbf{M}}{\Delta t^2} - \frac{2\mathbf{K}}{3} \\ \frac{7\mathbf{M}}{\Delta t^2} - \frac{\mathbf{K}}{6} & -\frac{12\mathbf{M}}{\Delta t^2} + \frac{2\mathbf{K}}{3} & \frac{5\mathbf{M}}{\Delta t^2} + \frac{\mathbf{K}}{2} \end{bmatrix} \tag{28}$$

Equation (28) 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 $\mathbf{K}$ and $\mathbf{M}$ with the exception of $\mathcal{K}_{22}$. It can also be expressed in the form of Kronecker products as shown below

$$\mathcal{K} = \frac{1}{6}\begin{bmatrix} 3 & -4 & 1 \\ 4 & 0 & -4 \\ -1 & 4 & 3 \end{bmatrix} \otimes \mathbf{K} + \frac{1}{\Delta t^2}\begin{bmatrix} 5 & -4 & -1 \\ -12 & 16 & -4 \\ 7 & -12 & 5 \end{bmatrix} \otimes \mathbf{M} \tag{29}$$

2.4 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 $\mathbf{U}^h(\mathbf{x}, t) = \left\{\mathbf{u}^h, \mathbf{v}^h\right\}$ and test functions $\delta\mathbf{U}^h(\mathbf{x}, t) = \left\{\delta\mathbf{u}^h, \delta\mathbf{v}^h\right\}$ 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

$$\mathbf{u}^{\mathrm{h}}(\mathbf{x}, t) \in U$$

$$U = \left\{ \mathbf{u}^h(\mathbf{x}, t) \middle| \mathbf{u}^h \in C^0\left(\bigcup_{n=1}^{N} Q_n\right),\ \mathbf{u}^h = \bar{\mathrm{u}} \text{ on } \Gamma_u \right\} \tag{30}$$

$$\mathbf{v}^{\mathrm{h}}(\mathbf{x}, t) \in V$$

$$V = \left\{ \mathbf{v}^h(\mathbf{x}, t) \middle| \mathbf{v}^h \in C^0\left(\bigcup_{n=1}^{N} Q_n\right),\ \mathbf{v}^h = \dot{\bar{\mathbf{u}}} \text{ on } \Gamma_u \right\} \tag{31}$$

and the spaces of the test functions are

$$\delta\mathbf{u}^{\mathrm{h}}(\mathbf{x}, t) \in U_0$$

$$U_0 = \left\{ \delta\mathbf{u}^h(\mathbf{x}, t) \middle| \delta\mathbf{u}^h \in C^0\left(\bigcup_{n=1}^{N} Q_n\right),\ \delta\mathbf{u}^h = 0 \text{ on } \Gamma_u \right\} \tag{32}$$

$$\delta\mathbf{v}^{\mathrm{h}}(\mathbf{x}, t) \in V_0$$

$$V_0 = \left\{ \delta\mathbf{v}^h(\mathbf{x}, t) \middle| \delta\mathbf{v}^h \in C^0\left(\bigcup_{n=1}^{N} Q_n\right),\ \delta\mathbf{v}^h = 0 \text{ on } \Gamma_u \right\} \tag{33}$$

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

$$B_{DG}\left(\delta\mathbf{U}^{\mathrm{h}}, \mathbf{U}^{\mathrm{h}}\right)_n = L_{DG}\left(\delta\mathbf{U}^{\mathrm{h}}\right)_n \tag{34}$$

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

$$\begin{aligned}B_{DG}\left(\delta\mathbf{U}^{\mathrm{h}}, \mathbf{U}^{\mathrm{h}}\right)_n &= \left(\delta\mathbf{v}^{\mathrm{h}}, \rho\dot{\mathbf{v}}^{\mathrm{h}}\right)_{Q_n} + a\left(\delta\mathbf{v}^{\mathrm{h}}, \mathbf{u}^{\mathrm{h}}\right)_{Q_n} \\ &\quad + a\left(\delta\mathbf{u}^{\mathrm{h}}, \left(\dot{\mathbf{u}}^{\mathrm{h}} - \mathbf{v}^{\mathrm{h}}\right)\right)_{Q_n} + \left(\delta\mathbf{v}^{\mathrm{h}}\left(t_{n-1}^{+}\right), \rho\mathbf{v}^{\mathrm{h}}\left(t_{n-1}^{+}\right)\right)_{\Omega} \\ &\quad + a\left(\delta\mathbf{u}^{\mathrm{h}}\left(t_{n-1}^{+}\right), \mathbf{u}^{\mathrm{h}}\left(t_{n-1}^{+}\right)\right)_{\Omega}\end{aligned} \tag{35}$$

$$\begin{aligned}L_{DG}\left(\delta\mathbf{U}^{\mathrm{h}}\right)_n &= \left(\delta\mathbf{v}^{\mathrm{h}}, \mathbf{f}\right)_{Q_n} + \left(\delta\mathbf{v}^{\mathrm{h}}, \mathbf{t}\right)_{(\Upsilon_t)_n} + \left(\delta\mathbf{v}^{\mathrm{h}}\left(t_{n-1}^{+}\right), \rho\mathbf{v}^{\mathrm{h}}\left(t_{n-1}^{-}\right)\right)_{\Omega} \\ &\quad + a\left(\delta\mathbf{u}^{\mathrm{h}}\left(t_{n-1}^{+}\right), \mathbf{u}^{\mathrm{h}}\left(t_{n-1}^{-}\right)\right)_{\Omega}\end{aligned} \tag{36}$$

The same decomposition of space–time shape function [Eq. (24)] is employed in the two-field formulation. Linear temporal shape functions are defined for both the displacement and velocity fields, i.e. the P1–P1 element in [8, 9] is adopted. The corresponding space–time stiffness matrix is obtained as

$$\mathcal{K} = \begin{bmatrix} \frac{1}{2}\mathbf{K} & \frac{1}{2}\mathbf{K} & -\frac{\Delta t}{3}\mathbf{K} & -\frac{\Delta t}{6}\mathbf{K} \\ -\frac{1}{2}\mathbf{K} & \frac{1}{2}\mathbf{K} & -\frac{\Delta t}{6}\mathbf{K} & -\frac{\Delta t}{3}\mathbf{K} \\ \frac{\Delta t}{3}\mathbf{K} & \frac{\Delta t}{6}\mathbf{K} & \frac{1}{2}\mathbf{M} & \frac{1}{2}\mathbf{M} \\ \frac{\Delta t}{6}\mathbf{K} & \frac{\Delta t}{3}\mathbf{K} & -\frac{1}{2}\mathbf{M} & \frac{1}{2}\mathbf{M} \end{bmatrix} \tag{37}$$

Similarly, Eq. (37) can be rewritten as

$$\mathcal{K} = \frac{1}{6}\begin{bmatrix} 3 & 3 & -2\Delta t & -\Delta t \\ -3 & 3 & -\Delta t & -2\Delta t \\ 2\Delta t & \Delta t & 0 & 0 \\ \Delta t & 2\Delta t & 0 & 0 \end{bmatrix} \otimes \mathbf{K} + \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \otimes \mathbf{M} \tag{38}$$

As shown in Eq. (37), the space–time stiffness matrix of the two-field formulation is weakly coupled, i.e., each term involves either **K** or **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 [12, 16, 29].

2.5 Enriched formulation

The enriched space–time approximation established in XTFEM is given as

$$\mathbf{u}(\mathbf{x}, t) = \sum_{I=1}^{n_s} \mathbf{N}_I(\mathbf{x}, t)\mathbf{d}_I + \sum_{J=1}^{n_e} \tilde{\mathbf{N}}_J(\mathbf{x}, t)\mathbf{a}_J \tag{39}$$

where $\mathbf{a}$ represents the enriched DOFs, n_s and n_e are the numbers of standard and enriched DOFs respectively. For the J-th node the enriched shape function is

$$\tilde{\mathbf{N}}_J(\mathbf{x}, t) = \mathbf{N}_J(\mathbf{x}, t)\Phi_J(\mathbf{x}, t) \tag{40}$$

in which

$$\Phi_J(\mathbf{x}, t) = \Phi(\mathbf{x}, t) - \Phi(\mathbf{x}_J, t_J) \tag{41}$$

With prior knowledge about problem physics, proper enrichment functions can be selected. For example, in fatigue life prediction problems, the oscillating components in structural response cannot be efficiently captured by conventional polynomials-based shape functions. Considering the cyclic nature of fatigue loading, a time-dependent harmonic enrichment function can be introduced. Similarly, discontinuous enrichment functions, e.g. Heaviside function, are suitable for dynamic responses with discontinuities or sharp gradients, such as shock wave propagation and fracture problems.

With the introduction of enrichment function, the space—time stiffness matrix can be generally expressed as

$$\mathcal{K}_\mathbf{e} = \begin{bmatrix} \mathcal{K} & \mathcal{K}_{\mathbf{ea}} \\ \mathcal{K}_{\mathbf{eb}} & \mathcal{K}_{\mathbf{ee}} \end{bmatrix} \tag{42}$$

where $\mathcal{K}$ is the same space–time matrix as in Eq. (28) of the single-field formulation, $\mathcal{K}_{\mathbf{ea}}$ and $\mathcal{K}_{\mathbf{eb}}$ reflect the coupling between enriched and regular DOFs, $\mathcal{K}_{\mathbf{ee}}$ reflects the coupling between enriched DOFs. Equation (42) can also be written in the form of Kronecker product as shown in Eq. (29). The temporal submatrices are more complex due to the enrichment functions. In the current implementation, these submatrices are evaluated analytically.

2.6 Numerical implementation

Table 1 illustrates the numerical implementation of space—time FEM. Computational cost analysis in previous study [27] showed that solving the space–time linear system of equations (line 9 in Table 1) is the most expensive part. It requires $O(N^2)$ in memory for full storage of the space–time matrix and $O(N^3)$ in CPU time for solving the equations using direct solver, where N is the total number of space—time DOFs.

Figure 1a–d illustrate sparsity patterns from the stiffness matrices that are obtained from regular FEM, single-field and two-field STFEM and XTFEM for the same number of spatial nodes. These patterns are generated based on a thin plate problem that will be described in Sect. 4.1. For spatial discretization, a 3D structured mesh (2880 8-node brick elements, 3965 nodes) is generated and leads to the banded pattern of stiffness matrix for regular FEM as shown Fig. 1a. The number of spatial DOFs in regular FEM is $n_s = 11{,}895$.

Table 1 Implementation of space–time FEM

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

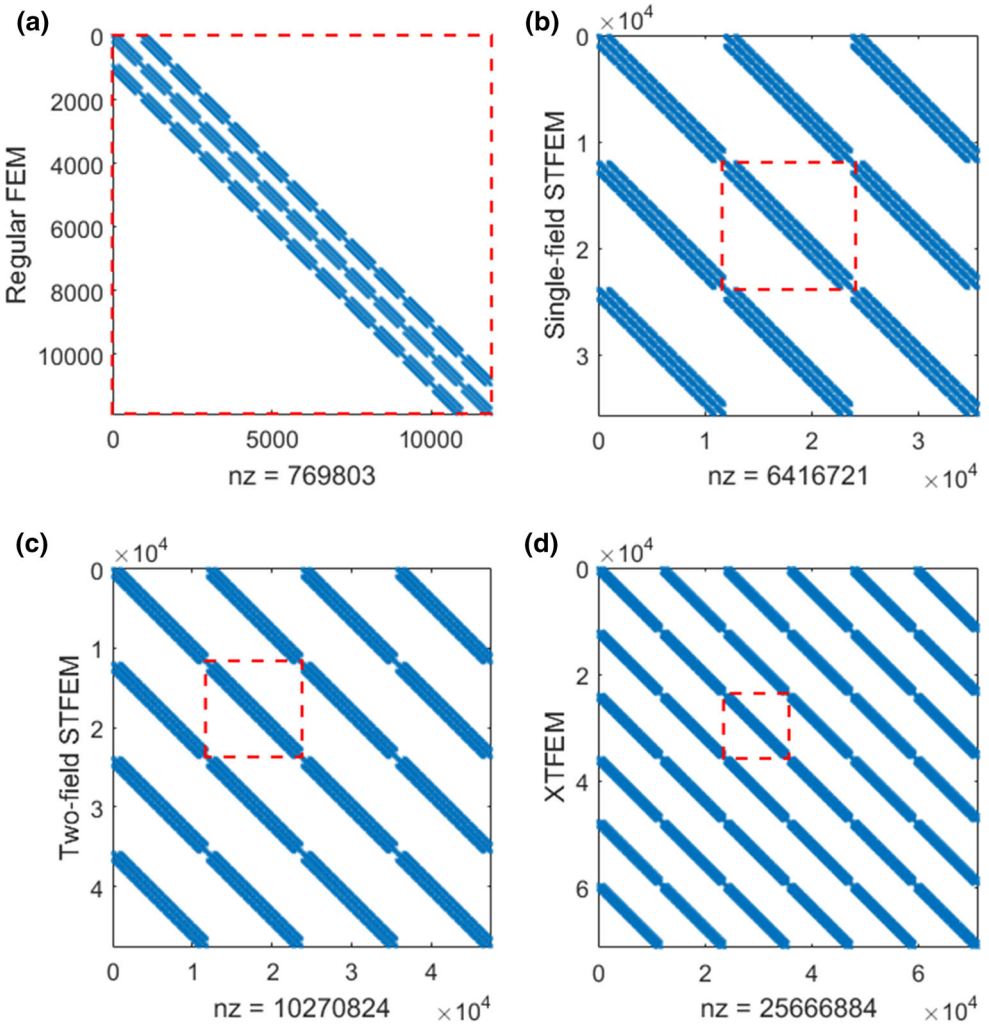

Fig. 1 Comparison among the sparse pattern of the stiffness matrices formed by **a** regular FEM, **b** single-field STFEM, **c** two-field STFEM and **d** XTFEM, dashed box indicates the size of the regular FEM stiffness matrix and *nz* represents the number of non-zero elements in the sparse matrix

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. As can be seen, direct solution of space–time FEM stiffness equation leads to computational cost that is several orders of magnitude higher comparing to solving the regular stiffness equation in FEM, which becomes a critical barrier for its extensive and practical application.

3 Proposed solution algorithm

For each space–time slab, the general linear system of equations derived from TDG methods is given as

$$\mathcal{K}\mathbf{d} = \mathcal{F} \tag{43}$$

where $\mathcal{K} \in \mathbb{R}^{N\times N}$ is a non-symmetric sparse matrix, $\mathbf{d} \in \mathbb{R}^N$ and $\mathcal{F} \in \mathbb{R}^N$ are unknowns and force vectors respectively. Considering the special block structure of space–time matrix, Eq. (43) can be rewritten as

$$(\mathbf{\Phi} \otimes \mathbf{K} + \mathbf{\Psi} \otimes \mathbf{M})\mathbf{d} = \mathcal{F} \tag{44}$$

In this work, we employ the nonstationary iterative approach to solve Eq. (44). It is well known that the efficiency and robustness of iterative methods largely depend on the quality of preconditioner. Therefore, a preconditioner is constructed and the original system is modified as

$$\left[\mathcal{P}^{-1}(\mathbf{\Phi} \otimes \mathbf{K} + \mathbf{\Psi} \otimes \mathbf{M})\right]\mathbf{d} = \mathcal{P}^{-1}\mathcal{F} \tag{45}$$

in which matrix $\mathcal{P}$ is the preconditioner and Eq. (45) is the left preconditioned system.

3.1 Preconditioned iterative method

Due to the asymmetry of space–time stiffness matrix, we employ the generalized minimum residual method (GMRES) as the iterative solver. GMRES is developed by Saad and Schultz [30] and widely used for asymmetric systems of linear equations. It starts from an initial guess $\mathbf{d}_0$. Then, initial residual is calculated by $\mathbf{r}_0 = \mathcal{P}^{-1}(\mathcal{F}-\mathcal{K}\mathbf{d}_0)$. The exact solution is approximated by

$$\mathbf{d}^{(i)} = \mathbf{d}_0 + y_1\mathbf{v}^{(1)} + \cdots + y_i\mathbf{v}^{(i)} \tag{46}$$

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

$$\mathcal{K}_m(\mathcal{P}^{-1}\mathcal{K}, \mathbf{r}_0) = span\left\{\mathbf{r}_0, \mathcal{P}^{-1}\mathcal{K}\mathbf{r}_0, \left(\mathcal{P}^{-1}\mathcal{K}\right)^2\mathbf{r}_0, \ldots, \left(\mathcal{P}^{-1}\mathcal{K}\right)^{m-1}\mathbf{r}_0\right\} \tag{47}$$

and y_i is solved from a linear least-squares problem that minimizes the residual. The bases are generated by modified Gram–Schmidt orthogonalization procedure in the Arnoldi iteration. The iterative procedure is stopped once residual satisfies a given error criterion. To save storage and computing resources, GMRES is restarted every m iterations. The left preconditioned GMRES algorithm is summarized in Table 2.

Computational cost of GMRES 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. Total cost is the product of these two factors. Therefore, a good preconditioner needs 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 2.

Table 2 Left preconditioned GMRES algorithm

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

3.2 Proposed preconditioner

The preconditioner is built by approximating the system matrix (the space–time stiffness matrix). A good preconditioner makes the modified linear system easier to solve and the computational saving gained far outweighs the extra cost of preconditioning. In previous study [27], a general method of preconditioning is established by the incomplete lower/upper (ILU) factorization of the space–time stiffness matrix $\mathcal{K}$. Since the choice and the quality of preconditioner greatly depend on the specific linear system, further improvements can be made by taking advantage of the special features of the space–time stiffness matrix. Several observations are made from the space–time stiffness matrices shown in Sect. 2:

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

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

$$\mathcal{P} = \mathbf{\Phi}\otimes\mathbf{P} \tag{48}$$

in which matrix $\mathbf{P}$ is derived by incomplete factorization of matrix $\mathbf{K}$. Since $\mathbf{K}$ is symmetric positive definite (SPD), several well-established incomplete factorization methods, such as the ILU method for general matrices and the incomplete Cholesky factorization (ICHOL) method, can be directly used. Matrix $\mathbf{P}$ can be then expressed as

$$\mathbf{P} = \mathbf{LU} = \mathbf{K} - \mathbf{R} \quad \text{for} \quad ILU \tag{49}$$

or

$$\mathbf{P} = \mathbf{LL}^T = \mathbf{K} - \mathbf{R} \quad \text{for} \quad ICHOL \tag{50}$$

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

The advantages of using the preconditioner as proposed by Eq. (48) are twofold. First, the preconditioner is obtained by incomplete factorization of spatial stiffness matrix $\mathbf{K}$ which is sparse, symmetric and better conditioned. Therefore, the computational cost is far less than the general approach of obtaining the preconditioner by incomplete factorization of the larger, coupled space–time matrix $\mathcal{K}$ that is also non-symmetric. Second, the preconditioning operation, i.e., $\mathcal{P}^{-1}\mathbf{v}$ with $\mathbf{v}$ a vector involved in the iteration, can be greatly simplified and accelerated by using properties of Kronecker product as shown later in Sect. 3.3.2.

3.3 Acceleration of matrix operations

As mentioned in Sect. 3.1, the most computationally intensive part in GMRES iterations is the following matrix operation,

$$\mathbf{w} = \mathcal{P}^{-1}(\mathcal{K}\mathbf{v}) \tag{51}$$

This operation can be further decomposed into two matrix operations. The first is the matrix–vector multiplication of $\mathcal{K}\mathbf{v}$. The second is the preconditioning operation. Direct evaluations of these matrix operations are less efficient. They can be optimized and accelerated by using the block structure of space–time matrix and inverse property of Kronecker product as follows.

3.3.1 Matrix–vector multiplication

The general form of matrix–vector multiplication is

$$\mathbf{y} = \mathcal{K}\mathbf{x} \tag{52}$$

in which $\mathbf{x}$ and $\mathbf{y}$ are vectors of size $N = n_\mathrm{t} \times n_\mathrm{s}$, 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.

By using the block structure of space–time matrix, Eq. (52) is rewritten as

$$\mathbf{y} = (\mathbf{\Phi} \otimes \mathbf{K} + \mathbf{\Psi} \otimes \mathbf{M})\mathbf{x} \tag{53}$$

The first step to simplify Eq. (53) is dividing vectors $\mathbf{x}$ and $\mathbf{y}$ into smaller segments,

$$\mathbf{x} = \begin{Bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(n_\mathrm{t})} \end{Bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{Bmatrix} \mathbf{y}^{(1)} \\ \mathbf{y}^{(2)} \\ \vdots \\ \mathbf{y}^{(n_\mathrm{t})} \end{Bmatrix} \tag{54}$$

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

Second, we define two intermediate vectors

$$\mathbf{u}^{(i)} = \mathbf{K}\mathbf{x}^{(i)} \quad \text{and} \quad \mathbf{v}^{(i)} = \mathbf{M}\mathbf{x}^{(i)}, \quad i = 1, 2, \ldots, n_\mathrm{t} \tag{55}$$

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

$$\mathbf{y}^{(i)} = \sum_{j=1}^{n_\mathrm{t}} \phi_{ij}\mathbf{u}^{(j)} + \sum_{j=1}^{n_\mathrm{t}} \psi_{ij}\mathbf{v}^{(j)}, \quad i = 1, 2, \ldots, n_\mathrm{t} \tag{56}$$

where ϕ_{ij} and ψ_{ij} are the corresponding elements in $\mathbf{\Phi}$ and $\mathbf{\Psi}$ matrices.

3.3.2 Preconditioning operation

The preconditioning operation is generally expressed as

$$\mathbf{y} = \mathcal{P}^{-1}\mathbf{x} \tag{57}$$

By using the preconditioner defined in Eq. (48), we have

$$\mathbf{y} = (\mathbf{\Phi} \otimes \mathbf{P})^{-1}\mathbf{x} \tag{58}$$

Due to the inverse property of Kronecker product, the preconditioning operation is rewritten as

$$\mathbf{y} = \left(\mathbf{\Phi}^{-1} \otimes \mathbf{P}^{-1}\right)\mathbf{x} \tag{59}$$

Similarly, we introduce an intermediate vector

$$\mathbf{z}^{(i)} = \mathbf{P}^{-1}\mathbf{x}^{(i)}, \quad i = 1, 2, \ldots, n_\mathrm{t} \tag{60}$$

From the solution of Eq. (60), the desired vector $\mathbf{y}$ is calculated by

$$\mathbf{y}^{(i)} = \sum_{j=1}^{n_\mathrm{t}} \varphi_{ij}\mathbf{z}^{(j)}, \quad i = 1, 2, \ldots, n_\mathrm{t} \tag{61}$$

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

3.4 Computational cost analysis

Computational cost of the proposed solution algorithm is mainly attributed to two major implementations: (1) generating the preconditioner and (2) the GMRES iterations. The cost of generating the preconditioner mostly comes from incomplete factorization of the **K** matrix. This implementation depends not only on the choice of incomplete factorization method but also on the system matrix to be factorized. However, this operation is the same as the corresponding in solving the static problems in the regular FEM assuming that the same type of preconditioned iterative solver is used. Therefore, the computational cost of generating the preconditioner is the same as the corresponding operation in the regular FEM.

As discussed in Sect. 3.3, GMRES iteration involves two main implementations: matrix–vector multiplication and preconditioning operation. Direct evaluation of matrix–vector product as shown in Eq. (52) requires $n_t n_s(n_t n_s - 1)$ additions and $(n_t n_s)^2$ multiplications. The proposed method as shown in Eqs. (54)–(56) reduces the numbers of operations to $n_t n_s(2n_t + 2n_s - 3)$ and $n_t n_s(2n_s + 2)$ respectively. For $n_s / n_t \gg 1$, cost of the proposed method is only $2/n_t$ of the direct evaluation method. Since explicit assembly of space—time matrix is no longer necessary in the present method, the memory cost reduces to $2/n_t^2$ of the full storage of space—time matrix. While the above discussions are based on the assumption of dense matrices, the order of the computational efficiency remains the same when sparse matrix format and operations are employed. Finally, from Eq. (60) it can be seen that the cost of preconditioning operation is n_t times the same operation in solving the corresponding static problems.

As a brief summary, we have demonstrated theoretically that the proposed solution algorithm reduces the cost of solving space–time linear system of equations to the same order of solving the corresponding static problems in regular FEM. Furthermore, numerous operations in the present method as shown in Eqs. (55), (56), (60) and (61) are well-suited and convenient for parallel computing.

4 Numerical examples

4.1 Problem statement

To demonstrate computational performance of the proposed solution algorithm, we study the problem of a rectangular thin plate with one end fixed and the other end subjected to a uniformly distributed load $p(t)$. Detailed geometric dimensions and boundary conditions of the problem are illustrated in Fig. 2. 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.

4.2 Accuracy and robustness of TDG methods

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 2880 elements and 3965 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 center difference method and the implicit Newmark-β ($\gamma = 0.5$, $\beta = 0.25$) method [31]. The TDG formulations described in Sect. 2, i.e., the single-field, two-field and enriched formulations are denoted as *TDG-1*, *TDG-2* and *TDG-e*, respectively. All those methods are implemented in MATLAB.

First, we consider the loading condition of a constant pressure, i.e. $p(t) = 100H(t)$ MPa, where $H(t)$ is the Heaviside function. The time steps for center difference, Newmark-β, *TDG-1* and *TDG-2* are 1.2×10^{-9}, 6.0×10^{-8}, 1.2×10^{-6} and 1.2×10^{-6} s, respectively. Figure 3 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 Fig. 4. 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.

Next, a fully-reversed cyclic load is considered, i.e. $p(t) = 100\sin(40\pi t)H(t)$ MPa. The time steps for center difference, Newmark-β, *TDG-1*, *TDG-2* and *TDG-e* methods are 7.2×10^{-8}, 5.0×10^{-5}, 6.3×10^{-3}, 6.3×10^{-3} 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.,

$$\Phi_J(t) = \sin(\omega t) - \sin(\omega t_J) \tag{62}$$

in which the frequency $\omega = 40\pi$ is same with the loading frequency. The displacement solutions are shown in Fig. 5. Although TDG methods use very large time steps compared with semi-discrete methods, they accurately captured the oscillating structural response. More extensive demonstration of the robustness of TDG methods can be found in Bhamare et al. [26].

4.3 Configurations for performance study

As the accuracy and robustness of TDG methods has been extensively demonstrated, we now focus on evaluating the computational performance of the proposed solution algorithm by using the configurations that are described as follows.

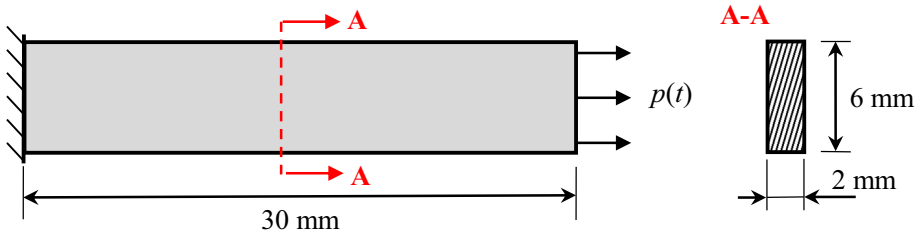

Fig. 2 The geometric dimensions and boundary conditions of the thin plate problem

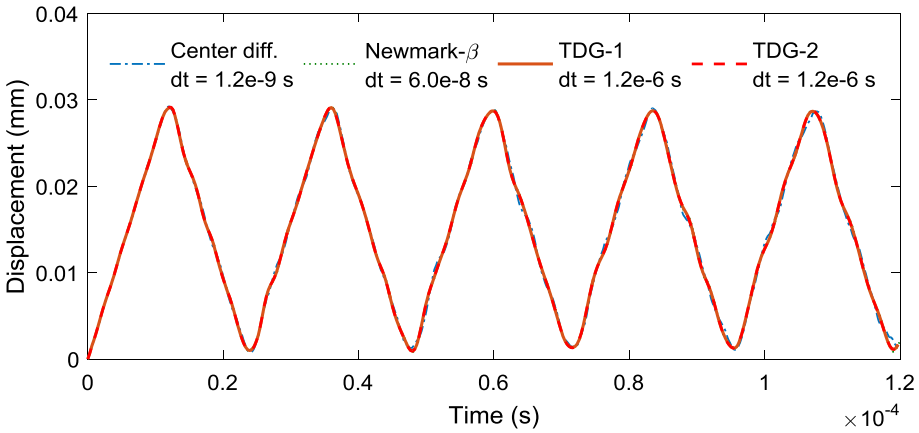

Fig. 3 Comparison of the displacement solutions at the right end of the plate

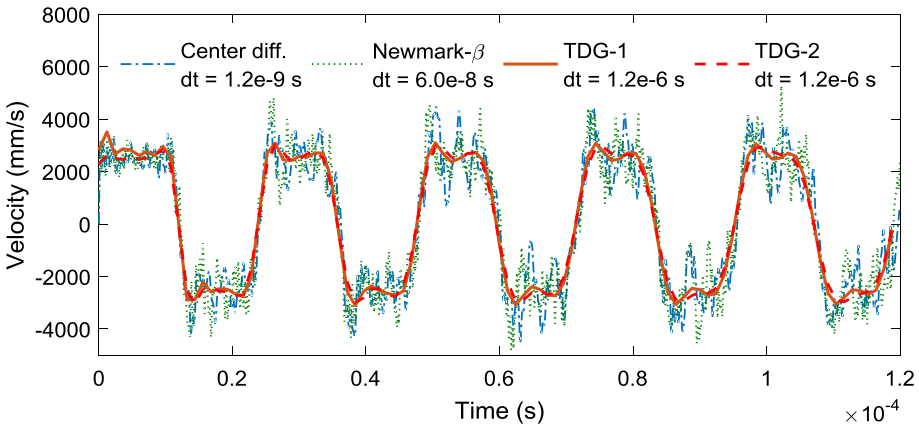

Fig. 4 Comparison of the velocity solutions at the right end of the plate

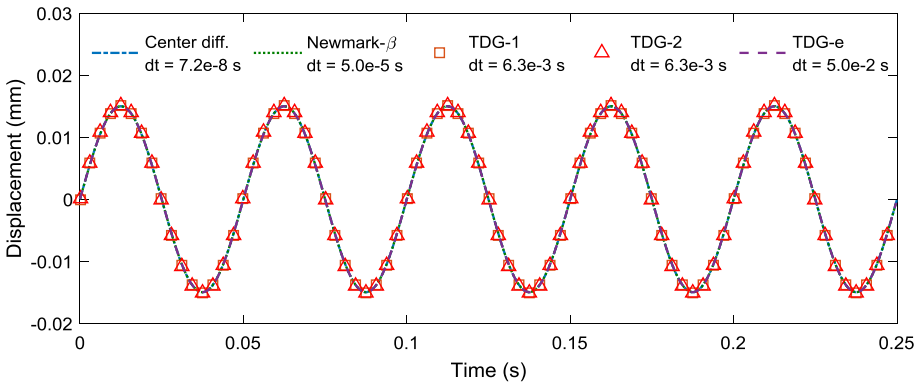

Fig. 5 Comparison of the displacement solutions under cyclic load

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 are used for 3D cases. Two sets of structured mesh grids are created by refining element size. All the TDG formulations, i.e., *TDG-1*, *TDG-2* and *TDG-e*, are employed to discretize the temporal domain.

Besides the proposed iterative solution algorithm, direct sparse solver (UMFPACK, a multi-frontal method [32]) and preconditioned iterative solver developed previously [27] are also employed for the purpose of comparison. 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 [33] and ILUTP—a more accurate variant of ILU algorithm are used in both the *Kronecker* and *Iterative* solvers. Same parameters associated with these two iterative solvers are used, see [27] for details.

For comparison purpose, the same problems are also simulated in semi-discrete FEM, in which the Newmark-β method is also employed. 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 as

$$\mathbf{K}_{\text{eff}} = \mathbf{K} + \frac{1}{\beta \Delta t^2}\mathbf{M} \tag{63}$$

The previously developed *Iterative* algorithm [27] is employed to solve the linear systems of equations with the stiffness matrix Eq. (63). Therefore, the result of theoretical cost analysis in Sect. 3.4 can be verified by comparing the computational performance of TDG methods with that of the implicit Newmark-β method.

The hardware platform for performance testing is a desktop workstation featured with the Intel Xeon E5-2623 v3 (3 GHz) CPU and 32 GB RAM. It was found that MATLAB (version R2015b) automatically accelerates some operations, e.g. those associated 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 run by starting MATLAB with the command line option—`singleCompThread`.

Both time and storage costs and complexities are presented in following sections. 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.

4.4 2D spatial discretization

We first evaluate the computational performance for the 2D cases. A summary of the mesh grids and the corresponding numbers of DOFs are presented in Table 3.

4.4.1 Single-field formulation

Computational performances of different solvers for the single-field formulation are plotted in Fig. 6. It shows that the previous proposed general *Iterative* solver achieves a similar efficiency as the multi-frontal *Direct* solver. The time complexities of these two solvers are both $O(N^{1.6})$. The present *Kronecker* solver demonstrates a significantly better performance. The time complexity is reduced to $O(N^{1.3})$. For $N > 10^4$, 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.

4.4.2 Two-field formulation

Figure 7 demonstrates the computational performances of different solvers for the two-field formulation. It shows that the *Kronecker* solver works more efficiently than the other solvers. The time complexity is reduced from $O(N^{1.7})$ to $O(N^{1.2})$. 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.

4.4.3 Enriched formulation

For the enriched formulation of XTFEM, the computational performances of different solvers are illustrated in Fig. 8. The *Direct* solver shows time complexity of $O(N^{1.6})$, which is slightly better than $O(N^{1.7})$ of the *Iterative* solver. The *Kronecker* solver further reduces the time complexity to $O(N^{1.2})$. The memory cost of the *Kronecker* solver is close to two orders of magnitude lower than the others.

To further demonstrate the 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 Fig. 9. 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.

4.5 3D spatial discretization

Next, we evaluate the computational performance of the proposed solution algorithm for 3D cases. A summary of the 3D mesh grids and corresponding numbers of DOFs are presented in Table 4.

Table 3 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	1260
3	0.75	320	369	738	2214	2952	4428
4	0.375	1280	1377	2754	8262	11,016	16,524
5	0.1875	5120	5313	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

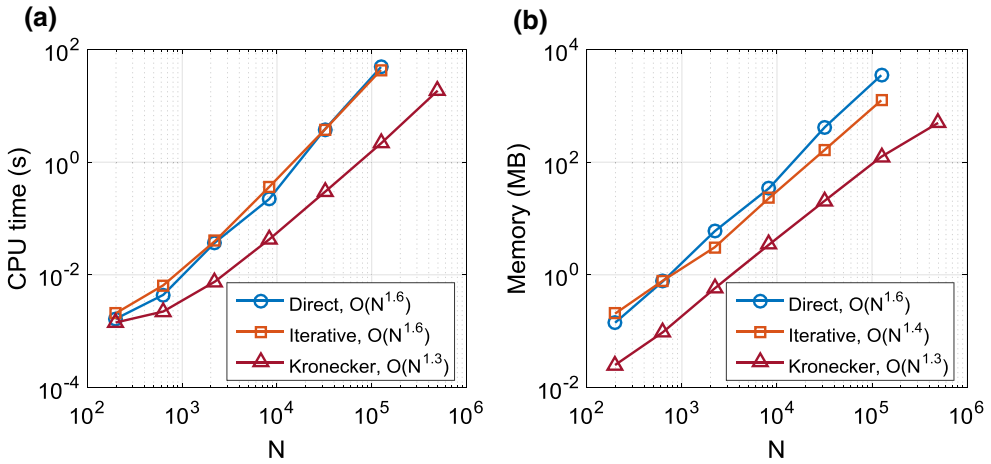

Fig. 6 Comparison of computational performance between different solvers for TDG-1 method: **a** CPU time and **b** memory usages

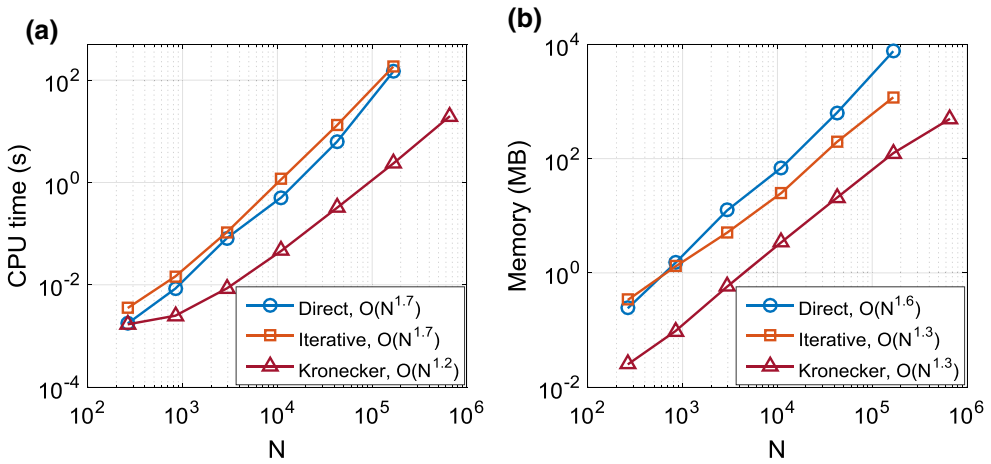

Fig. 7 Comparison of computational performances between different solvers for TDG-2 method: **a** CPU time and **b** memory usages

4.5.1 Single-field formulation

Computational performances of different solvers for the single-field formulations are shown in Fig. 10. The time efficiencies of the *Direct* solver and the *Iterative* solver are O($N^{2.5}$) and 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 deteriorated

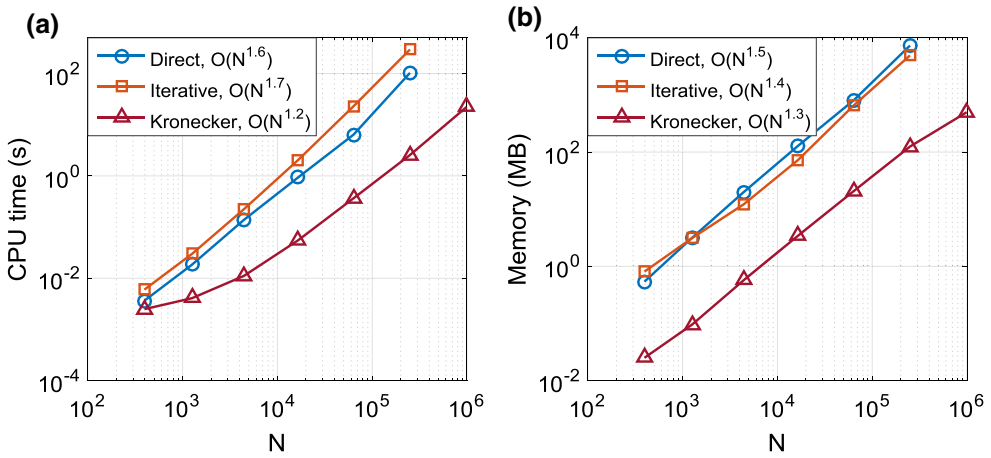

Fig. 8 Comparison of computational performances between different solvers for TDG-e method: **a** CPU time and **b** memory usages

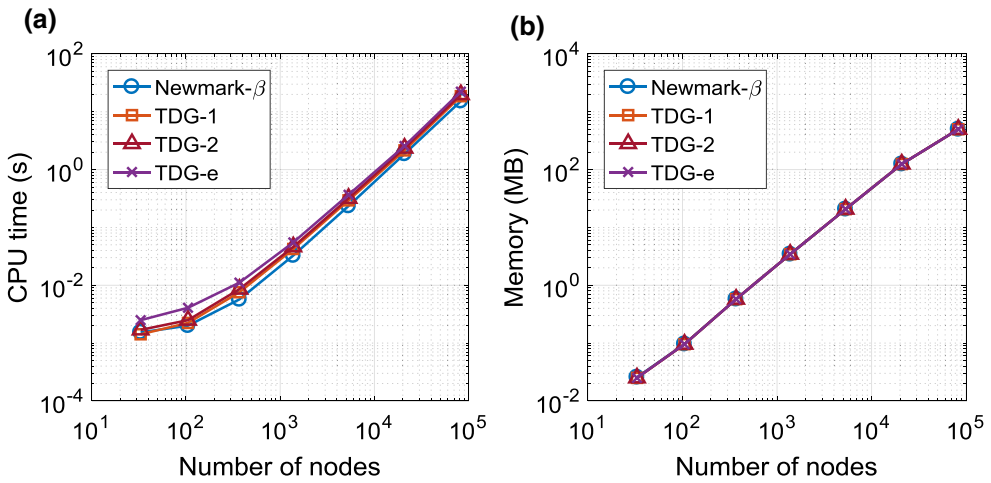

Fig. 9 Comparison of computational performances between Newmark-β method and TDG-based methods: **a** CPU time and **b** memory usages

Table 4 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	1152	1536	2304
2	1.5	80	210	630	1890	2520	3780
3	1.0	360	651	1953	5859	7812	11,718
4	0.75	960	1476	4428	13,284	17,712	26,568
5	0.5	2880	3965	11,895	35,685	47,580	71,370
6	0.4	5625	7296	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

greatly when comparing with the 2D cases. However, the *Kronecker* solver remains a high efficiency with a time complexity of $O(N^{1.7})$ and a storage complexity of $O(N^{1.6})$. 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 Fig. 10.

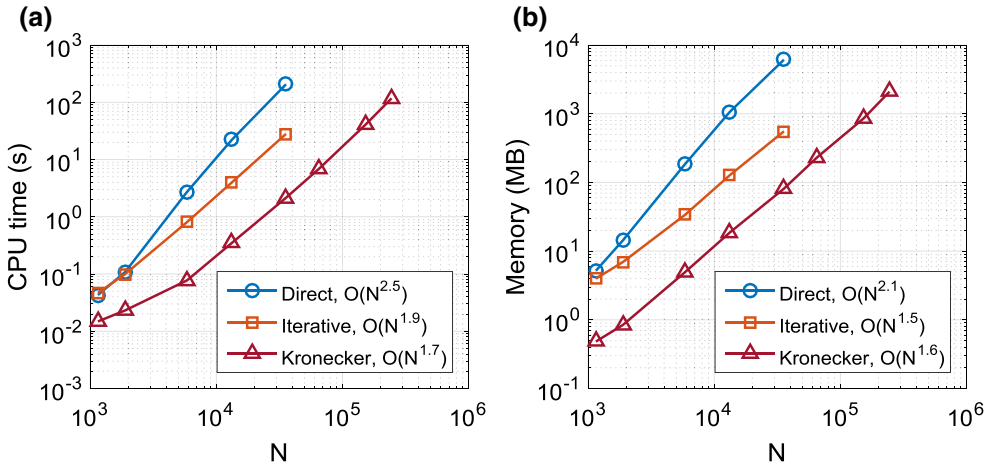

Fig. 10 Comparison of computational performances between different solvers for TDG-1 method: **a** CPU time and **b** memory usages

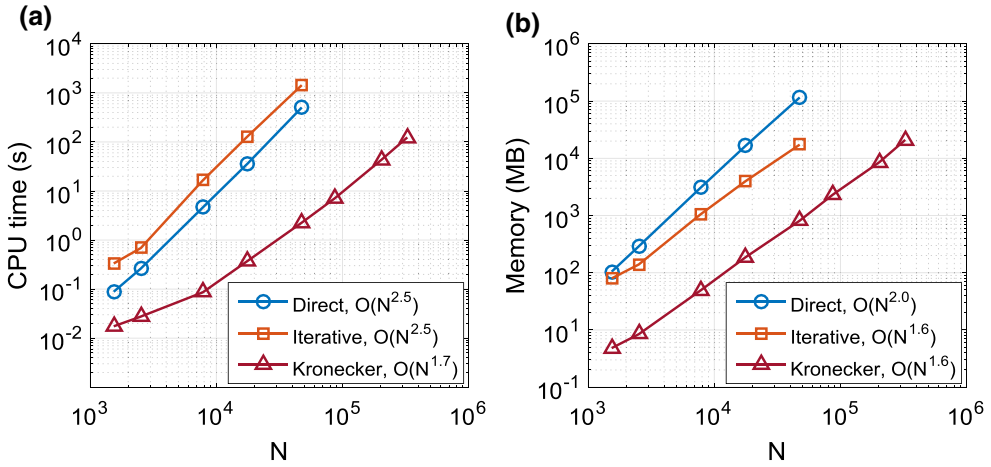

Fig. 11 Comparison of computational performances between different solvers for TDG-2 method: **a** CPU time and **b** memory usages

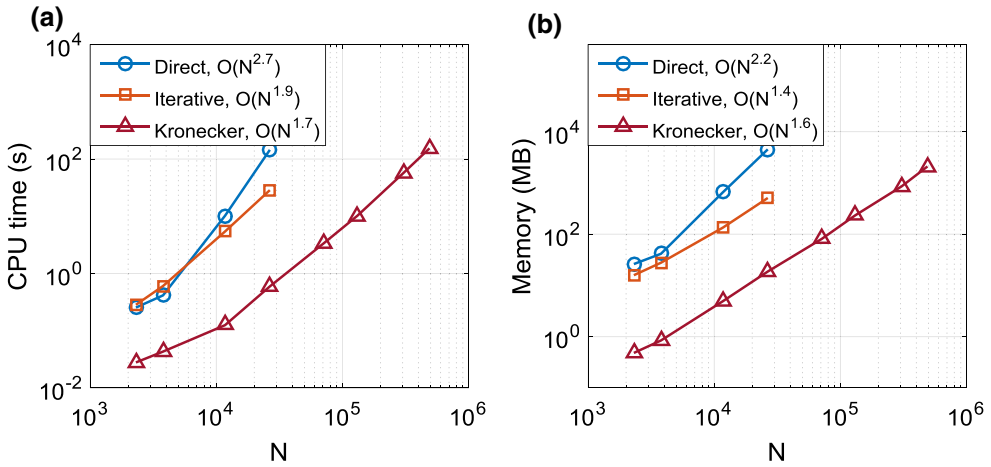

Fig. 12 Comparison of computational performances between different solvers for TDG-e method: **a** CPU time and **b** memory usages

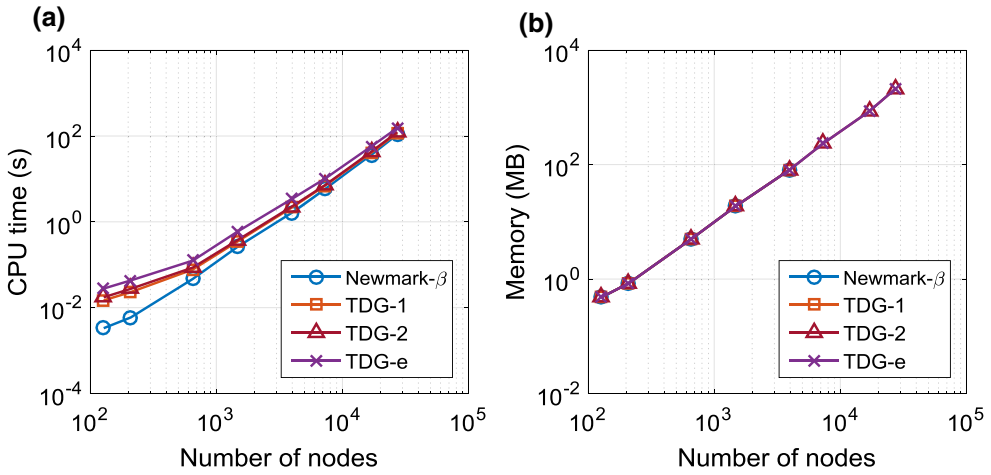

Fig. 13 Comparison of computational performances between Newmark-β method and TDG-based methods: **a** CPU time and **b** memory usages

4.5.2 Two-field formulation

For the two-field formulation in 3D cases, the *Kronecker* solver works significantly better than the other solvers as shown in Fig. 11. It reduces the time complexity from $O(N^{2.5})$ to $O(N^{1.7})$. The computational cost of the *Kronecker* solver is at least two orders of magnitude lower than the others for larger N.

4.5.3 Enriched formulation

Figure 12 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 $O(N^{2.7})$. The *Iterative* solver performs slightly better than the *Direct* solver. The *Kronecker* solver remains the same high efficiency of $O(N^{1.7})$ and its cost is almost 1–2 orders of magnitude lower than the other two for $N > 10^4$.

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 13 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 Sect. 3.4. As we have shown in Sect. 4.2 that the TDG methods generally employ a much larger time step than the semi-discrete schemes due 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 center difference method.

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

5 Conclusions

An efficient iterative solution algorithm has been developed in this work that enables the practical applications of TDG-based space–time FEM. The proposed algorithm features a novel preconditioner by utilizing the special block structure of the coupled space–time stiffness matrices that are expressed in the forms of Kronecker product. The most computationally intensive parts of the iterative solver are then accelerated by using the inverse property of Kronecker product. Based on the results of theoretical analysis and numerical examples, the following conclusions are made:

1. The proposed solution algorithm significantly accelerates the solution to the space–time linear systems of equations. Its numerical efficiency is at least 1–2 orders of magnitude better than the direct sparse solver and general preconditioned iterative solver for relatively large numbers of DOFs (e.g., $N > 10^4$). The computational cost of TDG methods is reduced to the same as the implicit algorithm implementation in the semi-discrete method, however, TDG method performs better since it is capable of larger time step size than the implicit algorithm for accomplishing comparable accuracy.
2. The proposed solution algorithm has been shown to 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 [12, 16, 29], this algorithm does not require partially decoupling of the space–time stiffness matrix. It is applicable

to both weakly and strongly coupled space–time linear systems of equations. Therefore, the proposed algorithm is generally applicable to all the TDG formulations.

With significant improvement in the computational efficiency that is enabled by the solution algorithm presented in this work, space–time FEM is ideal and efficient for simulating 3D and large-scale elastodynamics 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. In terms of computational efficiency, future efforts are directed towards integrating with massively parallel computing techniques.

Acknowledgements The work of R. Zhang is supported by the State Scholarship Fund (China) under Grant # 201406290125 and the Eugene McDermott Graduate Fellowship at The University of Texas at Dallas, which are gratefully acknowledged. The authors also would like to thank the National Science Foundation (Grant # CMMI-1727960) for financial support of this research. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the funding agencies.

References

1. Oden JT (1969) A general theory of finite elements. II. Applications. Int J Numer Methods Eng 1(3):247–259
2. Fried I (1969) Finite-element analysis of time-dependent phenomena. AIAA J 7(6):1170–1173
3. Argyris JH, Scharpf DW (1969) Finite elements in time and space. Nucl Eng Des 10(4):456–464
4. Hughes TJR, Franca LP, Mallet M (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. Comput Methods Appl Mech Eng 63(1):97–112
5. Reed WH, Hill TR (1973) Triangular mesh methods for the neutron transport equation. Los Alamos Scientific Laboratory, Los Alamos
6. Lesaint P, Raviart PA (1974) On a finite element method for solving the neutron transport equation. In: de Boor C (ed) Mathematical aspects of finite elements in partial differential equations. Academic Press, New York, pp 89–123
7. Hughes TJR, Hulbert GM (1988) Space–time finite element methods for elastodynamics: formulations and error estimates. Comput Methods Appl Mech Eng 66(3):339–363
8. Hulbert GM (1992) Time finite element methods for structural dynamics. Int J Numer Methods Eng 33(2):307–331
9. Hulbert GM, Hughes TJR (1990) Space–time finite element methods for second-order hyperbolic equations. Comput Methods Appl Mech Eng 84(3):327–348
10. Hughes TJR, Stewart JR (1996) A space–time formulation for multiscale phenomena. J Comput Appl Math 74(1–2):217–229
11. Li XD, Wiberg NE (1998) Implementation and adaptivity of a space–time finite element method for structural dynamics. Comput Methods Appl Mech Eng 156(1–4):211–229
12. Li XD, Wiberg NE (1996) Structural dynamic analysis by a time-discontinuous Galerkin finite element method. Int J Numer Methods Eng 39(12):2131–2152
13. Delfour M, Hager W, Trochu F (1981) Discontinuous Galerkin methods for ordinary differential equations. Math Comput 36(154):455–473
14. Johnson C (1988) Error estimates and adaptive time-step control for a class of one-step methods for Stiff Ordinary Differential Equations. SIAM J Numer Anal 25(4):908–926
15. Wiberg N-E, Li X (1999) Adaptive finite element procedures for linear and non-linear dynamics. Int J Numer Methods Eng 46(10):1781–1802
16. Chien CC, Wu TY (2000) An improved predictor/multi-corrector algorithm for a time-discontinuous Galerkin finite element method in structural dynamics. Comput Mech 25(5):430–437
17. Chien CC, Yang CS, Tang JH (2003) Three-dimensional transient elastodynamic analysis by a space and time-discontinuous Galerkin finite element method. Finite Elem Anal Des 39(7):561–580
18. Strouboulis T, Babuška I, Copps K (2000) The design and analysis of the generalized finite element method. Comput Methods Appl Mech Eng 181(1–3):43–69
19. Moës N, Dolbow J, Belytschko T (1999) A finite element method for crack growth without remeshing. Int J Numer Methods Eng 46(1):131–150
20. Melenk JM, Babuska I (1996) The partition of unity finite element method: basic theory and applications. Comput Methods Appl Mech Eng 139(1–4):289–314
21. Babuška I, Melenk JM (1997) The partition of unity method. Int J Numer Methods Eng 40(4):727–758
22. Chessa J, Belytschko T (2004) Arbitrary discontinuities in space—time finite elements by level sets and X-FEM. Int J Numer Methods Eng 61(61):2595–2614
23. Qian D, Chirputkar S (2014) Bridging scale simulation of lattice fracture using enriched space–time Finite Element Method. Int J Numer Methods Eng 97(11):819–850
24. Chirputkar S, Qian D (2008) Coupled atomistic/continuum simulation based on extended space–time finite element method. CMES Comput Model Eng Sci 24(2–3):185–202
25. Yang Y et al (2012) Enriched space–time finite element method: a new paradigm for multiscaling from elastodynamics to molecular dynamics. Int J Numer Methods Eng 92(2):115–140
26. Bhamare S et al (2014) A multi-temporal scale approach to high cycle fatigue simulation. Comput Mech 53(2):387–400
27. Zhang R et al (2016) Accelerated multiscale space–time finite element simulation and application to high cycle fatigue life prediction. Comput Mech 58(2):329–349
28. Wada S et al (2018) Simulation-based prediction of cyclic failure in rubbery materials using nonlinear space–time finite element method coupled with continuum damage mechanics. Finite Elem Anal Des 138:21–30
29. Kunthong P, Thompson LL (2005) An efficient solver for the high-order accurate time-discontinuous Galerkin (TDG) method for second-order hyperbolic systems. Finite Elem Anal Des 41(7–8):729–762
30. Saad Y, Schultz MH (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 7(3):856–869
31. Newmark NM (1959) A method of computation for structural dynamics. Proc Am Soc Civ Eng 85:67–94
32. Davis TA (2004) Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. ACM Trans Math Softw 30(2):196–199
33. Chan WM, George A (1980) A linear time implementation of the reverse Cuthill–McKee algorithm. BIT Numer Math 20(1):8–14