J Sci Comput (2013) 57:582–603
DOI 10.1007/s10915-013-9718-8

An Efficient High-Order Time Integration Method for Spectral-Element Discontinuous Galerkin Simulations in Electromagnetics

Misun Min · Paul Fischer

Received: 10 February 2011 / Revised: 5 April 2013 / Accepted: 14 April 2013 /
Published online: 25 June 2013

Abstract We investigate efficient algorithms and a practical implementation of an explicit-type high-order timestepping method based on Krylov subspace approximations, for possible application to large-scale engineering problems in electromagnetics. We consider a semi-discrete form of the Maxwell's equations resulting from a high-order spectral-element discontinuous Galerkin discretization in space whose solution can be expressed analytically by a large matrix exponential of dimension $\kappa \times \kappa$. We project the matrix exponential into a small Krylov subspace by the Arnoldi process based on the modified Gram–Schmidt algorithm and perform a matrix exponential operation with a much smaller matrix of dimension $m \times m$ ($m \ll \kappa$). For computing the matrix exponential, we obtain eigenvalues of the $m \times m$ matrix using available library packages and compute an ordinary exponential function for the eigenvalues. The scheme involves mainly matrix-vector multiplications, and its convergence rate is generally $O(\Delta t^{m-1})$ in time so that it allows taking a larger timestep size as m increases. We demonstrate CPU time reduction compared with results from the five-stage fourth-order Runge–Kutta method for a certain accuracy. We also demonstrate error behaviors for long-time simulations. Case studies are also presented, showing loss of orthogonality that can be recovered by adding a low-cost reorthogonalization technique.

Keywords Exponential time integration · Spectral-element discontinuous Galerkin method · Krylov approximation · Arnoldi process · Matrix exponential

1 Introduction

For many applications arising in electromagnetics, such as designing modern accelerator devices [25,26,29] and advanced nanomaterials [23,24,27,28] that are governed by the

M. Min (✉) · P. Fischer
Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA
e-mail: mmin@mcs.anl.gov

P. Fischer
e-mail: fischer@mcs.anl.gov

Maxwell's equations, realistic simulations often require computing the solutions for long-time propagation distance. For example, in particle accelerator physics applications, because of the orders of magnitude difference in the lengths between the beam and accelerator devices, very long time integrations are necessary to get the total effect of the electromagnetic radiations while the beam is passing through the whole device. For exploring light interaction with advanced nanophotonic materials featured by strongly enhanced surface scattering fields, it is more reliable to get accurate time-averaged energy fields or transmission properties of nanosystems by running simulations over several hundreds of wavelengths of traveling distance.

With the motivation for solving such application problems more efficiently and accurately, we consider a high-order time integration method, especially an exponential time integration method based on Krylov subspace approximation, which can possibly enhance the computational performance as well as improve the solution accuracy. Many studies in the literature on exponential time integration methods have focused on convergence theory, error estimates, efficient algorithms and implementation, and their applications for solving systems of equations. We refer to the review papers [1,2] for details and the history of the exponential time integration methods and also some other papers [4–9]. In [4], a theoretical analysis of some Krylov subspace approximations to the matrix exponential operator was presented with a priori and a posteriori error estimates based on rational approximations for computing the resulting small matrix exponential. In [5,6], Krylov subspace methods were applied to solve large linear systems on supercomputers with preconditionings and parabolic equations with time-varying forcing terms. In [7], convergence analysis and an efficient timestep-size control technique based on the Arnoldi algorithm was shown for integrating large-dimensional linear initial-value problems with source terms. In [8,9], exponential time integration methods were discussed for solving large systems of nonlinear differential equations, reaction-diffusion problems, and a time-dependent Schödinger equation. However, few studies have been done on applying an exponential time integration method for high-order spatial approximations, up to the approximation order $N = 20$ or more, for solving problems in electromagnetics.

In this paper, we consider applying such an exponential time integration method combined with a discontinuous Galerkin approach [11] using a spectral element discretization [34], referred as the spectral element discontinuous Galerkin (SEDG) method [27–30] throughout the paper, in space for solving the Maxwell's equations [31].

We simplify our governing equation by using the Maxwell's equations in free space with no source term as a primary step. We focus on a practical implementation and algorithms for an exponential time integration method based on Krylov subspace approximation. The main idea is to project a large matrix exponential operation onto a small dimension of Krylov subspace by the Arnoldi process [3] and compute the matrix exponential of the resulting Hessenberg matrix in a small dimension. In our implementation, instead of carrying out a matrix exponential based on Padé rational or Chebyshev approximations [4,7] for the resulting Hessenberg matrix, we use eigensolvers from existing library packages [22] and compute an ordinary exponential function for the eigenvalues of the Hessenberg matrix. Other than diagonalizing or computing matrix exponential for a very small-dimensional Hessenberg matrix, the algorithm requires only matrix-vector multiplications with the information of the field values at the current time. For this reason, the method can be easily parallelized, and we consider the method as an explicit-type timestepping method.

High-order spatial approximations are known to be more attractive than the conventional lower-order finite-difference method [31] for long-time integration, because the errors are proportional to the linear growth of the spatial error in time [21]. We discuss an SEDG discretization in space that uses a tensor-product basis of the one-dimensional Lagrange interpolation polynomials with the Gauss-Lobatto-Legendre grids [20]. We consider body-fitted, curvilinear hexahedral element discretizations that allow efficient operator evaluation with memory access costs scaling as $O(n)$ and work scaling as $O(nN)$, where $n = E(N+1)^d$ is the total number of grid points in d dimensions, E is the number of elements, and N is the polynomial approximation order.

For time evolution, there have been other studies on high-order time integration methods using simplectic integration approaches for finite element solutions of the time-dependent Maxwell's equations [13] and Hamilton's equations [15,16]. For example, [13] demonstrates high-order simplectic integration methods in conjunction with a high-order vector finite element method using the Nédeléc basis function [14]. For discontinuous Galerkin type approaches for solving Maxwell's equations, Runge–Kutta (RK) type of timestepping methods have been commonly used [10–12]. Thus, in this paper, we focus on comparing our computational results obtained by our exponential time integration method with those of RK methods. Especially, because of its low storage and larger stability region, we consider the five-stage fourth-order Runge–Kutta scheme [33], simply denoted as RK4 throughout the paper, for comparison. We remain further studies of our SEDG method combined with other time itegration methods for comparison as a future work. Future studies will also include adding divergence-free property, handling source term and absorbing boundary conditions within exponential time integration procedure and perform large scale simulations for real application problems.

In this paper, we begin with describing practical implementation for the Krylov approximation with the Arnoldi process. We demonstrate examples showing loss of orthogonality in the Arnoldi vectors obtained by the modified Gram–Schmidt algorithm [3,17], resulting in nonconvergence in their solutions as the spatial approximation order N increases. We use a reorthogonalization technique [3,19] at low cost that recovers full orthogonality of the Arnoldi vectors and achieves spectral convergence for the solutions up to machine accuracy. We provide convergence studies for time-harmonic solutions in one dimension and waveguide solutions in two and three dimensions, including parallel computations. We demonstrate a high-order convergence rate in space and time, depending on the approximation orders N and m. We examine error behaviors for long-time simulations and investigate maximum allowable timestep sizes as m increases. For the exponential time integration method, maximum allowable timestep sizes can be larger as the Krylov subspace dimension m increases. Although the computational cost increases linearly with increasing order m, the gain from taking larger timestep sizes for larger m and reducing the total number of time steps is much larger, so that one can still achieve cost reduction.

The paper is organized as follows. In Sect. 2, we discuss the Krylov approximation and the Arnoldi algorithm, present our implementation, and apply it to a system of ordinary differential equations. In Sect. 3, we specify a weak formulation of the Maxwell's equations using a discontinuous Galerkin approach and describe spatial discretizations. In Sect. 4 we demonstrate convergence studies for the exponential time integration method and error behaviors for long-time integrations. We demonstrate the efficiency of the exponential time integration method provided with timestep reduction and CPU time comparisons. We give conclusions in Sect. 5.

2 Exponential Time Integration Method

We approximate the matrix exponential operation $e^{A}\bar{q}$ as

$$e^A \bar{q} \approx p_{m-1}(A)\bar{q}, \tag{1}$$

where $A \in R^{\kappa \times \kappa}$, $\bar{q} \in R^{\kappa}$, and p_{m-1} is a polynomial of degree $m-1$. All possible polynomial approximations of degree at most $m-1$ can be represented by the Krylov subspace $K_m(A, \bar{q})$, defined as

$$K_m(A, \bar{q}) = \text{span}\{\bar{q}, A\bar{q}, A^2\bar{q}, \ldots, A^{m-1}\bar{q}\}. \tag{2}$$

The Arnoldi process [3,17] generates an orthonormal matrix $V_m \in R^{\kappa \times m}$ whose columns consist of vectors $\{v_1, \ldots, v_m\}$ that are a basis of the Krylov subspace $K_m(A, \bar{q})$ such that

$$h_{j+1,j} v_{j+1} = A v_j - \sum_{i=1}^{j} h_{ij} v_i \quad \text{for } j = 1, 2, \ldots, m \quad \text{while } h_{j+1,j} \neq 0. \tag{3}$$

The Eq. (3) can be expressed as $V_{m+1}\bar{H} = AV_m$ where $\bar{H} = [h_{ij}] \in R^{(m+1)\times m}$. Defining an upper Hessenberg matrix $H_m = [h_{ij}] \in R^{m \times m}$ with $h_{ij} = v_i^T A v_j$, we have $H_m = V_m^T A V_m$. This leads to an approximation for the matrix exponential in Eq. (1) by

$$e^A \bar{q} \approx V_m e^{H_m} V_m^T \bar{q}. \tag{4}$$

Note that usually $\kappa \gg m$ and we approximate a large matrix exponential calculation e^A for an $\kappa \times \kappa$ matrix A by a lower-dimensional matrix exponential calculation e^{H_m} for an $m \times m$ matrix H_m through a projection onto the Krylov subspace.

Now we describe a practical implementation for computing the right-hand side of Eq. (4) that can be expressed in several forms as

$$V_m e^{H_m} V_m^T \bar{q} = \|\bar{q}\| V_m e^{H_m} V_m^T v_1 = \|\bar{q}\| V_m e^{H_m} e_1 = \|\bar{q}\| V_m X e^{\Lambda_m} X^{-1} e_1, \tag{5}$$

where $v_1 = \bar{q}/\|\bar{q}\|$, $V_m^T v_1 = e_1 = (1, 0, \ldots, 0)^T \in R^m$, and $H_m = X\Lambda_m X^{-1}$ for a diagonalizer X and a diagonal matrix Λ_m ($\|\cdot\|$ is the Euclidean norm). In particular, we address two ways of computing $V_m^T \bar{q} = \|\bar{q}\| V_m^T v_1$:

$$\text{(i)} \qquad V_m^T \bar{q} = \|\bar{q}\| e_1, \tag{6}$$

$$\text{(ii)} \qquad V_m^T \bar{q} = \|\bar{q}\| \tilde{e}_1 = \|\bar{q}\| (v_1^T v_1, v_1^T v_2, \ldots, v_1^T v_m)^T, \tag{7}$$

where (i) is using the theoretical fact based on perfect orthonormality of V_m, namely, $V_m^T V_m = I_m$ for an identity matrix I_m of $m \times m$, and (ii) is using the numerical value $\tilde{e}_1$ for $V_m^T v_1$. Although e_1 is commonly used [3,9], we take the numerical value $\tilde{e}_1$ to get a fully numerical solution. Theoretically, $\tilde{e}_1$ and e_1 should give similar results. However, computed quantities can greatly deviate from their theoretical counterparts. Although the modified Gram–Schmidt Arnoldi algorithm shown in Table 1 is known to be a more reliable orthogonalization procedure than the standard Arnoldi algorithm [3], it can still show numerical difficulty in practice. The orthogonality of V_m can be destroyed by round-off so that the resulting quantity $V_m^T v_1 = \tilde{e}_1$ is not close to e_1. In Sect. 4.1, we demonstrate some examples showing nonconverging solution when using $\tilde{e}_1$, because of the loss of orthogonality in the Arnoldi vectors V_m obtained from the modified Gram–Schmidt Arnoldi algorithm. We ensure full orthogonality for V_m when using $\tilde{e}_1$ in order to guarantee reliable numerical scheme for accurate solutions. We show that a reorthogonalization technique described in Table 1 with only $m(m+1)/2$ additional vector multiplications recovers full orthogonality of V_m and gives converging solutions to a machine accuracy. One might consider the Householder algorithm [3] as an alternative; however, that causes some additional cost in computation.

Table 1 Algorithms for the Arnoldi process based on the modified Gram–Schmidt [3] and reorthogonalization [19] methods

Modified Gram-Schmidt	Reorthogonalization Added
$[H_m, V_m] = \text{arnoldi}\ (A, \bar{q})$	$[\tilde{H}_m, \tilde{V}_m] = \text{arnoldi}\ (A, \bar{q})$
$v_1 = \bar{q}/\|\bar{q}\|$	$\tilde{v}_1 = \bar{q}/\|\bar{q}\|$
`do` $j = 1, ..., m$	`do` $j = 1, ..., m$
$w = Av_j$	$\tilde{w} = A\tilde{v}_j$
`do` $i = 1, ..., j$	`do` $i = 1, ..., j$
$h_{i,j} = (w, v_i)$	$\tilde{h}_{i,j} = (\tilde{w}, \tilde{v}_i)$
$w = w - h_{i,j}v_i$	$\tilde{w} = \tilde{w} - \tilde{h}_{i,j}\tilde{v}_i$
`enddo`	`enddo`
	`do` $i = j : -1 : 1$
(no reorthogonalization)	$\tilde{w} = \tilde{w} - (\tilde{v}_i, \tilde{w})\tilde{v}_i$
	`enddo`
$h_{j+1,j} = \|w\|_2 \quad (if \neq 0)$	$\tilde{h}_{j+1,j} = \|\tilde{w}\|_2 \quad (if \neq 0)$
$v_{j+1} = w/h_{j+1,j}$	$\tilde{v}_{j+1} = \tilde{w}/\tilde{h}_{j+1,j}$
`enddo`	`enddo`

To compute matrix exponential e^{H_m}, one can use Padé and Chebyshev rational approximations, discussed in detail in [4,7]. In our implementation, we compute the eigenvalues of the Hessenberg matrix H_m using available library packages and compute an ordinary exponential function for the eigenvalues. For large-scale computations, we carry out our implementation in Fortran. We consider computing e^{H_m} by diagonalizing $H_m = X_m \Lambda_m X_m^{-1}$ with a diagonalizer X_m and a diagonal matrix $\Lambda_m = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_m\}$, so that it involves computing only an ordinary exponential function e^{λ_k} for each k instead of computing a matrix exponential. Matlab is useful for solving and analyzing small-scale problems with easy implementation. Matlab has a function for computing the eigenvalues Λ_m and the diagonalizer X_m for H_m. We summarize our implementation for Eq. (4) as follows:

1. To compute e^{H_m} using the relation $H_m = X_m e^{\Lambda_m} X_m^{-1}$,
   (a) *In Fortran:* use LAPACK package from Netlib [22].
      i. `call zgeev`: get a diagonalizer X_m and a diagonal matrix Λ_m of H_m such that $H_m X_m = X_m \Lambda_m$.
      ii. `call zgetrf`: get an LU factorized matrix $\tilde{X}_m$ for X_m.
      iii. `call zgetri`: get the inverse matrix $(X_m)^{-1}$ of X_m.
   (b) *In Matlab:* use existing Matlab functions.
      i. $[X_m, \Lambda_m] = \texttt{eig}(H_m)$: get a diagonalizer X_m and a diagonal matrix Λ_m of H_m such that $H_m X_m = X_m \Lambda_m$.
      ii. $[Y] = \texttt{inv}(X_m)$: get the inverse matrix $Y = (X_m)^{-1}$ of X_m.
2. Compute $\tilde{e}_1 = V_m^T v_1 \in R^m$ by setting $\tilde{e}_1 = (v_1^T v_1, v_2^T v_1, \dots, v_m^T v_1)^T$.
3. Compute $V_m e^{H_m} V_m^T \bar{q} = \|\bar{q}\| V_m X_m e^{\Lambda_m} X_m^{-1} \tilde{e}_1 = \|\bar{q}\| V_m (X_m C)$ where $C = \text{diag}\{\beta e^{\lambda_k}\}_{k=1}^m$ for a scalar $\beta = Y(1, :)\tilde{e}_1$.

We apply the Krylov approximation for solving a system of time-dependent linear ordinary differential equations given as

$$\mathbf{q}'(t) = A\mathbf{q}(t), \quad t > 0, \tag{8}$$

Table 2 Exponential time integration based on the Krylov approximation

```
do $\bar{n} = 0, 1, 2, ...,$ # of timesteps
    $\bar{q} = \mathbf{q}^{\bar{n}}$
    $[H_m, V_m] =$ arnoldi $(A, \bar{q})$
    $\tilde{e}_1 = (v_1^T v_1, ..., v_m^T v_1)^T$
    $\bar{q} = \|\bar{q}\| V_m e^{\Delta t H_m} \tilde{e}_1$
    $\mathbf{q}^{\bar{n}+1} = \bar{q}$
enddo
```

whose analytic solution is $\mathbf{q}(t) = e^{At}\mathbf{q}(0)$ with $\mathbf{q}(t) = (q_1(t), q_2(t), \ldots, q_n(t))^T$ and an SEDG spatial discretization operator $A \in R^{\kappa\times\kappa}$. For a given Δt, the solution at $t = (\bar{n}+1)\Delta t$ can be expressed as

$$\mathbf{q}^{\bar{n}+1} = e^{\Delta t A}\mathbf{q}^{\bar{n}}, \tag{9}$$

where $\mathbf{q}^{\bar{n}} = \mathbf{q}(\bar{n}\Delta t)$ for $t = \bar{n}\Delta t$ $(\bar{n} = 0, 1, 2, \ldots)$.

We summarize our exponential time integration scheme in Table 2. For the Arnoldi process, in general one can use the modified Gram–Schmidt algorithm in the first column of Table 1 to obtain the Arnoldi vectors and Hessenberg matrix. When the orthogonality of the Arnoldi vectors breaks down, one can add the reorthogonalization loop as in the second column of Table 1. The error arising from the approximation (4) for $e^{\Delta t A}$ is strictly dependent on the spectral properties of A that can be bounded with respect to Δt [4,9] as follows:

$$\|e^{\Delta t A}\bar{q} - V_m e^{\Delta t H_m} V_m^T \bar{q}\| \le C\Delta t^m, \tag{10}$$

where the constant C is a function of A and m.

3 Spatial Discretization

We consider applying the exponential time integration method to the SEDG scheme in space for solving the Maxwell's equations. In this section we describe a weak formulation using discontinuous Galerkin approach and spectral-element discretizations. Consider the nondimensional form of the source-free Maxwell's equations in free space defined on Ω as

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = 0, \quad \nabla \cdot \mathbf{H} = 0, \quad \nabla \cdot \mathbf{E} = 0, \tag{11}$$

where the field vectors $\mathbf{H} = (H_x, H_y, H_z)^T$ and $\mathbf{E} = (E_x, E_y, E_z)^T$ with

$$\mathbf{q} = \begin{bmatrix} \mathbf{H} \\ \mathbf{E} \end{bmatrix} \quad \text{and} \quad \mathbf{F}(\mathbf{q}) = \begin{bmatrix} F_{\mathbf{H}} \\ F_{\mathbf{E}} \end{bmatrix} = \begin{bmatrix} e_i \times \mathbf{E} \\ -e_i \times \mathbf{H} \end{bmatrix}, \tag{12}$$

where e_i $(i = x, y, z)$ are $e_x = (1, 0, 0)$, $e_y = (0, 1, 0)$, and $e_z = (0, 0, 1)$.

3.1 Discontinuous Galerkin Formulation

We begin by formulating a weak form of the Maxwell's equations defined on Ω with nonoverlapping elements Ω^e such that $\Omega = \cup_{e=1}^{E}\Omega^e$. Define local test functions $\Phi_l = (0, \ldots, \bar{\phi}_l, \ldots, 0)^T$ for $l = 1, \ldots, 6$ where $\bar{\phi}_l$ is a nonzero scalar function at the i-th location, to be defined later. Multiplying the local test functions Φ_l to Eq. (11) by vector multiplication and integrating by parts, we have

$$\int_{\Omega^e} \Phi_l \cdot \frac{\partial \mathbf{q}}{\partial t} d\Omega - \int_{\Omega^e} \mathbf{F}(\mathbf{q}) \cdot \nabla \Phi_l d\Omega = - \int_{\partial\Omega^e} \Phi_l \cdot \left[\mathbf{n} \cdot \mathbf{F}(\mathbf{q})\right] d\Omega, \tag{13}$$

where $\nabla\Phi_l = (0, \dots, \nabla\bar{\phi}_l, \dots, 0)^T$, $\partial\Omega^e$ represents the surface boundary of the element, and $\mathbf{n} = (n_x, n_y, n_z)$ is the unit normal vector pointing outward. In the discontinuous Galerkin approach, we define a numerical flux $\mathbf{F}^*$ that is a function of the local solution $\mathbf{q}$ and the neighboring solution $\mathbf{q}^+$ at the interfaces between neighboring elements. The numerical flux combines the two solutions that are allowed to be different at the interfaces. Replacing $\mathbf{F}(\mathbf{q})$ on the right-hand side of (13) by the numerical flux $\mathbf{F}^*(\mathbf{q})$ as

$$\int_{\Omega^e} \Phi_l \cdot \frac{\partial \mathbf{q}}{\partial t} d\Omega - \int_{\Omega^e} \mathbf{F}(\mathbf{q}) \cdot \nabla \Phi_l d\Omega = - \int_{\partial\Omega^e} \Phi_l \cdot \left[\mathbf{n} \cdot \mathbf{F}^*(\mathbf{q})\right] d\bar{\Omega}, \tag{14}$$

and integrating by parts again, we obtain a weak formulation as

$$\left(\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}), \Phi_l\right)_{\Omega^e} = \left(\mathbf{n} \cdot \left[\mathbf{F}(\mathbf{q}) - \mathbf{F}^*(\mathbf{q})\right], \Phi_l\right)_{\partial\Omega^e}. \tag{15}$$

With a properly chosen numerical flux $\mathbf{F}^*$, either a central or an upwind flux as in [11], we have the integrand for the right-hand side of (15) as

$$\mathbf{n} \cdot (F_{\mathbf{H}} - F^*_{\mathbf{H}}) = 1/2(-\mathbf{n} \times [\mathbf{E}] - \alpha \mathbf{n} \times \mathbf{n} \times [\mathbf{H}]) \tag{16}$$

$$\mathbf{n} \cdot (F_{\mathbf{E}} - F^*_{\mathbf{E}}) = 1/2(\mathbf{n} \times [\mathbf{H}] - \alpha \mathbf{n} \times \mathbf{n} \times [\mathbf{E}]), \tag{17}$$

where $[\mathbf{E}] = \mathbf{E}^+ - \mathbf{E}$ and $[\mathbf{H}] = \mathbf{H}^+ - \mathbf{H}$, and $\alpha = 0$ for the central flux and $\alpha = 1$ for the upwind flux. Boundary conditions are weakly imposed through the surface integration for the flux term. We consider problems with periodic and perfect electric boundary conditions.

3.2 Spectral Element Discretizations

We define a local approximate solution in Ω^e for each component of Eq. (11) that can be written as

$$q^N(x, y, z, t) = \sum_{i,j,k=0}^{N} q^N_{ijk} \psi_{ijk}(\xi, \eta, \gamma) \quad \text{for} \quad (\xi, \eta, \gamma) \in [-1, 1]^3, \tag{18}$$

where $q^N_{ijk} = q^N(x_i, y_j, z_k, t)$ and $\psi_{ijk}(\xi, \eta, \gamma) = l_i(\xi(x))l_j(\eta(y))l_k(\gamma(z))$ using the one-dimensional Lagrange interpolation basis $l_i(\xi)$ based on the Gauss-Lobatto-Legendre quadrature nodes $\{\xi_0, \xi_1, \dots, \xi_N\}$. The Gordon-Hall mapping transforms the physical domain $(x, y, z) \in \Omega^e$ into the reference domain $(\xi, \eta, \gamma) \in [-1, 1]^3$, and all the computations are carried out in the reference domain [20].

For time and spatial derivatives, we have

$$\frac{\partial q^N}{\partial t} = \sum_{i,j,k=0}^{N} \frac{dq^N_{ijk}}{dt} \psi_{ijk}, \qquad \frac{\partial q^N}{\partial x} = \sum_{i,j,k=0}^{N} q^N_{ijk} \frac{\partial \psi_{ijk}}{\partial x}, \tag{19}$$

$$\frac{\partial q^N}{\partial y} = \sum_{i,j,k=0}^{N} q^N_{ijk} \frac{\partial \psi_{ijk}}{\partial y}, \qquad \frac{\partial q^N}{\partial z} = \sum_{i,j,k=0}^{N} q^N_{ijk} \frac{\partial \psi_{ijk}}{\partial z}, \tag{20}$$

where the chain rule gives

$$\frac{\partial\psi_{ijk}}{\partial x} = \frac{\partial\psi_{ijk}}{\partial\xi}\frac{\partial\xi}{\partial x} + \frac{\partial\psi_{ijk}}{\partial\eta}\frac{\partial\eta}{\partial x} + \frac{\partial\psi_{ijk}}{\partial\gamma}\frac{\partial\gamma}{\partial x}, \tag{21}$$

$$\frac{\partial\psi_{ijk}}{\partial y} = \frac{\partial\psi_{ijk}}{\partial\xi}\frac{\partial\xi}{\partial y} + \frac{\partial\psi_{ijk}}{\partial\eta}\frac{\partial\eta}{\partial y} + \frac{\partial\psi_{ijk}}{\partial\gamma}\frac{\partial\gamma}{\partial y}, \tag{22}$$

$$\frac{\partial\psi_{ijk}}{\partial z} = \frac{\partial\psi_{ijk}}{\partial\xi}\frac{\partial\xi}{\partial z} + \frac{\partial\psi_{ijk}}{\partial\eta}\frac{\partial\eta}{\partial z} + \frac{\partial\psi_{ijk}}{\partial\gamma}\frac{\partial\gamma}{\partial z}. \tag{23}$$

We define the Jacobian J for the coordinate transformation as in [24] by

$$J = \begin{vmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial x}{\partial\eta} & \frac{\partial x}{\partial\gamma} \\ \frac{\partial y}{\partial\xi} & \frac{\partial y}{\partial\eta} & \frac{\partial y}{\partial\gamma} \\ \frac{\partial z}{\partial\xi} & \frac{\partial z}{\partial\eta} & \frac{\partial z}{\partial\gamma} \end{vmatrix} \tag{24}$$

from the following relation:

$$\begin{pmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial x}{\partial\eta} & \frac{\partial x}{\partial\gamma} \\ \frac{\partial y}{\partial\xi} & \frac{\partial y}{\partial\eta} & \frac{\partial y}{\partial\gamma} \\ \frac{\partial z}{\partial\xi} & \frac{\partial z}{\partial\eta} & \frac{\partial z}{\partial\gamma} \end{pmatrix} \begin{pmatrix} \frac{\partial\xi}{\partial x} & \frac{\partial\xi}{\partial y} & \frac{\partial\xi}{\partial z} \\ \frac{\partial\eta}{\partial x} & \frac{\partial\eta}{\partial y} & \frac{\partial\eta}{\partial z} \\ \frac{\partial\gamma}{\partial x} & \frac{\partial\gamma}{\partial y} & \frac{\partial\gamma}{\partial z} \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{25}$$

We denote our solution vector as $\mathbf{q^N} = (H_x^N, H_y^N, H_z^N, E_x^N, E_y^N, E_z^N)^T \in R^\kappa$. We express each field component of $\mathbf{q^N}$ in the form of (18), plug them into the weak formulation (15) with the test function Φ_l defined with $\bar{\phi}_l = \psi_{\hat{i}\hat{j}\hat{k}}$ for each l with $\hat{i}, \hat{j}, \hat{k} = 0, 1, \ldots, N$, and apply the Gauss quadrature rule to get the following semidiscrete form:

$$\mathbf{M}\frac{dH_x^N}{dt} = -(\mathbf{D}_y E_z^N - \mathbf{D}_z E_y^N) - \mathbf{R}(\mathbf{H}^N)_x, \tag{26}$$

$$\mathbf{M}\frac{dH_y^N}{dt} = -(\mathbf{D}_z E_x^N - \mathbf{D}_x E_z^N) - \mathbf{R}(\mathbf{H}^N)_y, \tag{27}$$

$$\mathbf{M}\frac{dH_z^N}{dt} = -(\mathbf{D}_x E_y^N - \mathbf{D}_y E_x^N) - \mathbf{R}(\mathbf{H}^N)_z, \tag{28}$$

$$\mathbf{M}\frac{dE_x^N}{dt} = (\mathbf{D}_y H_z^N - \mathbf{D}_z H_y^N) - \mathbf{R}(\mathbf{E}^N)_x, \tag{29}$$

$$\mathbf{M}\frac{dE_y^N}{dt} = (\mathbf{D}_z H_x^N - \mathbf{D}_x H_z^N) - \mathbf{R}(\mathbf{E}^N)_y, \tag{30}$$

$$\mathbf{M}\frac{dE_z^N}{dt} = (\mathbf{D}_x H_y^N - \mathbf{D}_y H_x^N) - \mathbf{R}(\mathbf{E}^N)_z, \tag{31}$$

where mass and stiffness matrices are defined as

$$\mathbf{M} = (\psi_{ijk}, \psi_{\hat{i}\hat{j}\hat{k}})_{\Omega^e}, \quad \mathbf{D}_x = \left(\frac{\partial\psi_{ijk}}{\partial x}, \psi_{\hat{i}\hat{j}\hat{k}}\right)_{\Omega^e}, \tag{32}$$

$$\mathbf{D}_y = \left(\frac{\partial\psi_{ijk}}{\partial y}, \psi_{\hat{i}\hat{j}\hat{k}}\right)_{\Omega^e}, \quad \mathbf{D}_z = \left(\frac{\partial\psi_{ijk}}{\partial z}, \psi_{\hat{i}\hat{j}\hat{k}}\right)_{\Omega^e}, \tag{33}$$

and the surface integration as

$$\mathbf{R}(\mathbf{H}^N) = \left(\mathbf{n}\cdot\left[F_{\mathbf{H}} - F_{\mathbf{H}}^*\right], \phi_{\hat{i}\hat{j}\hat{k}}\right)_{\partial\Omega^e}, \tag{34}$$

$$\mathbf{R}(\mathbf{E}^N) = \left(\mathbf{n}\cdot\left[F_\mathbf{E} - F_\mathbf{E}^*\right], \phi_{\hat{i}\hat{j}\hat{k}}\right)_{\partial\Omega^e}. \tag{35}$$

Applying the Gauss quadrature rule to (32–35), we have

$$\begin{aligned}
(\psi_{ijk}, \psi_{\hat{i}\hat{j}\hat{k}})_{\Omega^e} &= \sum_{l,m,n=0}^{N} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l_i(\xi_l)l_{\hat{j}}(\eta_m)l_j(\eta_m)l_{\hat{k}}(\gamma_n)l_k(\gamma_n)\\
&= J(\hat{M}\otimes\hat{M}\otimes\hat{M}), &(36)\\
\left(\frac{\partial\psi_{ijk}}{\partial x}, \psi_{\hat{i}\hat{j}\hat{k}}\right)_{\Omega^e} &= \sum_{l,m,n=0}^{N} G^{\xi x}_{lmn} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l'_i(\xi_l)l_{\hat{j}}(\eta_m)l_j(\eta_m)l_{\hat{k}}(\gamma_n)l_k(\gamma_n)\\
&\quad + \sum_{l,m,n=0}^{N} G^{\eta x}_{lmn} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l_i(\xi_l)l_{\hat{j}}(\eta_m)l'_j(\eta_m)l_{\hat{k}}(\gamma_n)l_k(\gamma_n)\\
&\quad + \sum_{l,m,n=0}^{N} G^{\gamma x}_{lmn} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l_i(\xi_l)l_{\hat{j}}(\eta_m)l_j(\eta_m)l_{\hat{k}}(\gamma_n)l'_k(\gamma_n)\\
&= (G^{\xi x}JD_\xi + G^{\eta x}JD_\eta + G^{\gamma x}JD_\gamma), &(37)\\
\left(\frac{\partial\psi_{ijk}}{\partial y}, \psi_{\hat{i}\hat{j}\hat{k}}\right)_{\Omega^e} &= \sum_{l,m,n=0}^{N} G^{\xi y}_{lmn} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l'_i(\xi_l)l_{\hat{j}}(\eta_m)l_j(\eta_m)l_{\hat{k}}(\gamma_n)l_k(\gamma_n)\\
&\quad + \sum_{l,m,n=0}^{N} G^{\eta y}_{lmn} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l_i(\xi_l)l_{\hat{j}}(\eta_m)l'_j(\eta_m)l_{\hat{k}}(\gamma_n)l_k(\gamma_n)\\
&\quad + \sum_{l,m,n=0}^{N} G^{\gamma y}_{lmn} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l_i(\xi_l)l_{\hat{j}}(\eta_m)l_j(\eta_m)l_{\hat{k}}(\gamma_n)l'_k(\gamma_n)\\
&= (G^{\xi y}JD_\xi + G^{\eta y}JD_\eta + G^{\gamma y}JD_\gamma), &(38)\\
\left(\frac{\partial\psi_{ijk}}{\partial z}, \psi_{\hat{i}\hat{j}\hat{k}}\right)_{\Omega^e} &= \sum_{l,m,n=0}^{N} G^{\xi z}_{lmn} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l'_i(\xi_l)l_{\hat{j}}(\eta_m)l_j(\eta_m)l_{\hat{k}}(\gamma_n)l_k(\gamma_n)\\
&\quad + \sum_{l,m,n=0}^{N} G^{\eta z}_{lmn} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l_i(\xi_l)l_{\hat{j}}(\eta_m)l'_j(\eta_m)l_{\hat{k}}(\gamma_n)l_k(\gamma_n)\\
&\quad + \sum_{l,m,n=0}^{N} G^{\gamma z}_{lmn} J_{lmn}\rho_{lmn} l_{\hat{i}}(\xi_l)l_i(\xi_l)l_{\hat{j}}(\eta_m)l_j(\eta_m)l'_k(\gamma_n)l_{\hat{k}}(\gamma_n)\\
&= (G^{\xi z}JD_\xi + G^{\eta z}JD_\eta + G^{\gamma z}JD_\gamma), &(39)
\end{aligned}$$

where $\rho_{lmn} = w_l w_m w_n$ using one-dimensional weight w_i, $J = \text{diag}\{J_{lmn}\}$ represents the Jacobian at each node, and $\hat{M}_{\hat{i}i} = \sum_{k=0}^{N} l_i(\xi_k)l_{\hat{i}}(\xi_k)w_k = \text{diag}\{w_i\}$ is the mass matrix associated with the one-dimensional reference domain $[-1, 1]$. The stiffness matrices can be represented by using tensor product forms of

$$D_\xi = \hat{M}\otimes\hat{M}\otimes\hat{M}\hat{D},\quad D_\eta = \hat{M}\otimes\hat{M}\hat{D}\otimes\hat{M},\quad D_\gamma = \hat{M}\hat{D}\otimes\hat{M}\otimes\hat{M}, \tag{40}$$

where the one-dimensional differentiation matrix is defined by $\hat{D}_{ji} = l'_i(\xi_j)$. The geometric factors $G^{\xi x} = \partial\xi/\partial x = \text{diag}\{G^{\xi x}_{lmn}\}$, $G^{\eta y} = \partial\eta/\partial y = \text{diag}\{G^{\eta y}_{lmn}\}$, and $G^{\gamma z} = \partial\gamma/\partial z =$

diag$\{G^{\gamma z}_{lmn}\}$ represent their values at each node $(\xi_l, \eta_m, \gamma_n)$, and similarly for $G^{\xi y}$, $G^{\xi z}$, $G^{\eta x}$, $G^{\eta z}$, $G^{\gamma x}$, and $G^{\gamma y}$. The two-dimensional surface integrations in Eqs. (34–35) are written as

$$\mathbf{R}(\mathbf{H}^N) = \sum_{f=1}^{6}\sum_{s=1}^{N_{2d}} \frac{1}{2}(-\mathbf{n} \times \mathcal{R}^f_s\{[\mathbf{E}^N_{ijk}]\} - \mathbf{n} \times \mathbf{n} \times \mathcal{R}^f_s\{[\mathbf{H}^N_{ijk}]\})w_s J^f_s, \tag{41}$$

$$\mathbf{R}(\mathbf{E}^N) = \sum_{f=1}^{6}\sum_{s=1}^{N_{2d}} \frac{1}{2}(\mathbf{n} \times \mathcal{R}^f_s\{[\mathbf{H}^N_{ijk}]\} - \mathbf{n} \times \mathbf{n} \times \mathcal{R}^f_s\{[\mathbf{E}^N_{ijk}]\})w_s J^f_s, \tag{42}$$

where $R^f_s\{\cdot\}$ extracts the information of $\{\cdot\}$ at the nodes situated on each face of the local element for the face number f; w_s is the weight on the surface, J^f_s is the surface Jacobian at the nodes on each face, and $N_{2d} = (N+1)^2$. To define the unit normal vector $\mathbf{n}$ corresponding to the face in the reference element with respect to ξ, η, and γ (i.e., $\mathbf{n}_{\xi\eta}$, $\mathbf{n}_{\eta\gamma}$, and $\mathbf{n}_{\gamma\xi}$, respectively), we consider the infinitesimal displacement $\mathbf{x} = (x, y, z)$ on the tangential plane along the boundary $\partial\Omega^e$, which can be written as $\epsilon_\xi = \frac{\partial \mathbf{x}}{\partial \xi}d\xi$, $\epsilon_\eta = \frac{\partial \mathbf{x}}{\partial \eta}d\eta$, and $\epsilon_\gamma = \frac{\partial \mathbf{x}}{\partial \gamma}d\gamma$. Then, the normal vectors are defined as

$$\mathbf{n}_{\xi\eta} = \frac{1}{J_{\xi\eta}}\left(\frac{\partial \mathbf{x}}{\partial \xi} \times \frac{\partial \mathbf{x}}{\partial \eta}\right), \mathbf{n}_{\eta\gamma} = \frac{1}{J_{\eta\gamma}}\left(\frac{\partial \mathbf{x}}{\partial \eta} \times \frac{\partial \mathbf{x}}{\partial \gamma}\right), \mathbf{n}_{\gamma\xi} = \frac{1}{J_{\gamma\xi}}\left(\frac{\partial \mathbf{x}}{\partial \gamma} \times \frac{\partial \mathbf{x}}{\partial \xi}\right),$$

where the surface Jacobians are defined for J^f_s as

$$J_{\xi\eta} = \left\|\frac{\partial \mathbf{x}}{\partial \xi} \times \frac{\partial \mathbf{x}}{\partial \eta}\right\|, J_{\eta\gamma} = \left\|\frac{\partial \mathbf{x}}{\partial \eta} \times \frac{\partial \mathbf{x}}{\partial \gamma}\right\|, J_{\gamma\xi} = \left\|\frac{\partial \mathbf{x}}{\partial \gamma} \times \frac{\partial \mathbf{x}}{\partial \xi}\right\|. \tag{43}$$

Finally, we can express the semidiscrete scheme of Eqs. (26–31) in matrix form as

$$\frac{d\mathbf{q}^{\mathbf{N}}}{dt} = A\mathbf{q}^{\mathbf{N}}, \tag{44}$$

where $A = (\bar{M})^{-1}\bar{A} \in R^{\kappa\times\kappa}$ with $\bar{A} = \bar{D} - \bar{R}$, $\kappa = 3n$ for $n = E(N+1)^2$ in two dimensions and $\kappa = 6n$ for $n = E(N+1)^3$ in three dimensions. The mass matrix can be written as

$$\bar{M} = \text{diag}\{M, M, M, M, M, M\}, \tag{45}$$

which is fully diagonal so that mass matrix inversion $(\bar{M})^{-1}$ gives also a fully diagonal matrix. The stiffness matrix can be written as

$$\bar{D} = \begin{bmatrix} 0 & 0 & 0 & 0 & D_z & -D_y \\ 0 & 0 & 0 & -D_z & 0 & D_x \\ 0 & 0 & 0 & D_y & -D_x & 0 \\ 0 & -D_z & D_y & 0 & 0 & 0 \\ D_z & 0 & -D_x & 0 & 0 & 0 \\ -D_y & D_x & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{46}$$

and $\bar{R}$ is the surface integration acting on the boundary face of the local element obtained from Eqs. (41–42).

3.3 Spatial Operator and Stability

We examine the structures and eigenvalue spectra of the SEDG spatial operator $\bar{A} = \bar{M}A$ from Eq. (44) for the cases of the central and upwind fluxes. Figures 1, 2 demonstrate the patterns of the structures and eigenvalue distributions of the two- and three-dimenional

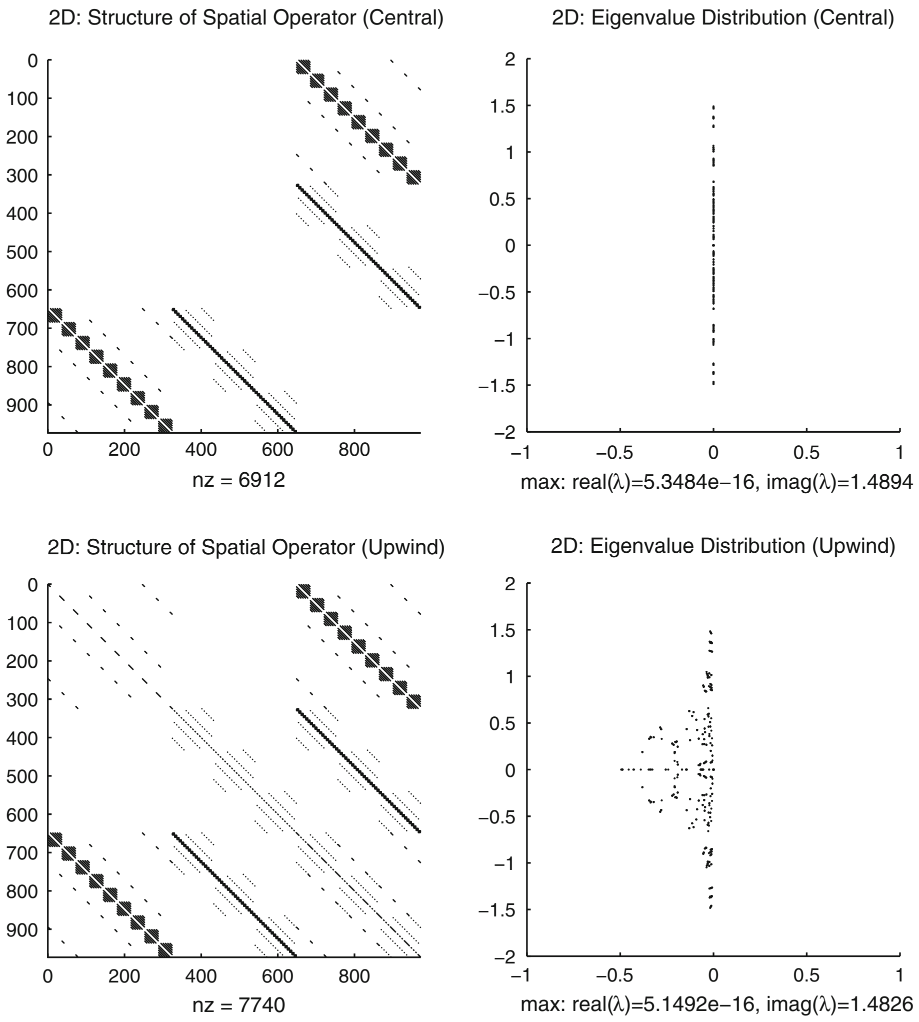

Fig. 1 Structures of the SEDG spatial operators $\bar{A}$ and eigenvalue distributions for the central and upwind fluxes in 2D with $E = 3 \times 3$ and $N = 5$, $\kappa = 3E(N+1)^2$. `nz` is the number of nonzeros in $A \in R^{\kappa \times \kappa}$

problems with periodic boundary conditions using relatively small E and N for simplicity. The dimension of $\bar{A}$ is $\kappa \times \kappa$ for $\kappa = 3n$ with $n = E(N+1)^2 = 324$ in two dimensions and for $\kappa = 6n$ with $n = E(N+1)^3 = 729$ in three dimensions. The eigenvalues λ for the central flux reside on the imaginary axis and those for the upwind flux on the negative half-plane. The RK4 (5-stage) [33] timestepping method has been considered for this type of spatial discretizations in [10,11] due to its low storage and larger stability region.

In this paper, we consider an exponential time integration approach. The solution of Eq. (44) can be expressed by Eq. (9) with $A = (\bar{M})^{-1}\bar{A}$ that has the similar patterns for the structure and eigenvalue distribution as those of $\bar{A}$ because $(\bar{M})^{-1}$ is a fully diagonal matrix. Applying the Arnoldi algorithm at each timestep, we obtain the upper Hessenberg matrix H_m and Arnoldi vectors that satisfy $H_m = V_m^T A V_m$ and Eq. (4). Defining the logarithmic norm μ for a square matix as in [18], the following condition [7] holds

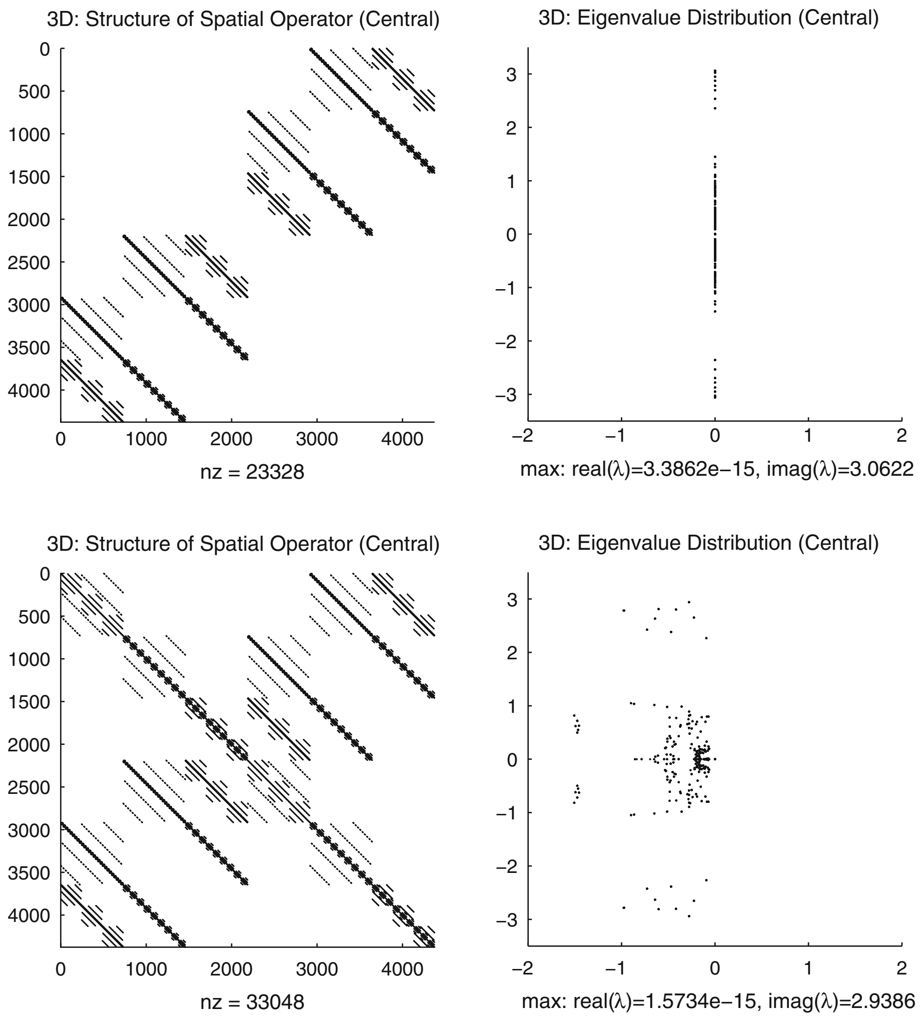

Fig. 2 Structures of the SEDG spatial operators $\bar{A}$ and eigenvalue distributions for the central and upwind fluxes in three dimensions with $E = 3 \times 3 \times 3$ and $N = 2$, $\kappa = 6E(N+1)^3$. `nz` is the number of nonzeros in $A \in R^{\kappa \times \kappa}$

$$\| V_m e^{\Delta t H_m} V_m^T \|_2 \le \| e^{\Delta t H_m} \|_2 \le e^{\mu(\Delta t H_m)} \le e^{\mu(\Delta t A)} \le 1, \tag{47}$$

if the eigenvalues of the spatial operator A are in the negative half-plane. This implies that the exponential time integration scheme is suitable for our SEDG spatial approximations to ensure the stability.

4 Computational Results

This section presents computational results of the exponential time integration method with our SEDG approximation (often denoted by EXP throughout this paper) for simulating a periodic solution in 1D and wave guide solutions in 2D and 3D [32], defined as follows:

Example 1 One-dimensional periodic solution:

$$H_y = -\sin kx \sin wt, \quad E_z = \cos kx \cos wt \quad \text{on} \quad \Omega = [-\pi, \pi], \tag{48}$$

where k and w are integers with $k = w$.

Example 2 Two-dimensional waveguide solution:

$$\begin{aligned} H_x &= 2(k_y/w)\sin(k_y y)\sin(k_x x + wt), \\ H_y &= 2(k_x/w)\cos(k_y y)\cos(k_x x + wt), \\ E_z &= 2\cos(k_y y)\cos(k_x x + wt), \end{aligned} \tag{49}$$

where $k_x = 2\pi$, $k_y = \pi$, and $w = \sqrt{k_x^2 + k_y^2}$ for $\Omega = [-0.5, 0.5]^2$. The solution represents the periodic boundary in x and PEC boundary in y.

Example 3 Three-dimensional waveguide solution:

$$\begin{aligned} H_x &= -k_y w\pi\gamma^{-2}\sin(k_x\pi x)\cos(k_y\pi y)\sin(wt - k_z z), \\ H_y &= k_x w\pi\gamma^{-2}\cos(k_x\pi x)\sin(k_y\pi y)\sin(wt - k_z z), \\ H_z &= 0, \\ E_x &= k_x k_z\pi\gamma^{-2}\cos(k_x\pi x)\sin(k_y\pi y)\sin(wt - k_z z), \\ E_y &= k_y k_z\pi\gamma^{-2}\sin(k_x\pi x)\cos(k_y\pi y)\sin(wt - k_z z), \\ E_z &= \sin(k_x\pi x)\sin(k_y\pi y)\cos(wt - k_z z), \end{aligned} \tag{50}$$

where $w = \sqrt{k_z^2 + \gamma^2}$ and $\gamma = \pi\sqrt{k_x^2 + k_y^2}$ on $\Omega = [0, 1]^2 \times [0, 2\pi]$. The solution represents the PEC boundary in x and y and periodic boundary in z.

4.1 Cases on Loss of Orthogonality for V_m

A practical implementation for computing e^{H_m} and $V_m e^{H_m} V_m^T q$ was addressed in Sect. 2. Here we focus on case studies showing nonconvergence behaviors of computing $V_m e^{H_m} V_m^T q$ by using the numerical quantity $\tilde{e}_1 = V_m^T v_1$ based on the modified Gram–Schmidt algorithm. Consider the one- and two-dimensional solutions defined in Eqs. (48) and (49). We investigate the closeness of $V_{m+1}^T V_{m+1}$ to an identity matrix I_{m+1} in a matrix norm $\|K\|_1 = \max_{1\le j\le m+1}\sum_{i=1}^{m+1}|k_{ij}|$ for $K = [k_{ij}]$. In Fig. 3, we demonstrate the orthogonality of V_{m+1} for varying $N = 3, 4, 5, \ldots, 24$. We consider $m = 3, 5, 7$ with $E = 3$ in one dimension, and $m = 3, 7, 11$ with $E = 3^2$ in two dimensions. We observe that orthogonality breaks down severely as the spatial approximation order N increases for both 1D and 2D implementations in Matlab and Fortran, respectively. In Table 3, we demonstrate each component of the matrix $\tilde{I} = V_{m+1}^T V_{m+1}$ depending on $N = 5, 10, 15$ for $m = 3$ for the one-dimensional example (48) with $k = 1$. It shows that V_{m+1} rapidly loses orthogonality as N increases. For the case of Table 3, the analytic solution (48) can be expressed by $q = c_1 z_1 + c_2 z_2 + \cdots + c_m z_m$ with $z_1 = (\sin x \sin t, 0)^T$ and $z_2 = (0, \cos x \cos t)^T$, $z_3 = \cdots = z_m = 0$. If the orthognalization algorithm is not good, the algorithm does not provide good Arnoldi vectors that are orthogonal to the previously computed Arnoldi vectors after two iterations during Arnoldi procedure. Table 3 shows nonzero values for $\tilde{I}(3,1)$, $\tilde{I}(4,1)$, $\tilde{I}(1,3)$, $\tilde{I}(1,4)$ as N increases, meaning that v_1 and v_3 are not orthogonal; the same is true for v_1 and v_4.

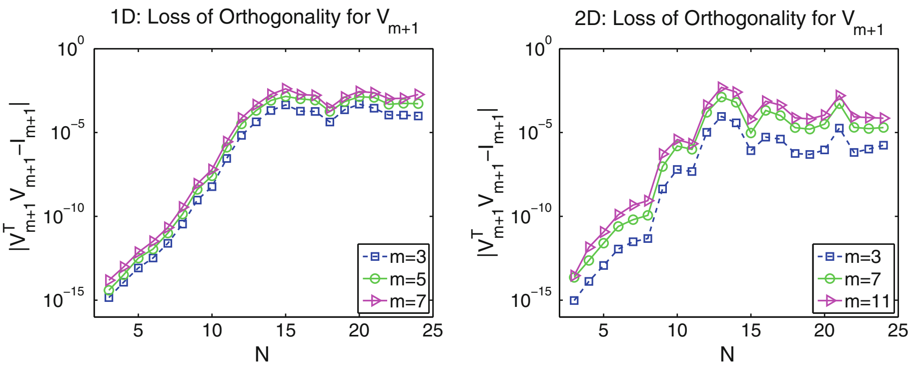

Fig. 3 $\|V_{m+1}^T V_{m+1} - I_{m+1}\|_1$ as a function of N and loss of orthogonality for V_{m+1}: 1D Matlab implementation with $m = 3, 5, 7$ and $E = 3$ (*left*) and 2D Fortran implementation with $m = 3, 7, 11$ and $E = 3^2$ (*right*)

Table 3 Loss of orthogonality of V_m by showing each component of the matrix $\tilde{I} = [\tilde{I}_{ij}] = V_{m+1}^T V_{m+1} \in R^{(m+1)\times(m+1)}$ for $m = 3$ with $N = 5, 10, 15$, considering the solution in Eq. (48) with $k = 1$

$\tilde{I} = V_m^T V_m$ by modified Gram–Schmidt Arnoldi algorithm

Order	$\tilde{I}(i, j)$	$\tilde{I}(:, 1)$	$\tilde{I}(:, 2)$	$\tilde{I}(:, 3)$	$\tilde{I}(:, 4)$
$N = 5$	$\tilde{I}(1, :)$	1.00e+00	0	2.74e−14	5.60e−14
	$\tilde{I}(2, :)$	0	1.00e+00	6.77e−20	4.42e−17
	$\tilde{I}(3, :)$	2.74e−14	6.77e−20	1.00e+00	7.31e−16
	$\tilde{I}(4, :)$	5.60e−14	4.42e−17	7.31e−16	1.00e+00
$N = 10$	$\tilde{I}(1, :)$	1.00e+00	0	-1.91e−09*	−4.02e−09*
	$\tilde{I}(2, :)$	0	1.00e+00	−2.77e−17	−1.12e−17
	$\tilde{I}(3, :)$	−1.91e−09*	−2.77e−17	1.00e+00	6.79e−16
	$\tilde{I}(4, :)$	−4.02e−09*	−1.12e−17	6.79e−16	1.00e+00
$N = 15$	$\tilde{I}(1, :)$	1.00e+00	0	1.31e−04*	3.12e−04*
	$\tilde{I}(2, :)$	0	1.00e+00	−1.04e−17	−3.20e−17
	$\tilde{I}(3, :)$	1.31e−04*	−1.04e−17	1.00e+00	−6.66e−16
	$\tilde{I}(4, :)$	3.12e−04*	−3.20e−17	−6.66e−16	1.00e+00

We examine convergence behaviors of the solution (48) for $N = 5, 10, 15, 20$ after 100 timesteps with $\Delta t = 0.001$, where Δt is small enough not to influence the spatial errors. Table 4 shows that the scheme does not converge further as N increases, because of the loss of orthogonality in the Arnoldi vectors as shown in Fig. 3, especially for $m \geq 3$. For $m = 2$, however, the modified Gram–Schmidt algorithm gives reasonable orthogonal Arnoldi vectors for the first two iterations in the Arnodi process and stops the iteration. Hence, spectral convergence can be observed in Table 4 for $m = 2$. For $m \geq 3$, we can recover full orthogonality and obtain converging solution by adding a reorthogonalization technique to the modified Gram–Schmidt algorithm as in Table 1; the results are shown in Table 5 for $m = 5$.

Table 4 Spatial convergence for Eq. (48) using the modified Gram–Schmidt algorithm with $m = 2, 3, 4$ and $N = 5, 10, 15, 20$ for $E = 3$ after 100 timesteps with $\Delta t = 0.001$

Order	$m = 2$	$m = 3$	$m = 4$
$N = 5$	2.3201e−04	2.3328e−04	2.3328e−04
$N = 10$	2.2220e−09	1.6375e−09	5.7495e−09
$N = 15$	8.3266e−15	2.3154e−05	9.8542e−05
$N = 20$	7.5495e−15	7.2517e−06	8.7112e−06

Table 5 Spatial convergence for Eq. (48) using the modified Gram–Schmidt with reorthogonalization algorithm for $m = 5$, $E = 3$ with $N = 5, 10, 15, 20$ after 100 timesteps with $\Delta t = 0.001$

Order	$m = 5$
$N = 5$	4.2691e−05
$N = 10$	1.1471e−10
$N = 15$	1.0935e−14
$N = 20$	1.0377e−14

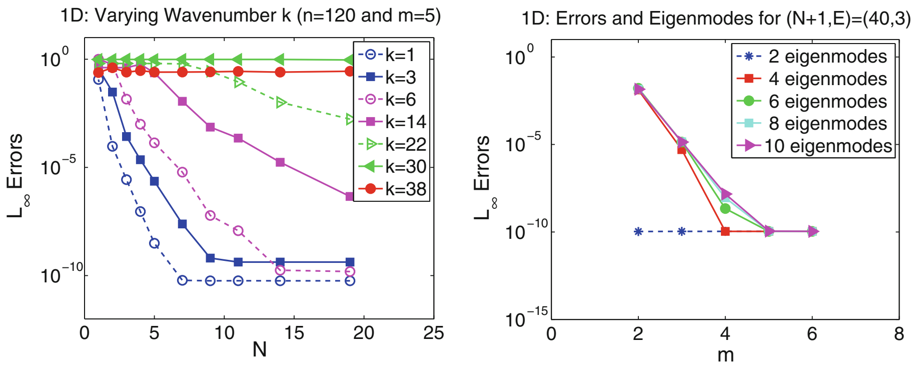

Fig. 4 Errors depending on point per wavelength ($ppw = n/k$) for varying wavenumber k with $n = 120$ at time $t = 100$ with $\Delta t = 0.0005$ (*left*). Errors depending on the Krylov dimension $m = 2, 3, \dots, 6$ for solutions with multiple eigenmodes (*right*)

4.2 Convergence and Eigenmodes

In this section, we first investigate the error behaviors depending on points per wavelength, which can indicate how many grids points per wavelength and what approximation order N are required for a desired level of accuracy. We consider the one-dimensional solution (48) of varying wavenumber k propagating the domain 15.9 times. We fix the resolution with a total number of grid points $n = E(N+1) = 120$ but with varying $N = 1, 2, 3, 4, 5, 7, 9, 11, 14, 19$. In Fig. 4, the left panel shows that, for a fixed Krylov subspace dimension $m = 5$ with $\Delta t = 0.0005$, the error drops rapidly with increasing N for a large number of points per wavelength ($ppw = n/k$), but accurate propagation for $ppw < 8$ requires $N > 8$.

In order to represent solution almost exactly by a linear combination of the orthogonal basis of the Krylov subspace of dimension m, one can choose the approximation order m greater than the number of eigenmodes in the solution. Here we examine error behaviors depending on m for the solution including multiple modes, which is defined by

$$H_y = -\sum_{\bar{k}=6}^{6-\bar{k}_0} \sin \bar{k}x \sin wt \quad \text{and} \quad E_z = \sum_{\bar{k}=6}^{6-\bar{k}_0} \cos \bar{k}x \cos wt, \tag{51}$$

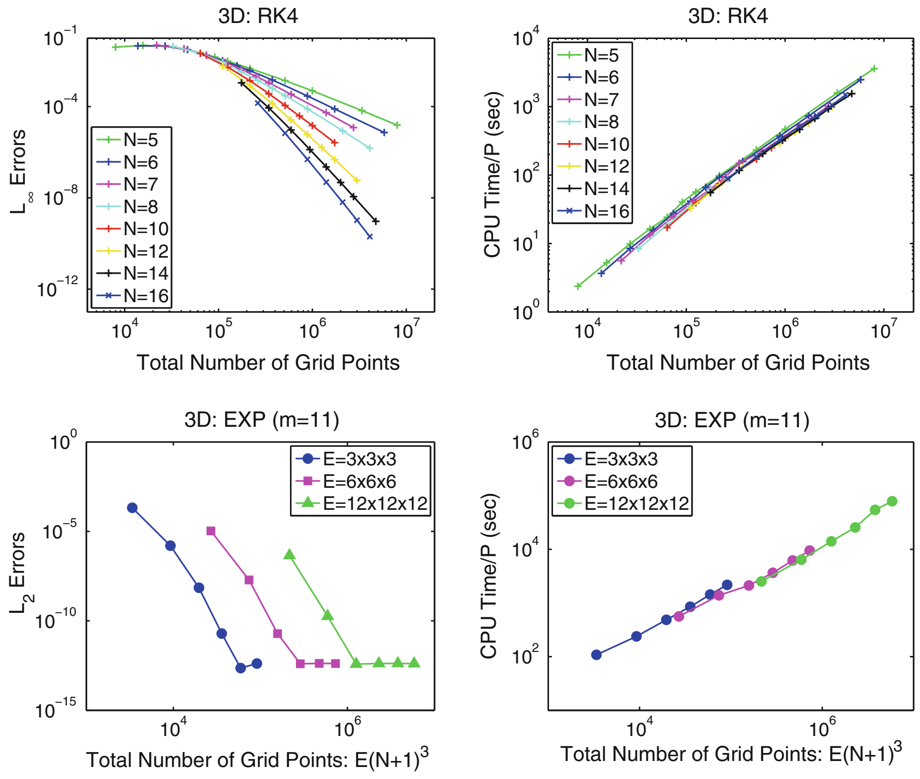

Fig. 5 *Top* (RK4): spatial convergence (*left*) and CPU time per core (right) after 1,000 timesteps on 32 cores of Linux clusters with $E = 4^3$–16^3 and N=5–16 for a periodic solution. *Bottom* (EXP, $m = 11$): spatial convergence (*left*) and CPU time per core (*right*) after 10,000 timesteps on the number of cores $P = 2^4, 2^7, 2^{10}$ on Argonne Blue Gene/P with $E = 3^3, 6^3, 12^3$, respectively, and N=4–14

where $\bar{k}_0 = 0, 1, \ldots, 4$. Equation (51) is represented by $2(\bar{k}_0 + 1)$ eigensolutions. In Fig. 4, the right panel shows that, for a single mode $\bar{k} = 6$ by setting $\bar{k}_0 = 0$, Krylov subspace dimension $m = 2$ is enough to get an accurate solution. For the solution represented by multimode eigensolutions, we also observe that the errors already reach to the limit 10^{-10}, which is dominated by the spatial resolution $n = 120$ for $E = 3$, when having $m \geq 2(\bar{k}_0 + 1)$ for $\bar{k}_0 = 1, 2, 3, 4$. Thus one can expect the best approximate solution in time when the Krylov subspace dimension m is larger than the number of the eigensolutions. This imples that, as an extreme example, a Gaussian pulse represented by 20 modes can be represented almost exactly in time by the Krylov subspace approximation of dimension $m = 40$.

4.3 Convergence in Space and Time

This section demonstrates convergence in space and time for the exponential time integration method applied to our SEDG method in higher dimensions. We also include results from parallel computations. No additional parallel implementation is required for the EXP scheme other than the flux communication between neighboring elements in the spatial operator.

Figure 5 shows spatial convergence for different problem sizes with varying approximation order N for RK4 and EXP with $m = 11$. For RK4, simulations are carried out for a

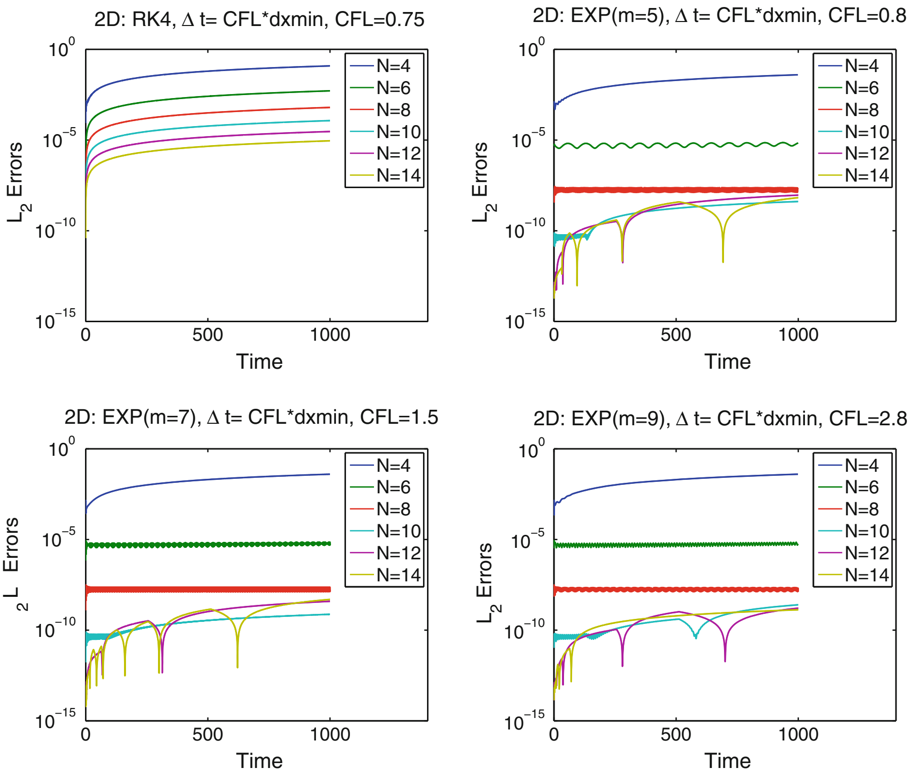

Fig. 6 Long-time integrations for 2D waveguide simulations on $\Omega = [-0.5, 0.5]^2$. Traveling distance is 666.66 wavelengths at time $t = 1,000$. Error behaviors in time for RK4 and EXP(m) with $m = 5, 7, 9$ and $N = 4, 6, 8, 10, 12, 14$ for a fixed $E = 3^2$

three-dimensional periodic solution with $N = 5$–16 and $E = 4^3$–16^3 on 32 cores of Linux clusters at Argonne. For EXP, simulations are performed for a waveguide solution with $N = 4, 6, 8, 10, 12, 14$ and $E = 3^2, 6^2, 12^2$ on $P = 2^4, 2^7, 2^{10}$ cores on the Argonne BG/P. The figures on the left show exponential convergence as N increases. We observe that for a fixed resolution, the accuracy is better with a larger N.

It is equally important that high-order methods be competitive in terms of computational costs. We demonstrate the CPU time per core for 1,000 and 10,000 timesteps for RK4 and EXP, respectively. We observe that the CPU time per core increases linearly depending on the total number of grid points $n = E(N + 1)^3$, but not solely depending on the approximation order N. This ensures that higher-order approximation N is not a source of increasing computational cost in space. We also note that a larger N generally affords less resolution for the same accuracy, particularly suitable for long-time integrations.

Figures 6, 7 demonstrate error behaviors in time and space for long-time integration with traveling distance of more than 666 and 238 wavelengths in 2D and 3D, respectively, for the monochromatic wave solutions in Eqs. (49–50). We consider the EXP scheme for $m = 5, 7, 9$ with a maximum allowable timestep size for each m and examine convergence for $N = 4, 6, 8, 10, 12, 14$ and $E = 3^2$ in 2D and $E = 3^3$ in 3D. We choose a timestep size $\Delta t = \text{CFL*dxmin}$ by defining $\text{CFL} = \frac{c\Delta t}{\text{dxmin}}$ with $c = 1$ and $\text{dxmin} = \min_{N,E}\{\Delta\}$, where $\Delta = \frac{1}{2}\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$. We find the CFL number numerically that gives the

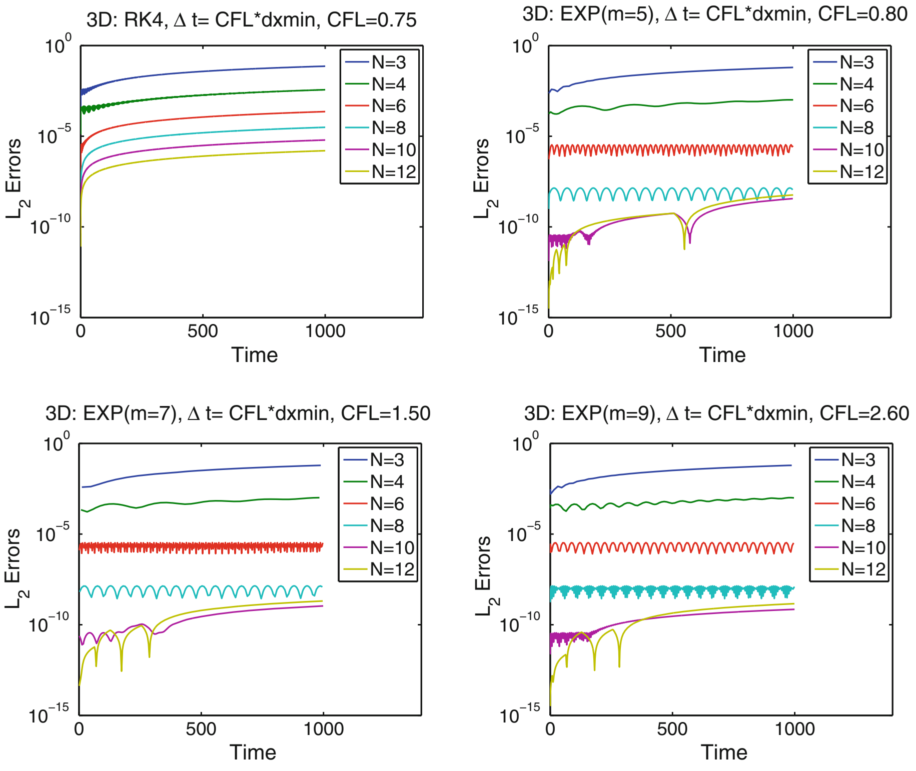

Fig. 7 Long-time integrations for 3D rectangular waveguide simulations on $\Omega = [-0.5, 0.5]^2 \times [0, 2\pi]$. Traveling distance is 238.73 wavelengths at time $t = 1,000$. Error behaviors in time for RK4 and EXP(m) with $m = 5, 7, 9$ and $N = 3, 4, 6, 8, 10, 12$ for a fixed $E = 3^3$

maximum allowable Δt for a stable solution. For comparison, we carried out the same simulations with RK4 (5-stage). For RK4, we use CFL$\approx$0.75. Although our EXP scheme is expected to have bounded solutions because of the A-stable property, the timestep size has to be reasonably small to get accurate solutions. For the EXP scheme, the maximum allowable timestep increases as m increases. We use CFL$\approx$0.8,1.5,2.8 for $m = 5, 7, 9$ in 2D and CFL$\approx$0.8,1.5,2.6 for $m = 5, 7, 9$ in 3D. According to the theoretical studies showing convergence rate of $O(\Delta t^{m-1})$ for the EXP scheme [4,7], we consider EXP(m=5) as the fourth-order scheme that can be compared to RK4. We observe that the CFL numbers are very close to each other for EXP(m=5) and RK4, but the EXP scheme shows superconvergence for the monochromatic wave solutions, with several orders of magnitude difference as N increases.

4.4 Computational Costs

This section demonstrates convergence rate depending on the timestep size and the computational cost depending on m, provided with comparisons between RK4 and EXP.

Figure 8 shows convergence in time with respect to CFL/m for EXP and CFL/5 for RK4, based on the same cost (recall that the 5-stage RK4 involves five times the spatial operation per timestep and EXP requires m times the spatial operation per timestep, but neglecting vector-vector multiplications and additions in the Arnoldi process). For a monochromatic

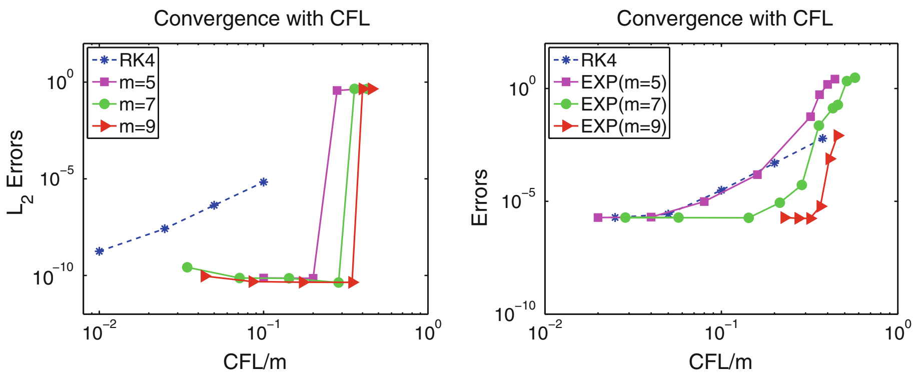

Fig. 8 Convergence in time for variable CFL numbers for 2D waveguide simulations with $E = 4^2$ and $N = 10$ at time $t = 100$ for a traveling distance of 66.66 wavelengths. Error comparison for RK4 and EXP with $m = 5, 7, 9, 11$ for a monochromatic solution (*left*) and a solution represented by 25 different wavemodes (*right*)

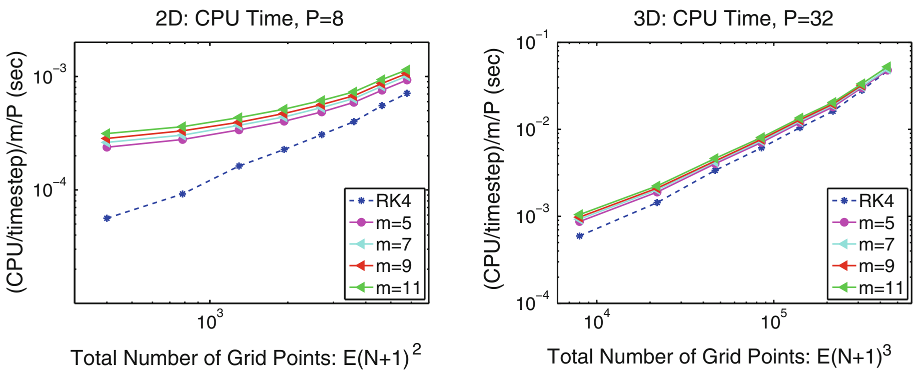

Fig. 9 CPU time: comparison between RK4 and EXP with $E = 4^2$, $P = 8$ in 2D (*left*) and with $E = 4^3$, $P = 32$ in 3D (*right*) for $N = 4, 6, 8, 10, 12, 14, 16, 18$ and $m = 5, 7, 9, 11$. Parallel runs are performed on the Argonne BG/P

wave solution, we observe superconvergence for the EXP scheme. In practice, however, many physics problems involve more complicated wave phenomena than a single-mode wave structure. Thus, in general, convergence as a function of timestep size typically behaves as illustrated in the right side of Fig. 8. In particular, considering an accuracy of 1×10^{-7}, EXP allows a CFL number 8–9 times larger with $m = 7$–9, compared with RK4.

Figure 9 demonstrates the CPU cost between RK4 and the EXP scheme by examining (CPU time per timestep)/m per core depending on the total number of grid points for $N = 4, 6, 8, 10, 12, 16, 18$ with $E = 4^2$ on $P = 8$ cores in two dimensions and $E = 4^3$ on $P = 32$ in three dimensions. In 2D, for problem sizes greater than 10^3, the CPU cost per timestep per core divided by m is about 2 times larger with the EXP scheme compared with that divided by 5 with RK4. This implies that one can get cost reduction when using $m = 7, 9, 11$ by taking a 10–12 times larger timestep size for a single-mode solution and an 8–9 times larger timestep size for multimode solutions from the analysis of Fig. 8. For the problem sizes of less than 10^3, one can still gain cost reduction for single-mode solutions. In 3D, the CPU time per timestep per core divided by m increases 2–4 times larger for problem sizes of

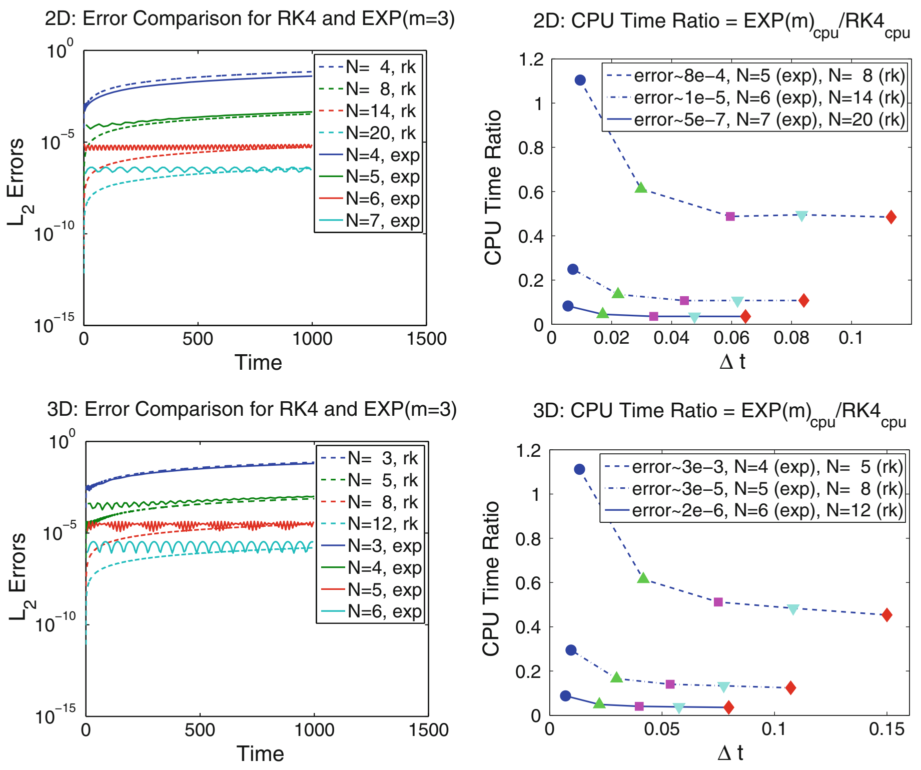

Fig. 10 Comparable errors and corresponding order N for EXP(m) and RK4 for $E = 3^2$ in 2D (*top left*) and $E = 3^3$ in 3D (*bottom left*) for long-time integration up to time $t = 1,000$. CPU time ratio EXP(m)/RK4 for the comparable level of accuracy with $m = 3(circle)$, $m = 5(triangle)$, $m = 7(square)$, $m = 9$ (*inverted triangle*), and $m = 11(diamond)$ in 2D for $E = 4^2$ (*top right*) and 3D for $E = 4^3$ (*bottom right*). Simulations are performed on $P = 2^4$ cores on the Argonne BG/P

$n = 10^4$–10^5 and almost no significant difference for problem sizes greater than $n = 10^5$. This promises that the EXP scheme can deliver dramatic cost reduction, allowing a larger timestep compared with RK4 as the problem size increases beyond 10^6 for very large-scale application problems.

Here we note that the tensor product evaluations in Eq. (40) require the arithmatic operations of $O(nN)$ for $n = E(N+1)^3$ in space and thus the total work scales as $O(mnN)$ per time step for the EXP scheme and $O(5nN)$ for the (5-stage) RK4. Then the total amount of work dramatically increases depending on N^4 as N increases. This explains the weak dependency of m or the five times of spatial operations in RK4 for the range of larger N which we observe in the CPU costs shown in Fig. 9.

Let us denote $\mathtt{t}_{\mathtt{EXP}}$ and $\mathtt{t}_{\mathtt{RK4}}$ as the CPU time per timestep per core divided by m and 5, respectively, with $\mathtt{t}_{\mathtt{EXP}} = a * \mathtt{t}_{\mathtt{RK4}}$. Assuming that, for a fixed resolution, the EXP scheme allows a timestep size b times larger than does RK4 (i.e., $\Delta t_{\mathrm{EXP}} = b * \Delta t_{\mathrm{RK4}}$), the total CPU time of RK4 and the EXP scheme for nsteps can be written as

$$\mathtt{T}^{\mathtt{C}}_{\mathtt{RK4}} = 5 * \mathtt{t}_{\mathtt{RK4}} * \mathtt{nsteps}, \tag{52}$$

$$\mathtt{T}^{\mathtt{C}}_{\mathtt{EXP}} = m * a * \mathtt{t}_{\mathtt{RK4}} * \frac{\mathtt{nsteps}}{b}, \tag{53}$$

which implies that one can expect a cost reduction when $b > \frac{m*a}{5}$ for the timestep size $\Delta t_{\mathtt{EXP}}$ for EXP(m). For large-scale problems, $a \approx 1$, so that one can estimate the CPU cost for EXP(m) as $\left(\frac{m}{5b}\right)\%$ of RK4. For the case of the right panel in Fig. 8 with relatively small $n = E(N+1)^2 = 1,936$, we observe $a \approx 2$ and $b \approx 9$ for $m = 9$ so that total CPU time reduction can be estimated as 60 % from the CPU time ratio $\mathtt{T}^{\mathtt{C}}_{\mathtt{EXP}}/\mathtt{T}^{\mathtt{C}}_{\mathtt{RK}} \approx 40$.

Figure 10 compares the total CPU time at a certain accuracy for single-mode solutions in 2D and 3D. The figure shows superconvergence with the EXP scheme using low resolution compared with RK4. The figures in the left panels show that the errors after long-time integration are approximately similar to the cases of RK4 with $N = 3$–20 using EXP($m = 3$) and $N = 3$–7. In such cases, we observe much higher reduction in cost, as shown in the right panels. For example, at the level of accuracy at 1×10^{-5}, one can achieve more than 70–90 % cost reduction for $m = 3, 5, 7, 9, 11$ with the EXP scheme in two and three dimensions.

5 Conclusions

We have presented an efficient high-order time integration method based on the Krylov subspace approximation using the modified Gram–Schmidt algorithm and a reorthogonalization technique for the Arnoldi process. For the spatial approximation, we used a SEDG scheme based on hexahedral spectral elements, which gives a fully diagonal mass matrix. We considered the source-free Maxwell's equations in nondimensional form. Computational results are shown for periodic solutions and waveguide simulations in 1D, 2D, and 3D. We demonstrate the convergence behaviors, long-time integrations, and the CPU cost of the SEDG scheme, compared with the RK4 (5-stage) and exponential time integration methods. Our numerical experiments show that the exponential time integration method allows a larger timestep size, compared with RK4, with significant cost reduction up to 70–90 % for single-mode solutions using Krylov subspace dimension $m = 3$–11 and about 60 % CPU time reduction for a two-dimensional solution containing 25 multiple modes with $m = 9$.

Acknowledgments This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

References

1. Moler, C., Loan, C.V.: Nineteen dubios ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. **45**(1), 3–49 (2003)
2. Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numer. **19**, 209–286 (2010)
3. Saad, Y.: Iterative Methods for Sparse Linear Systems. PWS Publishing, Boston (1996)
4. Saad, Y.: Analysis of some Krylov subspace approximation to the matrix exponential operator. SIAM J. Numer. Anal. **29**, 209–228 (1992)
5. Saad, Y.: Krylov subspace methods on supercomputers. SIAM J. Sci. Stat. Comput. **10**(6), 1200–1232 (1989)
6. Gallopoulos, E., Saad, Y.: Efficient solution of parabolic equations by Krylov approximation methods. SIAM J. Sci. Stat. Comput. **13**(5), 1236–1264 (1992)
7. Novati, P.: A low cost Arnoldi method for large linear initial value problems. Int. J. Comput. Math. **81**(7), 835–844 (2004)
8. Hochbruck, M., Lubich, C., Selhofer, H.: Exponential integrators for large systems of differential equations. SIAM J. Sci. Comput. **19**(5), 1552–1574 (1996)
9. Hochbruck, M., Lubich, C.: On the Krylov subspace approximations to the matrix exponential operator. SIAM J. Numer. Anal. **34**(5), 1911–1925 (1997)

10. Hesthaven, J.S., Warburton, T.: Nodal hihg-order methods on unstructured grids. I: time-domain solution of Maxwell's equations. J. Comput. Phys. **181**(1), 186–221 (2002)
11. Hesthaven, J.S., Warburton, T.: Nodal Discontinuous Galerkin Methods, Algorithms, Analysis, and Applications, Texts in Applied Mathematics. Springer, Berlin (2008)
12. Cockburn, B., Li, F., Shu, C.W.: Locally divergence-free discontinuous Galerkin methods. J. Comp. Phys. **194**, 588–610 (2004)
13. Rieben, R., White, D., Rodrigue, R.: High-order symplectic integration methods for finite element solutions to time dependent Maxwell equations. IEEE Trans. Antennas Propag. **56**(8), 2190–2195 (2004)
14. Nédeléc, J.C.: Mixed finite elements in R3. Numer. Math. **159**(1), 315–341 (1980)
15. Forest, E., Ruth, R.D.: Fourth-order sympletic integration. Physica D **43**, 105–117 (1990)
16. Candy, J., Rozmus, W.: A simplectic integration algorithm for separable Hamiltonian functions. J. Comput. Phys. **92**, 230–256 (1991)
17. Golub, G.H., Van Loan, C.F.: Matrix Computations. North Oxford Academic, England (1986)
18. Strom, T.: On logarithmic norms. SIAM J. Numer. Anal. **12**(5), 741–753 (1975)
19. Parlett, B.N.: The Symmetric Eigenvalue Problem. Prentice Hall, Englewood Clifts, N.J. (1980)
20. Deville, M.O., Fischer, P.F., Mund, E.H.: High-Order Methods for Incompressible Fluid Flow. Cambridge Monographs on Applied and Computational Mathematics, vol. 9. Cambridge University Press, Cambridge (2002)
21. Hesthaven, J.S., Gottlieb, S., Gottlieb, D.: Spectral Methods for Time-Dependent Problems. Cambridge Monographs on Applied and Computational Mathematics, vol. 21. Cambridge University Press, Cambridge (2007)
22. LAPACK, Linear Algebra PACKage, http://www.netlib.org/lapack
23. Gray, S.K., Kupka, T.: Propagation of light in metallic nanowire arrays: Finite-difference time domain studies of silver cylinders. Phys. Rev. B **68**, 045415/1–045415/11 (2003)
24. Oliva, J.M., Gray, S.K.: Theoretical study of dielectrically coated metallic nanowires. Chem. Phys. Lett. **379**, 325–331 (2003)
25. Zagorodnov, I.: TE/TM field solver for particle beam simulations without numerical Cherenkov radiation. Phys. Rev. Spec. Top. Accel. Beams **8**, 042001 (2005)
26. Gjonaj, E., Lau, T., Schnepp, S., Wolfheimer, F., Weiland, T.: Accurate modeling of charged particle beams in linear accelerators. New J. Phys. **8**, 285 (2006)
27. Min, M.S., Lee, T.W., Fischer, P.F., Gray, S.K.: Fourier spectral simulations and Gegenbauer reconstructions for electromagnetic waves in the presence of a metal nanoparticle. J. Comput. Phys. **213**(2), 730–747 (2006)
28. Min, M.S., Fischer, P.F., Montgomery, J., Gray, S.K.: Large-scale electromagnetic modeling based on high-order methods: nanoscience applications. J. Phys. Conf. Ser. **180**, 012016 (2009)
29. Min, M.S., Fischer, P.F., Chae, Y.C.: Spectral-element discontinuous Galerkin simulations for bunched beam in accelerating structures. In: Proceedings of PAC07, pp. 3432–3434 (2007)
30. Min, M.S., Lee, T.: A spectral-element discontinuous Galerkin lattice-Boltzmann method for incompressible flows. J. Comput. Phys. **230**, 245–259 (2011)
31. Taflove, A., Hagness, S.C.: Computational Electrodynamics, The Finite Difference Time Domain Method. Artech House, Norwood, MA (2000)
32. Wolf, D.A.: Essentials of Electromagnetics for Engineering. Cambridge University Press, Cambridge (2000)
33. Carpenter, M.H., Kennedy, C.: Fourth-order $2N$-storage Runge-Kutta schemes, NASA Report TM 109112, NASA Langley Research Center (1994)
34. Deville, M.O., Fischer, P.F., Mund, E.H.: High-Order Methods for Incompressible Fluid Flow. Cambridge University Press, Cambridge (2002)