J Sci Comput (2017) 73:980–1027
DOI 10.1007/s10915-017-0398-7

Computational Complexity Study on Krylov Integration Factor WENO Method for High Spatial Dimension Convection–Diffusion Problems

Dong Lu[1] · **Yong-Tao Zhang**[1]

Received: 28 July 2016 / Revised: 13 February 2017 / Accepted: 20 February 2017 /
Published online: 13 March 2017

Abstract Integration factor (IF) methods are a class of efficient time discretization methods for solving stiff problems via evaluation of an exponential function of the corresponding matrix for the stiff operator. The computational challenge in applying the methods for partial differential equations (PDEs) on high spatial dimensions (multidimensional PDEs) is how to deal with the matrix exponential for very large matrices. Compact integration factor methods developed in Nie et al. (J Comput Phys 227:5238–5255, 2008) provide an approach to reduce the cost prohibitive large matrix exponentials for linear diffusion operators with constant diffusion coefficients in high spatial dimensions to a series of much smaller one dimensional computations. This approach is further developed in Wang et al. (J Comput Phys 258:585–600, 2014) to deal with more complicated high dimensional reaction–diffusion equations with cross-derivatives in diffusion operators. Another approach is to use Krylov subspace approximations to efficiently calculate large matrix exponentials. In Chen and Zhang (J Comput Phys 230:4336–4352, 2011), Krylov subspace approximation is directly applied to the implicit integration factor (IIF) methods for solving high dimensional reaction–diffusion problems. Recently the method is combined with weighted essentially non-oscillatory (WENO) schemes in Jiang and Zhang (J Comput Phys 253:368–388, 2013) to efficiently solve semilinear and fully nonlinear convection–reaction–diffusion equations. A natural question that arises is how these two approaches may perform differently for various types of problems. In this paper, we study the computational power of Krylov IF-WENO methods for solving high spatial dimension convection–diffusion PDE problems (up to four

Dedicated to Professor Chi-Wang Shu on the occasion of his 60th birthday.

Research supported by NSF Grant DMS-1620108.

✉ Yong-Tao Zhang
yzhang10@nd.edu

Dong Lu
dlv1@nd.edu

[1] Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, USA

spatial dimensions). Systematical numerical comparison and complexity analysis are carried out for the computational efficiency of the two different approaches. We show that although the Krylov IF-WENO methods have linear computational complexity, both the compact IF method and the Krylov IF method have their own advantages for different type of problems. This study provides certain guidance for using IF-WENO methods to solve general high spatial dimension convection–diffusion problems.

Keywords Implicit integration factor methods · Weighted essentially non-oscillatory schemes · Krylov subspace approximation · High spatial dimensions · Convection–diffusion equations

1 Introduction

Efficient and accurate temporal numerical schemes are important for the performance of high order accuracy numerical simulations. A number of state-of-the-art high order time-stepping methods were developed in the literature. Here we just give a few examples and do not provide a complete list. For example, the total variation diminishing (TVD) Runge–Kutta (RK) schemes [11,12,41,42]; spectral deferred correction (SDC) methods [4,8,16,26,34]; high order implicit–explicit (IMEX) multistep/RK methods [1,23,25,48,53]; hybrid methods of SDC and high order RK schemes [6]; etc.

Integration factor (IF) methods are a class of "exactly linear part" time discretization methods for the solution of nonlinear partial differential equations (PDEs) with the linear highest spatial derivatives. This class of methods performs the time evolution of the stiff linear operator via evaluation of an exponential function of the corresponding matrix. Hence the integration factor type time discretization can remove both the stability constrain and time direction numerical errors from the high order derivatives [3,7,22,24,33]. Here time direction numerical errors are numerical errors for solving the semi-discretized ODE system resulting from spatial discretizations of the PDE. In [37], a class of efficient implicit integration factor (IIF) methods were developed for solving systems with both stiff linear and nonlinear terms. A novel property of the methods is that the implicit terms are free of the exponential operation of the linear terms. Hence the exact evaluation of the linear part is decoupled from the implicit treatment of the nonlinear terms. As a result, if the nonlinear terms do not involve spatial derivatives, the size of the nonlinear system arising from the implicit treatment is independent of the number of spatial grid points; it only depends on the number of the original PDEs. This distinguishes IIF methods [37] from implicit exponential time differencing (ETD) methods in [3].

Nonlinear convection–diffusion–reaction (CDR) systems of equations [18] are common mathematical models in applications from biology, chemistry and physics. A CDR system defined on a multidimensional spatial domain has the following general form

$$\vec{u}_t + \sum_{i=1}^{d} \vec{f}_i(\vec{u})_{x_i} = \nabla \cdot (\mathbf{D}(\vec{u})\nabla\vec{u}) + \vec{r}(\vec{u}), \tag{1}$$

where $\vec{u}$ is the unknown vector function, $\vec{f}_i$, $i = 1, \ldots, d$ are flux vector functions in d spatial dimensions, $\mathbf{D}(\vec{u})$ is the diffusion matrix and it could be nonlinear, and $\vec{r}$ is the reaction term. Often the CDR models in applications have nonlinear convection and reaction terms, but a linear diffusion term $\nabla \cdot (\mathbf{D}\nabla\vec{u})$, where $\mathbf{D}$ is the diffusion matrix that is independent of $\vec{u}$. In such case, the system is semilinear. To numerically solve this time-dependent problem

(1), a nonlinear stable discretization suitable for hyperbolic PDEs is needed for the nonlinear convection terms, to deal with the convection-dominated cases or a spatial mixture of convection-dominated and diffusion-dominated cases. Weighted essentially non-oscillatory (WENO) schemes are such kind of nonlinear stable discretizations. They are a class of popular high order numerical methods for solving hyperbolic PDEs whose solutions have complex solution structures. It is robust to apply WENO schemes in discretizing the convection terms in a general convection–diffusion problem, as that shown in [32]. We use WENO schemes to solve convection–diffusion equations so that various situations in a general problem can be dealt with directly.

WENO schemes have the advantage of attaining uniform high order accuracy in smooth regions while maintaining sharp and essentially monotone transitions in large gradient regions of the solution. WENO schemes are designed based on the successful ENO schemes in [13,42]. The first WENO scheme was constructed in [28] for a third order finite volume version. In [19], third and fifth order finite difference WENO schemes in multi-space dimensions were constructed, with a general framework for the design of the smoothness indicators and non-linear weights. The main idea of the WENO scheme is to form a weighted combination of several local reconstructions based on different stencils (usually referred to as small stencils) and use it as the final WENO reconstruction. The combination coefficients (also called non-linear weights) depend on the linear weights, often chosen to increase the order of accuracy over that on each small stencil, and on the smoothness indicators which measure the smoothness of the reconstructed function in the relevant small stencils. Hence an adaptive interpolation or reconstruction procedure is actually the essential part of the WENO schemes. Later, WENO schemes on unstructured meshes (e.g. arbitrary triangular or tetrahedral meshes) were developed to deal with complex domain geometries, see e.g. [15,30,51,52].

Recently, we developed IIF-WENO methods for solving nonlinear CDR systems in [20]. The methods can be designed for arbitrary order of accuracy. The stiffness of the system is resolved well and the methods are stable by using time step sizes which are just determined by the non-stiff hyperbolic part of the system. Large time step size computations are obtained. For CDR systems (1) defined on high dimensional spatial domains, the computational challenge in applying the methods is how to deal with the matrix exponential for very large matrices. Currently there are two approaches to deal with the large matrix exponential problem in IIF methods. One is the class of compact implicit integration factor (cIIF) methods in [29,38]. cIIF methods reduce the cost prohibitive large matrix exponentials for linear diffusion operators with constant diffusion coefficients in high spatial dimensions to a series of much smaller one dimensional computations. This approach is further extended in [49] as an array-representation technique to deal with more complicated high dimensional reaction–diffusion equations with cross-derivatives in diffusion operators. The method is termed as array-representation compact implicit integration factor (AcIIF) method. Another approach is to use Krylov subspace approximations to efficiently calculate large matrix exponentials. In [5], Krylov subspace approximation is directly applied to the IIF methods for solving high dimensional reaction–diffusion problems. A natural question that arises is how these two approaches may perform differently for various types of problems when they are applied to solve more complicated CDR equations. In this paper, we study the computational power of Krylov IIF-WENO methods for solving high spatial dimension convection–diffusion PDE problems (up to four spatial dimensions) by direct numerical simulations. Systematical numerical comparison and complexity analysis are carried out for the computational efficiency of the two different approaches. We show that although the Krylov IIF-WENO methods have linear computational complexity, both the compact IIF method and

the Krylov IIF method have their own advantages for different type of problems. This study provides certain guidance for using IIF-WENO methods to solve high spatial dimension problems.

The rest of the paper is organized as following. In Sect. 2, we first review the IIF-WENO methods for solving CDR equations developed in [20]. Then we present two different approach to deal with the high dimensional problems, i.e., the direct Krylov approach and the AcIIF approach. The AcIIF method was developed to solve reaction–diffusion equations in [49]. In order to compare it with the Krylov approach, we combine the AcIIF method with WENO method for solving CDR equations. In Sect. 3, we perform systematical numerical comparison and complexity analysis for applying these two approaches to various high dimensional problems including three and four dimensional Fokker–Planck equations. Discussions and conclusions are given in Sect. 4.

2 Numerical Methods

In this section, we first briefly review the IIF-WENO methods for solving CDR equations developed in [20]. Then we present two approaches for dealing with high dimensional problems. For the AcIIF method designed in [49], we combine it with WENO method and derive the corresponding schemes for solving CDR equations.

2.1 IIF-WENO Methods

The method of lines (MOL) approach is applied to the Eq. (1). For the simplicity of presentation, we consider the scalar equation case. The system case is solved component by component following the same procedure as the scalar case. For nonlinear convection terms $\sum_{i=1}^{d} f_i(u)_{x_i}$, the third order finite difference WENO scheme with Lax–Friedrichs flux splitting [43] is used. The second or fourth order central finite difference scheme (depending on the order of accuracy of IIF time discretizations) is used to discretize the diffusion terms.

For the convection terms, the conservative finite-difference schemes we use approximate the point values at a uniform (or smoothly varying) grid in a conservative fashion. The finite difference WENO schemes approximate derivatives of multi-dimension in a dimension by dimension way. For example, the x-direction derivative $f(u)_x$ at a grid point is approximated by a conservative flux difference

$$f(u)_x|_{x=x_i} \approx \frac{1}{\Delta x}\left(\hat{f}_{i+1/2} - \hat{f}_{i-1/2}\right), \tag{2}$$

where for the third order WENO scheme the numerical flux $\hat{f}_{i+1/2}$ depends on the three-point values $f(u_l)$ (here for the simplicity of notations, we use u_l to denote the value of the numerical solution u at the point x_l along the line $y = y_j, z = z_k$ with the understanding that the value could be different for different y and z coordinates), $l = i-1, i, i+1$, when the wind is positive (i.e., when $f'(u) \geq 0$ for the scalar case, or when the corresponding eigenvalue is positive for the system case with a local characteristic decomposition). This numerical flux $\hat{f}_{i+1/2}$ is written as a convex combination of two second order numerical fluxes based on two different substencils of two points each, and the combination coefficients depend on a "smoothness indicator" measuring the smoothness of the solution in each substencil. The detailed formulae is

$$\hat{f}_{i+1/2} = w_0\left[\frac{1}{2}f(u_i) + \frac{1}{2}f(u_{i+1})\right] + w_1\left[-\frac{1}{2}f(u_{i-1}) + \frac{3}{2}f(u_i)\right], \tag{3}$$

where

$$w_r = \frac{\alpha_r}{\alpha_1 + \alpha_2}, \quad \alpha_r = \frac{d_r}{(\epsilon + \beta_r)^2}, \quad r = 0, 1. \tag{4}$$

$d_0 = 2/3, d_1 = 1/3$ are called the "linear weights", and $\beta_0 = (f(u_{i+1}) - f(u_i))^2, \beta_1 = (f(u_i) - f(u_{i-1}))^2$ are called the "smoothness indicators". ϵ is a small positive number chosen to avoid the denominator becoming 0. We take $\epsilon = 10^{-3}$ in this paper.

When the wind is negative (i.e., when $f'(u) < 0$), right-biased stencil with numerical values $f(u_i)$, $f(u_{i+1})$ and $f(u_{i+2})$ are used to construct a third order WENO approximation to the numerical flux $\hat{f}_{i+1/2}$. The formulae for negative and positive wind cases are symmetric with respect to the point $x_{i+1/2}$. For the general case of $f(u)$, we perform the "Lax–Friedrichs flux splitting"

$$\begin{aligned} f^+(u) &= \frac{1}{2}(f(u) + \alpha u), \\ f^-(u) &= \frac{1}{2}(f(u) - \alpha u), \end{aligned} \tag{5}$$

where $\alpha = \max_u |f'(u)|$. $f^+(u)$ is the positive wind part, and $f^-(u)$ is the negative wind part. Corresponding WENO approximations are applied to find numerical fluxes $\hat{f}^+_{i+1/2}$ and $\hat{f}^-_{i+1/2}$ respectively. Similar procedures are applied to the other directions for $g(u)_y$ and $h(u)_z$. See [19,43] for more details. For diffusion terms, central differences are used. After spatial discretizations, a semi-discretized ODE system

$$\frac{d\vec{U}}{dt} = \vec{F}_d(\vec{U}) + \vec{F}_a(\vec{U}) + \vec{R}(\vec{U}) \tag{6}$$

is obtained. Here $\vec{U} = (u_i)_{1\le i\le N}$, $\vec{F}_d(\vec{U}) = (\hat{F}_{di}(\vec{U}))_{1\le i\le N}$, $\vec{F}_a(\vec{U}) = (\hat{F}_{ai}(\vec{U}))_{1\le i\le N}$, $\vec{R} = (r(u_i))_{1\le i\le N}$. N is the total number of grid points, $\vec{F}_d(\vec{U})$ is the approximation for the diffusion terms by the second or fourth order finite difference schemes, and $\hat{F}_{di}$ is a linear or nonlinear function of numerical values on the approximation stencil. If the diffusion term is linear, $\vec{F}_d(\vec{U}) = C\vec{U}$ where C is the approximation matrix for the linear diffusion operator by the central finite difference scheme. $\vec{F}_a(\vec{U})$ is the approximation for the nonlinear advection terms by the third order finite difference WENO scheme, and $\hat{F}_{ai}$ is a nonlinear function of several numerical values on the WENO approximation stencil. $\vec{R}(\vec{U})$ is the nonlinear reaction term, and $r(u_i)$ is a nonlinear function which only depends on numerical values at one grid point. In [20], we developed a method to deal with the nonlinear diffusion terms by factoring out the linear part which mainly contributes to the stiffness of the nonlinear diffusion terms, then applying the integration factor approach to remove this stiffness. In this paper, our main focus is on studying the computational complexity of Krylov and compact IIF methods for high dimensional problems. Hence we simplify our discussions to problems with linear diffusion, i.e., $\vec{F}_d(\vec{U}) = C\vec{U}$. IIF methods for (6) are constructed by exactly integrating the linear part of the system. Directly multiply (6) by the integration factor e^{-Ct} and integrate over one time step from t_n to $t_{n+1} \equiv t_n + \Delta t_n$ to obtain

$$\begin{aligned} \vec{U}(t_{n+1}) = e^{C\Delta t_n}\vec{U}(t_n) &+ e^{C\Delta t_n}\int_0^{\Delta t_n} e^{-C\tau}\vec{F}_a(\vec{U}(t_n+\tau))d\tau \\ &+ e^{C\Delta t_n}\int_0^{\Delta t_n} e^{-C\tau}\vec{R}(\vec{U}(t_n+\tau))d\tau. \end{aligned} \tag{7}$$

Two of the nonlinear terms in (7) have different properties. The nonlinear reaction term $\vec{R}(\vec{U})$ is usually stiff but local, while the nonlinear term $\vec{F}_a(\vec{U})$ derived from WENO approxima-

tions to the convection term is nonstiff but couples numerical values at grid points of the stencil. Hence we use different methods to treat them and avoid solving a large coupled nonlinear system. For the stiff reaction term $e^{-C\tau}\vec{R}(\vec{U}(t_n+\tau))$, we approximate it implicitly by an $(r-1)$th order Lagrange polynomial with interpolation points at $t_{n+1}, t_n, \ldots, t_{n+2-r}$. The nonstiff convection term is highly nonlinear due to the WENO approximations. Different from the nonlinear reaction term, we approximate the nonlinear convection term $e^{-C\tau}\vec{F}_a(\vec{U}(t_n+\tau))$ explicitly by an $(r-1)$th order Lagrange polynomial with interpolation points at $t_n, t_{n-1}, \ldots, t_{n+1-r}$. The rth order IIF scheme for CDR equations is obtained as

$$\vec{U}_{n+1} = e^{C\Delta t_n}\vec{U}_n + \Delta t_n \left\{ \alpha_{n+1}\vec{R}(\vec{U}_{n+1}) + \sum_{i=2-r}^{0} \alpha_{n+i} e^{C(\Delta t_n-\tau_i)}\vec{R}(\vec{U}_{n+i}) + \sum_{i=1-r}^{0} \beta_{n+i} e^{C(\Delta t_n-\tau_i)}\vec{F}_a(\vec{U}_{n+i}) \right\}, \tag{8}$$

where the coefficients

$$\alpha_{n+i} = \frac{1}{\Delta t_n}\int_0^{\Delta t_n} \prod_{j=2-r, j\neq i}^{1} \frac{\tau-\tau_j}{\tau_i-\tau_j} d\tau, \quad i = 1, 0, -1, \ldots, 2-r; \tag{9}$$

$$\beta_{n+i} = \frac{1}{\Delta t_n}\int_0^{\Delta t_n} \prod_{j=1-r, j\neq i}^{0} \frac{\tau-\tau_j}{\tau_i-\tau_j} d\tau, \quad i = 0, -1, -2, \ldots, 1-r. \tag{10}$$

$\tau_1 = \Delta t_n, \tau_0 = 0, \tau_i = -\sum_{k=i}^{-1}\Delta t_{n+k}$ for $i = -1, -2, -3, \ldots, 1-r$. $\vec{U}_{n+i}$ is the numerical solution for $\vec{U}(t_{n+i})$. Specifically, the second order scheme (IIF2) is of the following form

$$\vec{U}_{n+1} = e^{C\Delta t_n}\vec{U}_n + \Delta t_n \left\{ \alpha_{n+1}\vec{R}(\vec{U}_{n+1}) + \alpha_n e^{C\Delta t_n}\vec{R}(\vec{U}_n) + \beta_{n-1} e^{C(\Delta t_n+\Delta t_{n-1})}\vec{F}_a(\vec{U}_{n-1}) + \beta_n e^{C\Delta t_n}\vec{F}_a(\vec{U}_n) \right\}, \tag{11}$$

where

$$\alpha_n = \frac{1}{2},\ \alpha_{n+1} = \frac{1}{2},\ \beta_{n-1} = -\frac{\Delta t_n}{2\Delta t_{n-1}},$$
$$\beta_n = \frac{1}{\Delta t_{n-1}}\left(\frac{\Delta t_n}{2} + \Delta t_{n-1}\right).$$

And the third order scheme (IIF3) is

$$\begin{aligned}\vec{U}_{n+1} = e^{C\Delta t_n}\vec{U}_n + \Delta t_n \Big\{ &\alpha_{n+1}\vec{R}(\vec{U}_{n+1}) + \alpha_n e^{C\Delta t_n}\vec{R}(\vec{U}_n) + \alpha_{n-1} e^{C(\Delta t_n+\Delta t_{n-1})}\vec{R}(\vec{U}_{n-1}) \\ &+ \beta_{n-2} e^{C(\Delta t_n+\Delta t_{n-1}+\Delta t_{n-2})}\vec{F}_a(\vec{U}_{n-2}) + \beta_{n-1} e^{C(\Delta t_n+\Delta t_{n-1})}\vec{F}_a(\vec{U}_{n-1}) \\ &+ \beta_n e^{C\Delta t_n}\vec{F}_a(\vec{U}_n) \Big\},\end{aligned} \tag{12}$$

where

$$\alpha_{n+1} = \frac{1}{(\Delta t_n + \Delta t_{n-1})}\left(\frac{\Delta t_n}{3} + \frac{\Delta t_{n-1}}{2}\right),$$
$$\alpha_n = \frac{1}{\Delta t_{n-1}}\left(\frac{\Delta t_n}{6} + \frac{\Delta t_{n-1}}{2}\right),$$

$$\alpha_{n-1} = -\frac{\Delta t_n^2}{6\Delta t_{n-1}\left(\Delta t_{n-1}+\Delta t_n\right)},$$
$$\beta_n = 1 + \frac{1}{\Delta t_{n-1}\left(\Delta t_{n-1}+\Delta t_{n-2}\right)}\left[\frac{\Delta t_n^2}{3} + \frac{\Delta t_n}{2}\left(2\Delta t_{n-1}+\Delta t_{n-2}\right)\right],$$
$$\beta_{n-1} = -\frac{1}{\Delta t_{n-1}\Delta t_{n-2}}\left[\frac{\Delta t_n^2}{3} + \frac{\Delta t_n}{2}\left(\Delta t_{n-1}+\Delta t_{n-2}\right)\right],$$
$$\beta_{n-2} = \frac{1}{\Delta t_{n-2}\left(\Delta t_{n-1}+\Delta t_{n-2}\right)}\left(\frac{\Delta t_n^2}{3} + \frac{\Delta t_n \Delta t_{n-1}}{2}\right).$$

Remark Theoretical analysis including stability and error analysis of the IIF schemes for convection–diffusion–reaction equations is given in [20,21]. Due to the nonlinearity of WENO schemes [43] and the global property of the exponential integrator in the IIF schemes, theoretical analysis of the complete IIF-WENO schemes is still an open problem and it will be one of our future work.

2.2 Two Approaches for High Dimensional Problems

The efficiency of IIF schemes for high dimensional problems largely depends on the methods to evaluate the product of the matrix exponential and a vector, for example $e^{C\Delta t}v$. For PDEs defined on high spatial dimensions (2D and above), a large and sparse matrix C is generated in the schemes (8). But the exponential matrix $e^{C\Delta t}$ is dense. For high dimensional problems, direct computation and storage of such exponential matrix are prohibitive in terms of both CPU cost and computer memory. Two approaches have been developed to solve this problem. Here we discuss and compare the computational efficiency of these two approaches when they are applied to IIF-WENO methods for solving high dimensional problems. We first review the Krylov approximation method. The Krylov approximation method was applied to IIF schemes in [5]. It has been applied for solving CDR equations in [20].

2.2.1 Krylov Approximation Method

Notice that we do *not* need the full exponential matrices such as $e^{C\Delta t}$ itself, but only the products of the exponential matrices and some vectors in the schemes (8). The Krylov subspace approximations to the matrix exponential operator is an excellent choice in terms of both accuracy and efficiency. Follow the literature (e.g. [10,36]), we describe the Krylov subspace methods to approximate $e^{C\Delta t}v$ as following.

The large sparse matrix C is projected to the Krylov subspace

$$K_M = \text{span}\left\{v, Cv, C^2v, \ldots, C^{M-1}v\right\}. \tag{13}$$

The dimension M of the Krylov subspace is **much** smaller than the dimension N of the large sparse matrix C. In all numerical computations of this paper, we take $M = 25$ for different N, and accurate results are obtained in the numerical experiments. An orthonormal basis $V_M = [v_1, v_2, v_3, \ldots, v_M]$ of the Krylov subspace K_M is generated by the well-known Arnoldi algorithm [47]:

1. Compute the initial vector: $v_1 = v/\|v\|_2$.
2. Perform iterations: Do $j = 1, 2, \ldots, M$:
   1) Compute the vector $w = Cv_j$.
   2) Do $i = 1, 2, \ldots, j$:

(a) Compute the inner product $h_{i,j} = (w, v_i)$.
(b) Compute the vector $w = w - h_{i,j} v_i$.
3) Compute $h_{j+1,j} = \|w\|_2$.
4) If $h_{j+1,j} \equiv 0$, then
stop the iteration;
else
compute the next basis vector $v_{j+1} = w/h_{j+1,j}$.

In the Arnoldi algorithm, if $h_{j+1,j} \equiv 0$ for some $j < M$, it means that the convergence has occurred and the Krylov subspace is $K_M = \text{span}\{v_1, v_2, \ldots, v_j\}$, so the iteration can be stopped at this step j, and we assign the value of this j to M. This algorithm will produce an orthonormal basis V_M of the Krylov subspace K_M. Denote the $M \times M$ upper Hessenberg matrix consisting of the coefficients $h_{i,j}$ by H_M. Since the columns of V_M are orthogonal, we have

$$H_M = V_M^T C V_M. \tag{14}$$

This means that the very small Hessenberg matrix H_M represents the projection of the large sparse matrix C to the Krylov subspace K_M, with respect to the basis V_M. Also since V_M is orthonormal, the vector $V_M V_M^T e^{C\Delta t} v$ is the orthogonal projection of $e^{C\Delta t} v$ on the Krylov subspace K_M, namely, it is the best approximation to $e^{C\Delta t} v$ in K_M. Therefore

$$e^{C\Delta t} v \simeq V_M V_M^T e^{C\Delta t} v = \beta V_M V_M^T e^{C\Delta t} v_1 = \beta V_M V_M^T e^{C\Delta t} V_M e_1,$$

where $\beta = \|v\|_2$, and e_1 denotes the first column of the $M \times M$ identity matrix I_M. Using (14) we obtain the approximation

$$e^{C\Delta t} v \simeq \beta V_M e^{H_M \Delta t} e_1. \tag{15}$$

Thus the large $e^{C\Delta t}$ matrix exponential problem is replaced with the much smaller problem $e^{H_M \Delta t}$. The small matrix exponential $e^{H_M \Delta t}$ will be computed using a scaling and squaring algorithm with a Padé approximation, see [10,14,36]. Then the Krylov approximations are directly applied in schemes (8), (11) or (12) to obtain Krylov IIF schemes for CDR equations [20]. The rth order Krylov IIF scheme for CDR equations has the following form

$$\begin{aligned} \vec{U}_{n+1} = {} & \Delta t_n \alpha_{n+1} \vec{R}(\vec{U}_{n+1}) + \gamma_{0,n} V_{M,0,n} e^{H_{M,0,n}\Delta t_n} e_1 \\ & + \Delta t_n \left(\beta_{n+1-r} \gamma_{1-r,n} V_{M,1-r,n} e^{H_{M,1-r,n}(\Delta t_n - \tau_{1-r})} e_1 \right. \\ & \left. + \sum_{i=2-r}^{-1} \gamma_{i,n} V_{M,i,n} e^{H_{M,i,n}(\Delta t_n - \tau_i)} e_1 \right), \end{aligned} \tag{16}$$

where $\gamma_{0,n} = \|U_n + \Delta t_n(\alpha_n \vec{R}(\vec{U}_n) + \beta_n \vec{F}_a(\vec{U}_n))\|_2$, $V_{M,0,n}$ and $H_{M,0,n}$ are orthonormal basis and upper Hessenberg matrix generated by the Arnoldi algorithm with the initial vector $U_n + \Delta t_n(\alpha_n \vec{R}(\vec{U}_n) + \beta_n \vec{F}_a(\vec{U}_n))$. $\gamma_{1-r,n} = \|\vec{F}_a(\vec{U}_{n+1-r})\|_2$, $V_{M,1-r,n}$ and $H_{M,1-r,n}$ are orthonormal basis and upper Hessenberg matrix generated by the Arnoldi algorithm with the initial vector $\vec{F}_a(\vec{U}_{n+1-r})$. $\gamma_{i,n} = \|\alpha_{n+i} \vec{R}(\vec{U}_{n+i}) + \beta_{n+i} \vec{F}_a(\vec{U}_{n+i})\|_2$, $V_{M,i,n}$ and $H_{M,i,n}$ are orthonormal basis and upper Hessenberg matrix generated by the Arnoldi algorithm with the initial vectors $\alpha_{n+i} \vec{R}(\vec{U}_{n+i}) + \beta_{n+i} \vec{F}_a(\vec{U}_{n+i})$, for $i = 2-r, 3-r, \ldots, -1$. Notice that $V_{M,0,n}$, $V_{M,1-r,n}$ and $V_{M,i,n}$, $i = 2-r, 3-r, \ldots, -1$ are orthonormal bases of different Krylov subspaces for the same matrix C, which are generated with different initial vectors in the Arnoldi algorithm. Specifically, the second order Krylov IIF (KrylovIIF2) scheme has the following form

$$\vec{U}_{n+1} = \frac{1}{2}\Delta t_n \vec{R}(\vec{U}_{n+1}) + \gamma_{0,n} V_{M,0,n} e^{H_{M,0,n}\Delta t_n} e_1 - \frac{(\Delta t_n)^2}{2\Delta t_{n-1}} \left(\gamma_{-1,n} V_{M,-1,n} e^{H_{M,-1,n}(\Delta t_n + \Delta t_{n-1})} e_1\right), \tag{17}$$

where $\gamma_{0,n} = \left\| U_n + \Delta t_n \left(\frac{1}{2}\vec{R}(\vec{U}_n) + \frac{1}{\Delta t_{n-1}}(\frac{\Delta t_n}{2} + \Delta t_{n-1})\vec{F}_a(\vec{U}_n)\right)\right\|_2$, $V_{M,0,n}$ and $H_{M,0,n}$ are orthonormal basis and upper Hessenberg matrix generated by the Arnoldi algorithm with the initial vector $U_n + \Delta t_n \left(\frac{1}{2}\vec{R}(\vec{U}_n) + \frac{1}{\Delta t_{n-1}}(\frac{\Delta t_n}{2} + \Delta t_{n-1})\vec{F}_a(\vec{U}_n)\right)$. $\gamma_{-1,n} = \|\vec{F}_a(\vec{U}_{n-1})\|_2$, $V_{M,-1,n}$ and $H_{M,-1,n}$ are orthonormal basis and upper Hessenberg matrix generated by the Arnoldi algorithm with the initial vector $\vec{F}_a(\vec{U}_{n-1})$. And the third order Krylov IIF (KrylovIIF3) scheme has the form

$$\begin{aligned}\vec{U}_{n+1} = {} & \frac{2\Delta t_n + 3\Delta t_{n-1}}{6(\Delta t_n + \Delta t_{n-1})}\Delta t_n \vec{R}(\vec{U}_{n+1}) + \gamma_{0,n} V_{M,0,n} e^{H_{M,0,n}\Delta t_n} e_1 \\ & + \Delta t_n \left(\frac{2(\Delta t_n)^2 + 3\Delta t_n \Delta t_{n-1}}{6\Delta t_{n-2}(\Delta t_{n-1} + \Delta t_{n-2})}\gamma_{-2,n} V_{M,-2,n} e^{H_{M,-2,n}(\Delta t_n + \Delta t_{n-1} + \Delta t_{n-2})} e_1\right. \\ & \left. + \gamma_{-1,n} V_{M,-1,n} e^{H_{M,-1,n}(\Delta t_n + \Delta t_{n-1})} e_1\right),\end{aligned} \tag{18}$$

where $\gamma_{0,n} = \|U_n + \Delta t_n(\alpha_n \vec{R}(\vec{U}_n) + \beta_n \vec{F}_a(\vec{U}_n))\|_2$, $V_{M,0,n}$ and $H_{M,0,n}$ are orthonormal basis and upper Hessenberg matrix generated by the Arnoldi algorithm with the initial vector $U_n + \Delta t_n(\alpha_n \vec{R}(\vec{U}_n) + \beta_n \vec{F}_a(\vec{U}_n))$. $\gamma_{-2,n} = \|\vec{F}_a(\vec{U}_{n-2})\|_2$, $V_{M,-2,n}$ and $H_{M,-2,n}$ are orthonormal basis and upper Hessenberg matrix generated by the Arnoldi algorithm with the initial vector $\vec{F}_a(\vec{U}_{n-2})$. $\gamma_{-1,n} = \|\alpha_{n-1}\vec{R}(\vec{U}_{n-1}) + \beta_{n-1}\vec{F}_a(\vec{U}_{n-1})\|_2$, $V_{M,-1,n}$ and $H_{M,-1,n}$ are orthonormal basis and upper Hessenberg matrix generated by the Arnoldi algorithm with the initial vectors $\alpha_{n-1}\vec{R}(\vec{U}_{n-1}) + \beta_{n-1}\vec{F}_a(\vec{U}_{n-1})$. See the Eq. (12) for values of $\alpha_n, \beta_n, \alpha_{n-1}, \beta_{n-1}$.

As that pointed out in [20], in the implementation of the Krylov approximation methods we do *not* store matrices C, because only multiplications of matrices C with a vector are needed in the methods, and they correspond to certain finite difference operations.

Remark By the analysis in [10,17], an error estimation of the Krylov subspace approximation (15) is

$$||e^{C\Delta t}v - \beta V_M e^{H_M \Delta t} e_1||_2 \leq 10\beta e^{-M^2/(5\rho\Delta t)}, \tag{19}$$

where M is the dimension of the Krylov subspace, and eigenvalues of the matrix C are in the interval $[-4\rho, 0]$. For a fixed $\rho\Delta t$, the Krylov approximation error (19) decays exponentially with respect to the square of the Krylov subspace dimension M.

2.2.2 Compact/Array-Representation Method

We first review the compact IIF (cIIF) method and the array-representation compact IIF (AcIIF) method for solving high dimensional reaction–diffusion equations, developed in [38] and [49]. Then we discuss how to apply the cIIF/AcIIF method in the IIF-WENO schemes for solving high dimensional CDR equations.

(1) cIIF/AcIIF for reaction–diffusion equations

We illustrate the cIIF method by solving a two-dimensional reaction–diffusion equation with constant diffusion coefficient

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + R(u), \quad (x, y) \in \Omega = \{a < x < b, c < y < d\}, \tag{20}$$

with periodic boundary conditions in the $y-$direction and no-flux boundary conditions in the x-direction. The spatial domain is partitioned by a rectangular mesh with N_x+2 and N_y+2 grid points in each direction. The grid sizes are $h_x=\frac{b-a}{N_x+1}$, and $h_y=\frac{d-c}{N_y+1}$. Using the second order central difference discretization on the diffusion terms, a system of ODEs

$$\frac{du_{i,j}}{dt}=D\left(\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h_x^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{h_y^2}\right)+R(u_{i,j}) \tag{21}$$

is obtained. The idea of cIIF method [38] is that in stead of representing numerical values $u_{i,j}$ in a large vector, numerical values are organized and stored in a matrix (see (23)). The semi-discretized ODE system is written in a compact form

$$\frac{d\boldsymbol{U}}{dt}=\boldsymbol{A}\boldsymbol{U}+\boldsymbol{U}\boldsymbol{B}+R(\boldsymbol{U}), \tag{22}$$

where the three matrices $\boldsymbol{U}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ are

$$\boldsymbol{U}_{N_x\times(N_y+1)}=\begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,N_y} & u_{1,N_y+1}\\ u_{2,1} & u_{2,2} & \cdots & u_{2,N_y} & u_{2,N_y+1}\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ u_{N_x,1} & u_{N_x,2} & \cdots & u_{N_x,N_y} & u_{N_x,N_y+1}\end{pmatrix}, \tag{23}$$

$$\boldsymbol{A}_{N_x\times N_x}=\frac{D}{h_x^2}\begin{pmatrix} -\frac{2}{3} & \frac{2}{3} & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots\\ & & & 1 & -2 & 1\\ & & & & \frac{2}{3} & -\frac{2}{3}\end{pmatrix}, \tag{24}$$

$$\boldsymbol{B}_{(N_y+1)\times(N_y+1)}=\frac{D}{h_y^2}\begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 1\\ 1 & -2 & 1 & 0 & \cdots & 0\\ 0 & 1 & -2 & 1 & \cdots & 0\\ & & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & 1 & -2 & 1\\ 1 & 0 & \cdots & 0 & 1 & -2\end{pmatrix}. \tag{25}$$

Then following the similar procedure for deriving IIF methods [37], we multiply (22) by the integration factors $e^{-\boldsymbol{A}t}$ from the left and $e^{-\boldsymbol{B}t}$ from the right, and integrate over one time step from t_n to $t_{n+1}\equiv t_n+\triangle t$ to obtain

$$\boldsymbol{U}_{n+1}=e^{\boldsymbol{A}\triangle t}\boldsymbol{U}_n e^{\boldsymbol{B}\triangle t}+e^{\boldsymbol{A}\triangle t}\left(\int_0^{\triangle t}e^{-\boldsymbol{A}\tau}R(\boldsymbol{U}(t_n+\tau))e^{-\boldsymbol{B}\tau}d\tau\right)e^{\boldsymbol{B}\triangle t}. \tag{26}$$

We approximate the integrand in (26) by an $(r-1)$th order lagrange interpolation polynomial with interpolation points at $t_{n+1}, t_n, \ldots, t_{n+2-r}$, and obtain the rth order cIIF scheme for two-dimensional reaction–diffusion equations

$$\boldsymbol{U}_{n+1}=e^{\boldsymbol{A}\triangle t}\boldsymbol{U}_n e^{\boldsymbol{B}\triangle t}+\triangle t\Big(\alpha_1 R(\boldsymbol{U}_{n+1})+\sum_{i=0}^{r-2}\alpha_{-i}e^{(i+1)\boldsymbol{A}\triangle t}R(\boldsymbol{U}_{n-i})e^{(i+1)\boldsymbol{B}\triangle t}\Big), \tag{27}$$

where

$$\alpha_{-i} = \frac{1}{\triangle t}\int_0^{\triangle t}\prod_{\substack{k=-1\\k\neq i}}^{r-2}\frac{\tau + k\triangle t}{(k-i)\triangle t}d\tau, \quad -1 \le i \le r-2. \tag{28}$$

In particular, the second order cIIF scheme (cIIF2) is

$$\boldsymbol{U}_{n+1} = e^{\boldsymbol{A}\triangle t}\Big(\boldsymbol{U}_n + \frac{\triangle t}{2}R(\boldsymbol{U}_n)\Big)e^{\boldsymbol{B}\triangle t} + \frac{\triangle t}{2}R(\boldsymbol{U}_{n+1}). \tag{29}$$

Note that the matrices A and B have sizes of a one-dimensional problem. Hence in cIIF schemes (27), (29) for a two-dimensional problem, we only need to compute matrix exponentials for matrices with sizes of one-dimensional problems. This fact also holds for cIIF schemes of three-dimensional reaction–diffusion equations, as shown in [38].

In order to solve reaction–diffusion problems with cross-derivatives and non-constant diffusion coefficients on higher spatial dimensions, cIIF method has been extended to the array-representation compact IIF (AcIIF) method in [49]. We review the AcIIF method [49] in the following and then describe the procedure to apply this approach to our IIF schemes for CDR equations in the next subsection. The numerical solutions are stored in multi-dimensional arrays, for example, a two-dimensional array $U = (U_{k_1,k_2}), k_1 = 1,\ldots,N_x; k_2 = 1,\ldots,N_y+1$ for the two-dimensional problem (21)–(25). If we fix the second index k_2, the two-dimensional array U defines a vector

$$U(:,k_2) = (U_{1,k_2}, U_{2,k_2}, \ldots, U_{N_x,k_2})^T. \tag{30}$$

Then the array U can be considered as the collection of these vectors on a one-dimensional array, with k_2 going through from 1 to N_y+1. This collection is presented using symbol $\bigotimes$ in [49], so we can write

$$U = \bigotimes_{1\le k_2\le N_y+1} U(:,k_2). \tag{31}$$

The finite difference operators are linear operators in (21) since the diffusion terms here are linear. Define finite difference operators $\mathcal{L}_x$ and $\mathcal{L}_y$ as

$$(\mathcal{L}_x U)_{k_1,k_2} = D\left(\frac{U_{k_1+1,k_2} - 2U_{k_1,k_2} + U_{k_1-1,k_2}}{h_x^2}\right), \tag{32}$$

and

$$(\mathcal{L}_y U)_{k_1,k_2} = D\left(\frac{U_{k_1,k_2+1} - 2U_{k_1,k_2} + U_{k_1,k_2-1}}{h_y^2}\right), \tag{33}$$

then the semi-discretized scheme (21) with the array U can be written as

$$\frac{dU}{dt} = (\mathcal{L}_x + \mathcal{L}_y)U + R(U). \tag{34}$$

Apply IIF schemes, e.g., the second order IIF scheme (IIF2) [37] in (34) to obtain

$$U_{n+1} = e^{(\mathcal{L}_x+\mathcal{L}_y)\triangle t}\Big(U_n + \frac{\triangle t}{2}R(U_n)\Big) + \frac{\triangle t}{2}R(U_{n+1}). \tag{35}$$

To implement the scheme (35) using array-representation technique, we first represent

$$\mathcal{L}_x U = \bigotimes_{1\le k_2\le N_y+1} \boldsymbol{A}U(:,k_2), \tag{36}$$

where $\boldsymbol{A}$ is given in (24). So the exponential of $\mathcal{L}_x$ can have the array-representation

$$e^{\mathcal{L}_x \triangle t} U = \bigotimes_{1 \le k_2 \le N_y+1} e^{\boldsymbol{A} \triangle t} U(:, k_2). \tag{37}$$

Similarly,

$$e^{\mathcal{L}_y \triangle t} U = \bigotimes_{1 \le k_1 \le N_x} e^{\boldsymbol{B} \triangle t} U(k_1, :), \tag{38}$$

where $\boldsymbol{B}$ is given in (25). Since $\mathcal{L}_x$ and $\mathcal{L}_y$ commute with each other for this constant diffusion coefficient equation case, $e^{(\mathcal{L}_x+\mathcal{L}_y)\triangle t} = e^{\mathcal{L}_x \triangle t} e^{\mathcal{L}_y \triangle t}$. The array-representation form of the IIF2 scheme [37], i.e., the AcIIF2 scheme for the 2D reaction–diffusion equation (20), is

$$U_{n+1} - \frac{\triangle t}{2} R(U_{n+1}) = \bigotimes_{1 \le k_2 \le N_y+1} e^{\boldsymbol{A} \triangle t} \left(\bigotimes_{1 \le k_1 \le N_x} e^{\boldsymbol{B} \triangle t} V(k_1, :) \right)(:, k_2), \tag{39}$$

where $V = U_n + \frac{\triangle t}{2} R(U_n)$. Similarly the AcIIF2 scheme for a 3D reaction–diffusion equation with constant diffusion coefficient and without cross-derivatives is

$$\begin{aligned} &U_{n+1} - \frac{\triangle t}{2} R(U_{n+1}) \\ &= \bigotimes_{\substack{1 \le k_2 \le N_y \\ 1 \le k_3 \le N_z}} e^{\boldsymbol{A_{11}} \triangle t} \left(\bigotimes_{\substack{1 \le k_1 \le N_x \\ 1 \le k_3 \le N_z}} e^{\boldsymbol{A_{22}} \triangle t} \left(\bigotimes_{\substack{1 \le k_1 \le N_x \\ 1 \le k_2 \le N_y}} e^{\boldsymbol{A_{33}} \triangle t} V(k_1, k_2, :) \right)(k_1, :, k_3) \right)(:, k_2, k_3), \end{aligned} \tag{40}$$

where $V = U_n + \frac{\triangle t}{2} R(U_n)$, U is a three-dimensional array to store the numerical values of u, N_x, N_y, N_z are number of spatial grid points in x, y, z directions respectively. $\boldsymbol{A_{11}}$, $\boldsymbol{A_{22}}$, $\boldsymbol{A_{33}}$ are differential matrices for approximating diffusion operators in x, y, z directions respectively, and they have sizes of a one-dimensional problem, i.e., $N_x \times N_x$, $N_y \times N_y$ and $N_z \times N_z$.

It is easy to see that the AcIIF2 scheme (39) is equivalent to the cIIF2 scheme (29). As that pointed out in [49], AcIIF schemes are actually equivalent to cIIF schemes for reaction–diffusion equations without cross-derivatives. However, AcIIF schemes can be easily applied to more general high dimensional reaction–diffusion equations with cross-derivatives as shown in [49].

(2) AcIIF-WENO schemes for CDR equations

Since AcIIF method is an efficient approach for solving high dimensional reaction–diffusion equations, we apply it in the IIF-WENO schemes for solving high dimensional CDR equations. We present the schemes for the general three and four spatial dimension cases that CDR equations have cross-derivatives and the diffusion coefficients can be non-constant, such as the Fokker–Planck equations in the following Sect. 3. For such cases with non-constant diffusion coefficients, differential matrices can not commute and an operator splitting is needed to achieve the second order accuracy in AcIIF approach. Hence we use the second order AcIIF scheme here.

Consider the three dimensional case of CDR equation (1), $d = 3$, with cross-derivatives for the linear diffusion terms and periodic boundary conditions. For the simplicity of presentation, we consider the scalar equation case. The system case is solved component by component following the same procedure as the scalar case. The diffusion matrix $\mathbf{D}$ is

$$\mathbf{D} = \begin{pmatrix} a_1 + a_2 & b_1 & b_2 \\ b_1 & a_3 + c_1 & b_3 \\ b_2 & b_3 & c_2 + c_3 \end{pmatrix}, \tag{41}$$

where a_i, b_i and c_i, $i = 1, 2, 3$ are constant or non-constant coefficients of the diffusion terms. The diffusion terms can be grouped into three classes for the convenience of applying the AcIIF method, i.e., $(a_1 \frac{\partial^2}{\partial x_1{}^2} + 2b_1 \frac{\partial^2}{\partial x_1 \partial x_2} + c_1 \frac{\partial^2}{\partial x_2{}^2})u$, $(a_2 \frac{\partial^2}{\partial x_1{}^2} + 2b_2 \frac{\partial^2}{\partial x_1 \partial x_3} + c_2 \frac{\partial^2}{\partial x_3{}^2})u$, and $(a_3 \frac{\partial^2}{\partial x_2{}^2} + 2b_3 \frac{\partial^2}{\partial x_2 \partial x_3} + c_3 \frac{\partial^2}{\partial x_3{}^2})u$. Applying the second order IIF-WENO scheme (11) to the equation and re-grouping the exponential terms, we obtain

$$\begin{aligned} \vec{U}_{n+1} &= e^{C\triangle t_n}\Big(\vec{U}_n + \triangle t_n \alpha_n \vec{R}(\vec{U}_n) + \triangle t_n \beta_n \vec{F}_a(\vec{U}_n)\Big) \\ &\quad + e^{C(\triangle t_n + \triangle t_{n-1})}\Big(\triangle t_n \beta_{n-1} \vec{F}_a(\vec{U}_{n-1})\Big) + \triangle t_n \alpha_{n+1} \vec{R}(\vec{U}_{n+1}) \\ &= \mathbf{\Theta_1} + \mathbf{\Theta_2} + \triangle t_n \alpha_{n+1} \vec{R}(\vec{U}_{n+1}), \end{aligned} \tag{42}$$

where

$$\mathbf{\Theta_1} = e^{C\triangle t_n} \vec{V}_1, \quad \vec{V}_1 \triangleq \vec{U}_n + \triangle t_n \alpha_n \vec{R}(\vec{U}_n) + \triangle t_n \beta_n \vec{F}_a(\vec{U}_n), \tag{43}$$

$$\mathbf{\Theta_2} = e^{C(\triangle t_n + \triangle t_{n-1})} \vec{V}_2, \quad \vec{V}_2 \triangleq \triangle t_n \beta_{n-1} \vec{F}_a(\vec{U}_{n-1}). \tag{44}$$

$\alpha_n, \alpha_{n+1}, \beta_{n-1}, \beta_n$ are given in (11). Then we can apply the array representation approach in computations of the matrix exponentials. Numerical solutions for u are stored in a three-dimensional array U with size $N_1 \times N_2 \times N_3$, where N_1, N_2 and N_3 are numbers of grid points of three spatial directions respectively. First we use $\mathcal{L}_{12}$ to denote the second order central finite difference approximation of $(a_1 \frac{\partial^2}{\partial x_1{}^2} + 2b_1 \frac{\partial^2}{\partial x_1 \partial x_2} + c_1 \frac{\partial^2}{\partial x_2{}^2})$ as

$$\begin{aligned} (\mathcal{L}_{12}U)_{k_1,k_2,k_3} &= \frac{a_1}{h_1^2}(U_{k_1+1,k_2,k_3} - 2U_{k_1,k_2,k_3} + U_{k_1-1,k_2,k_3}) \\ &\quad + \frac{2b_1}{4h_1h_2}(U_{k_1+1,k_2+1,k_3} + U_{k_1-1,k_2-1,k_3} - U_{k_1+1,k_2-1,k_3} - U_{k_1-1,k_2+1,k_3}) \\ &\quad + \frac{c_1}{h_2^2}(U_{k_1,k_2+1,k_3} - 2U_{k_1,k_2,k_3} + U_{k_1,k_2-1,k_3}). \end{aligned} \tag{45}$$

where h_1, h_2 and h_3 (not used in the above equation) are the grid sizes of the three spatial directions respectively. Similarly we can define finite difference operators $\mathcal{L}_{13}$ and $\mathcal{L}_{23}$. The diffusion terms in the equation are approximated by $F_d(\vec{U}) = C\vec{U} = (\mathcal{L}_{12} + \mathcal{L}_{13} + \mathcal{L}_{23})U$. To derive the array representation of the operator $\mathcal{L}_{12}$, we fix k_3 in the three-dimensional array $U(:, :, k_3)$ which represents a $N_1 \times N_2$ matrix, and collect all these two-dimensional matrices along a vector. This leads to

$$U = \bigotimes_{1 \le k_3 \le N_3} U(:, :, k_3).$$

For constant diffusion coefficient cases, we can define a linear mapping $\mathcal{A}_{12}$, from a matrix space consisting of all $N_1 \times N_2$ matrices to itself as following

$$\begin{aligned} (\mathcal{A}_{12}M)_{i,j} &= \frac{2b_1}{4h_1h_2}(M_{i+1,j+1} + M_{i-1,j-1} - M_{i-1,j+1} - M_{i+1,j-1}) \\ &\quad + \frac{a_1}{h_1^2}(M_{i+1,j} - 2M_{i,j} + M_{i-1,j}) + \frac{c_1}{h_2^2}(M_{i,j+1} - 2M_{i,j} + M_{i,j-1}). \end{aligned} \tag{46}$$

Then, the array representation of $\mathcal{L}_{12}$ and its exponential are

$$\mathcal{L}_{12}U = \bigotimes_{1\le k_3\le N_3} \mathcal{A}_{12}U(:,:,k_3),$$
$$e^{\mathcal{L}_{12}\triangle t}U = \bigotimes_{1\le k_3\le N_3} e^{\mathcal{A}_{12}\triangle t}U(:,:,k_3).$$

Similarly, the array representations for $\mathcal{L}_{13}$ and $\mathcal{L}_{23}$ can be written in terms of $\mathcal{A}_{13}$ and $\mathcal{A}_{23}$ respectively. Note that here $\mathcal{A}_{12}$, $\mathcal{A}_{13}$, $\mathcal{A}_{23}$ and their exponentials are actually $(N_1 \cdot N_2) \times (N_1 \cdot N_2)$, $(N_1 \cdot N_3) \times (N_1 \cdot N_3)$ and $(N_2 \cdot N_3) \times (N_2 \cdot N_3)$ matrices respectively.

For schemes (42)–(44), vectors $\vec{V}_1$ and $\vec{V}_2$ are stored in three-dimensional arrays V_1 and V_2 as that for U. If $\mathcal{L}_{12}$, $\mathcal{L}_{13}$ and $\mathcal{L}_{23}$ commute with each other as the case that the diffusion coefficients are constants, application of array representations to (43) and (44) leads to direct decomposition of large matrix exponentials for C to much smaller ones. For detailed formulas in implementation the method, see the equations in (69) in "Appendix".

If $\mathcal{L}_{12}$, $\mathcal{L}_{13}$ and $\mathcal{L}_{23}$ do not commute with each other as the case that the diffusion coefficients are not constants, two modifications to the method are needed. One is that the finite difference operators $\mathcal{L}_{12}$, $\mathcal{L}_{13}$ and $\mathcal{L}_{23}$ may depend on other spatial dimensions since the diffusion coefficients can be functions of all spatial variables. For example, different index k_3 results in different finite difference operators $\mathcal{L}_{12}$ and different linear mappings $\mathcal{A}_{12}$. Hence the linear mappings are represented by $\mathcal{A}_{12}^{k_3}$, $\mathcal{A}_{13}^{k_2}$ and $\mathcal{A}_{23}^{k_1}$ in such cases. The other is that the Strang operator splitting [45] is needed to obtain a second order accuracy. By the Strang symmetric operator splitting, we have

$$e^{C\triangle t_n} = e^{(\mathcal{L}_{12}+\mathcal{L}_{13}+\mathcal{L}_{23})\triangle t_n} = e^{\frac{\triangle t_n}{2}\mathcal{L}_{12}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{13}} e^{\triangle t_n \mathcal{L}_{23}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{13}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{12}} + O(\triangle {t_n}^3). \quad (47)$$

Then array representations are applied in (43) and (44) for decomposition of large matrix exponentials of C. See the equations in (70) and (71) in "Appendix" for detailed implementation formulas.

Similarly, for a four dimensional CDR equation (1), $d = 4$, with cross-derivatives for the linear diffusion terms and periodic boundary conditions, the diffusion matrix $\mathbf{D}$ is

$$\mathbf{D} = \begin{pmatrix} a_1+a_2+a_3 & b_1 & b_2 & b_3 \\ b_1 & a_4+a_5+c_1 & b_4 & b_5 \\ b_2 & b_4 & a_6+c_2+c_4 & b_6 \\ b_3 & b_5 & b_6 & c_3+c_5+c_6 \end{pmatrix}, \quad (48)$$

where a_i, b_i and c_i, $i = 1, 2, 3, 4, 5, 6$ are constant or non-constant coefficients of the diffusion terms. The diffusion terms can be grouped into six classes for the convenience of applying the AcIIF method, i.e., $(a_1 \frac{\partial^2}{\partial {x_1}^2} + 2b_1 \frac{\partial^2}{\partial x_1 \partial x_2} + c_1 \frac{\partial^2}{\partial {x_2}^2})u$, $(a_2 \frac{\partial^2}{\partial {x_1}^2} + 2b_2 \frac{\partial^2}{\partial x_1 \partial x_3} + c_2 \frac{\partial^2}{\partial {x_3}^2})u$, $(a_3 \frac{\partial^2}{\partial {x_1}^2} + 2b_3 \frac{\partial^2}{\partial x_1 \partial x_4} + c_3 \frac{\partial^2}{\partial {x_4}^2})u$, $(a_4 \frac{\partial^2}{\partial {x_2}^2} + 2b_4 \frac{\partial^2}{\partial x_2 \partial x_3} + c_4 \frac{\partial^2}{\partial {x_3}^2})u$, $(a_5 \frac{\partial^2}{\partial {x_2}^2} + 2b_5 \frac{\partial^2}{\partial x_2 \partial x_4} + c_5 \frac{\partial^2}{\partial {x_4}^2})u$, $(a_6 \frac{\partial^2}{\partial {x_3}^2} + 2b_6 \frac{\partial^2}{\partial x_3 \partial x_4} + c_6 \frac{\partial^2}{\partial {x_4}^2})u$. We apply the second order IIF-WENO scheme (11) and obtain the same form schemes (42)–(44), but with a much larger system size. Again we can apply the array representation approach in computations of the matrix exponentials. Numerical solutions for u are stored in a four-dimensional array U with size $N_1 \times N_2 \times N_3 \times N_4$, where N_1, N_2, N_3 and N_4 are numbers of grid points of four spatial directions respectively. We use $\mathcal{L}_{12}$ to denote the second order central finite difference approximation of $(a_1 \frac{\partial^2}{\partial {x_1}^2} + 2b_1 \frac{\partial^2}{\partial x_1 \partial x_2} + c_1 \frac{\partial^2}{\partial {x_2}^2})$ as

$$
\begin{aligned}
(\mathcal{L}_{12}U)_{k_1,k_2,k_3,k_4} = & \frac{a_1}{h_1^2}(U_{k_1+1,k_2,k_3,k_4} - 2U_{k_1,k_2,k_3,k_4} + U_{k_1-1,k_2,k_3,k_4}) \\
& + \frac{2b_1}{4h_1h_2}(U_{k_1+1,k_2+1,k_3,k_4} + U_{k_1-1,k_2-1,k_3,k_4} - U_{k_1+1,k_2-1,k_3,k_4} \\
& - U_{k_1-1,k_2+1,k_3,k_4}) + \frac{c_1}{h_2^2}(U_{k_1,k_2+1,k_3,k_4} - 2U_{k_1,k_2,k_3,k_4} + U_{k_1,k_2-1,k_3,k_4}).
\end{aligned} \tag{49}
$$

Similarly $\mathcal{L}_{13}$, $\mathcal{L}_{14}$, $\mathcal{L}_{23}$, $\mathcal{L}_{24}$ and $\mathcal{L}_{34}$ are defined. Then the diffusion terms in the equation are approximated by $F_d(\vec{U}) = C\vec{U} = (\mathcal{L}_{12} + \mathcal{L}_{13} + \mathcal{L}_{14} + \mathcal{L}_{23} + \mathcal{L}_{24} + \mathcal{L}_{34})U$. To derive the array representation of the operator $\mathcal{L}_{12}$, we fix k_3 and k_4 in the four-dimensional array $U(:,:,k_3,k_4)$ which represents a $N_1 \times N_2$ matrix, and collect all these two-dimensional matrices along a vector to obtain

$$
U = \bigotimes_{\substack{1\le k_3\le N_3 \\ 1\le k_4\le N_4}} U(:,:,k_3,k_4).
$$

The same linear mapping $\mathcal{A}_{12}$ is defined as (46) for three dimensional cases. The array representation of $\mathcal{L}_{12}$ and its exponential are

$$
\begin{aligned}
\mathcal{L}_{12}U &= \bigotimes_{\substack{1\le k_3\le N_3 \\ 1\le k_4\le N_4}} \mathcal{A}_{12}U(:,:,k_3,k_4), \\
e^{\mathcal{L}_{12}\triangle t}U &= \bigotimes_{\substack{1\le k_3\le N_3 \\ 1\le k_4\le N_4}} e^{\mathcal{A}_{12}\triangle t}U(:,:,k_3,k_4).
\end{aligned}
$$

Similarly, the array representation for $\mathcal{L}_{13}$, $\mathcal{L}_{14}$, $\mathcal{L}_{23}$, $\mathcal{L}_{24}$ and $\mathcal{L}_{34}$ can be written in terms of $\mathcal{A}_{13}$, $\mathcal{A}_{14}$, $\mathcal{A}_{23}$, $\mathcal{A}_{24}$ and $\mathcal{A}_{34}$ respectively.

For schemes (42)–(44), vectors $\vec{V}_1$ and $\vec{V}_2$ are stored in four-dimensional arrays V_1 and V_2 as that for U. If $\mathcal{L}_{12}$, $\mathcal{L}_{13}$, $\mathcal{L}_{14}$, $\mathcal{L}_{23}$, $\mathcal{L}_{24}$ and $\mathcal{L}_{34}$ commute with each other, array representation is applied in schemes (42)–(44) to decompose large matrix exponentials for C to much smaller ones. For detailed formulas in implementation of the method, see the Eqs. (72) and (73) in "Appendix".

If $\mathcal{L}_{12}$, $\mathcal{L}_{13}$, $\mathcal{L}_{14}$, $\mathcal{L}_{23}$, $\mathcal{L}_{24}$ and $\mathcal{L}_{34}$ do not commute with each other(e.g., the case that the diffusion coefficients are not constants), again two modifications are needed in the method. One is that the linear mappings may depend on other spatial dimensions since the diffusion coefficients can be functions of all spatial variables. For example, different indexes k_3, k_4 result in different finite difference operators $\mathcal{L}_{12}$ and different linear mappings $\mathcal{A}_{12}$. Hence the linear mappings are represented by $\mathcal{A}_{12}^{k_3,k_4}$, $\mathcal{A}_{13}^{k_2,k_4}$, $\mathcal{A}_{14}^{k_2,k_3}$, $\mathcal{A}_{23}^{k_1,k_4}$, $\mathcal{A}_{24}^{k_1,k_3}$ and $\mathcal{A}_{34}^{k_1,k_2}$ in such cases. The other is that again the Strang symmetric operator splitting is needed to achieve a second order accuracy. Namely, we have

$$
\begin{aligned}
e^{C\triangle t_n} &= e^{(\mathcal{L}_{12}+\mathcal{L}_{13}+\mathcal{L}_{14}+\mathcal{L}_{23}+\mathcal{L}_{24}+\mathcal{L}_{34})\triangle t_n} \\
&= e^{\frac{\triangle t_n}{2}\mathcal{L}_{34}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{24}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{23}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{14}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{13}} e^{\triangle t_n\mathcal{L}_{12}} \\
&\quad e^{\frac{\triangle t_n}{2}\mathcal{L}_{13}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{14}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{23}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{24}} e^{\frac{\triangle t_n}{2}\mathcal{L}_{34}} + O(\triangle t_n^3).
\end{aligned} \tag{50}
$$

Then application of array representation in (43) and (44) leads to decomposition of large matrix exponentials of C into much smaller ones. See the Eqs. (74)–(77) in "Appendix" for detailed implementation formulas.

Remark All linear mappings (i.e., $\mathcal{A}_{12}$, $\mathcal{A}_{13}$, etc) here are actually $N^2 \times N^2$ matrices if all spatial directions have the same number of grid points N. Although matrix exponentials in any higher dimensional problems can be reduced to computations of such $N^2 \times N^2$ matrices' exponentials, it is still expensive to directly calculate them as shown in the following numerical experiments. Applications of Krylov subspace approximations of Sect. 2.2.1 in computations of these $N^2 \times N^2$ matrices' exponentials are still necessary for the efficiency of the AcIIF-WENO method for high dimensional CDR problems.

Remark An advantage of cIIF/AcIIF schemes is that they have simpler formulations than the Krylov IIF schemes, hence easier to code the algorithms. For multidimensional CDR or reaction–diffusion problems whose diffusion terms do *not* have cross-derivatives, cIIF/AcIIF schemes can be directly applied because we only need to compute matrix exponentials for matrices with sizes of one-dimensional problems, i.e. $N \times N$ matrices with N the number of grid points in one spatial direction. Such matrix exponentials are computed using a scaling and squaring algorithm with a Padé approximation. They are computed and stored before the time evolution, and directly used at every time step [38]. As that shown in the numerical experiments of the Sect. 3, the cIIF/AcIIF schemes implemented this way are more efficient than the Krylov IIF schemes for problems which do *not* have cross-derivative diffusion terms, on not very refined meshes.

3 Numerical Experiments

In this section, we use different types of numerical examples to systematically compare the computational efficiency of two different approaches in using integration factor methods for solving high dimensional problems. Examples include equations with analytical solutions, convection-dominated equation, a stiff reaction problem from mathematical modeling of the dorsal-ventral patterning in *Drosophila* embryos, and three dimensional and four dimensional Fokker–Planck equations. We test the convergence and CPU times, and analyze computational complexity of numerical schemes via mesh refinement studies. We perform simulations on different meshes including very fine ones. Computations on fine meshes are needed to resolve small structures in complicated solutions which often arise in application problems. Comparisons of computational efficiency by different methods on very fine meshes in this paper can provide certain guidance in choosing the suitable numerical methods. All of the numerical simulations in this paper are performed on a 2.3 GHz, 16GB RAM Linux workstation.

3.1 Diffusion Problems

We first test problems without convection, i.e., study computational complexity of both approaches without considering the cost of WENO scheme. Then the complete convection–diffusion problems are tested in the next subsection.

3.1.1 Diffusion Problems Without Cross-Derivatives

Example 1 (*A problem with linear reaction*). We consider a reaction–diffusion problem with linear reaction

$$\frac{\partial u}{\partial t} = 0.2\nabla \cdot (\nabla u) + 0.1u.$$

First we test the two dimensional case defined on the domain $\Omega = \{0 < x < 2\pi, 0 < y < 2\pi\}$, subject to no-flux boundary conditions at $x = 0, x = 2\pi$ and periodic boundary conditions in the y-direction, i.e.,

$$\frac{\partial u}{\partial x}(0, y, t) = \frac{\partial u}{\partial x}(2\pi, y, t) = 0; \quad u(x, 0, t) = u(x, 2\pi, t).$$

The initial condition is $u(x, y, 0) = \cos(x) + \sin(y)$. The exact solution of the problem is $u(x, y, t) = e^{-0.1t}(\cos(x) + \sin(y))$. We compute the problem until the final time $T = 1$ by the second order cIIF/AcIIF scheme (29) or (39) (they are equivalent), and the second order Krylov IIF scheme (17) with the convection term $F_a = 0$. Since the problem has a linear reaction term, the local implicit equation is just a linear equation and can be solved directly. We test the L^∞ errors, numerical accuracy orders and CPU times on successively refined meshes to compare the two approaches. The total numbers of multiplication and division operations at one time step are counted. The cIIF2 method needs $2N^3 + 8N^2 + 6N$ operations, where N is the number of grid points in each spatial direction. The KrylovIIF2 method for this problem needs $(M^2+12M+7)N^2+(M^2+20M+7)N+O(M^3)$ operations at every time step. M is the dimension of Krylov subspace. $M = 25$ for all examples in this paper, and M does *not* need to be increased when the spatial-temporal resolution is refined. Here $O(M^3)$ term is the number of operations for computing matrix exponential of a small $M \times M$ matrix such as $e^{H_M \Delta t}$. Since it is a small constant which is independent of N, we omit it. Hence for $M = 25$, the number of operations at one time step for the KrylovIIF2 scheme is estimated to be $932N^2 + 1132N$. This is a two dimensional problem with N^2 grid points. So the KrylovIIF2 scheme has a linear computational complexity, while the computational complexity of the cIIF2 scheme is not linear. However, their computational efficiency depends on the size of the problem. The numerical errors, accuracy orders, CPU times (time unit: second) for a complete simulation, for time evolution part and for one time step are listed in Tables 1 and 2 for the cIIF2 scheme and the KrylovIIF2 scheme. We also list the ratios of corresponding CPU times on an $N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2}$ mesh, to study the computational complexity of these two approaches. Both methods give the same numerical errors and the second order accuracy. For this two dimensional time dependent parabolic problem, we achieve large time step size computation $\Delta t = 0.5h$ by using the IIF method. A linear computational complexity method should have the CPU time ratio be 8 for a complete time evolution and the ratio 4 for one time step. The KrylovIIF2 scheme's CPU time ratios shown in Table 2 verify its linear computational complexity. On the other hand, although the cIIF2 scheme's CPU time ratios shown in Table 1 are not linear, the cIIF2 scheme is more efficient than the KrylovIIF2 scheme on 40×40, 80×80 and 160×160 meshes, because the cIIF2 scheme has a much smaller coefficient 2 in its leading operation amount than the KrylovIIF2 whose leading operation amount coefficient is 932. On more refined meshes 640×640 and 1280×1280, the KrylovIIF2 scheme is more efficient than the cIIF2. On 320×320 mesh, the cIIF2 scheme is more efficient than the KrylovIIF2 scheme for one time step, but KrylovIIF2 is more efficient for the complete simulation and for the whole time evolution. This is because that cIIF schemes compute matrix exponentials (e.g., matrix exponentials for $N \times N$ matrices $A\Delta t$ and $B\Delta t$) before the time evolution and at the last time step when Δt changes to reach the final time T. So additional CPU times are needed. Other strategies to improve computational efficiency can be explored further here, for example, interpolation in time for the last time step rather than recomputing matrix exponentials. This will be one of our future work.

We perform the same test for third order schemes. The third order cIIF scheme cIIF3 (the scheme (27) with $r = 3$) and the third order KrylovIIF scheme KrylovIIF3 (18) are used

Table 1 Example 1: 2D case, cIIF2 scheme, $\Delta t = 0.5h$, final time $T = 1.0$

$N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
40×40	7.45×10^{-4}		0.13		0.09		0.0031	
80×80	1.86×10^{-4}	2.00	1.43	11.06	1.04	12.21	0.025	7.92
160×160	4.66×10^{-5}	2.00	18.26	12.73	14.21	13.66	0.20	8.02
320×320	1.16×10^{-5}	2.00	269.66	14.77	225.03	15.84	1.77	8.88
640×640	2.91×10^{-6}	2.00	4,667.67	17.31	4,328.65	19.24	19.58	11.07
1280×1280	7.28×10^{-7}	2.00	79,855.09	17.11	76,837.65	17.75	180.42	9.22

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2}$ mesh

Table 2 Example 1: 2D case, KrylovIIF2 scheme, $\Delta t = 0.5h$, final time $T = 1.0$

$N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
40×40	7.45×10^{-4}		0.50		0.50		0.04	
80×80	1.86×10^{-4}	2.00	3.56	7.16	3.56	7.16	0.14	3.58
160×160	4.66×10^{-5}	2.00	27.34	7.68	27.34	7.68	0.54	3.92
320×320	1.16×10^{-5}	2.00	219.15	8.02	219.15	8.02	2.15	4.01
640×640	2.91×10^{-6}	2.00	1,828.21	8.34	1,828.21	8.34	8.91	4.15
1280×1280	7.28×10^{-7}	2.00	14,174.02	7.75	14,174.02	7.75	34.66	3.89

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2}$ mesh

to compute the same two-dimensional problem until the final time $T = 1$. The comparison results are presented in Tables 3 and 4. Both methods have comparable numerical errors and accuracy orders. We observe higher than third order (around fourth order) numerical accuracy orders because we used a fourth order central difference scheme to discretize the diffusion terms. This is for the purpose of having comparable spatial and temporal numerical errors. Again as that in the second order schemes, the Krylov IIF scheme KrylovIIF3 shows a linear computational complexity, while the cIIF scheme cIIF3 does not. However, cIIF3 is more efficient than KrylovIIF3 on not very refined meshes such as 40×40, 80×80, 160×160 and 320×320. On very refined meshes 640×640 and 1280×1280, KrylovIIF3 is more efficient.

Then we test the three dimensional case defined on the domain $\Omega = \{0 \le x \le \pi, 0 \le y \le \pi, 0 \le z \le \pi\}$, subject to no-flux boundary conditions. The initial condition is $u(x, y, z, 0) = \cos(x) + \cos(y) + \cos(z)$. The exact solution is $u(x, y, z, t) = e^{-0.1t}(\cos(x) + \cos(y) + \cos(z))$. We count the total numbers of multiplication and division operations at one time step. The cIIF2 scheme needs $3N^4 + 4N^3$ operations, while the KrylovIIF2 scheme requires $(M^2 + 8M + 6)N^3 + 12MN^2 + O(M^3)$ operations. N is the number of grid points in each spatial direction. Again M is the dimension of the Krylov subspace and $M = 25$. $O(M^3)$ term is the number of operations for computing matrix exponential of a small $M \times M$ matrix such as $e^{H_M \Delta t}$. Since it is a small constant which is independent of N, we omit it. Hence for

Table 3 Example 1: 2D case, cIIF3 scheme, $\Delta t = 0.5h$, final time $T = 1.0$

$N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
40×40	1.47×10^{-5}		0.21		0.16		0.01	
80×80	9.18×10^{-7}	4.00	2.43	11.37	1.96	12.31	0.05	7.82
160×160	5.74×10^{-8}	4.00	34.41	14.18	30.38	15.49	0.49	9.44
320×320	3.59×10^{-9}	4.00	433.57	12.60	397.46	13.08	3.41	7.01
640×640	2.29×10^{-10}	3.97	7,782.29	17.95	7,385.89	18.58	33.51	9.83
1280×1280	2.89×10^{-11}	2.99	145,987.45	18.76	141,798.99	19.20	332.66	9.93

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2}$ mesh

Table 4 Example 1: 2D case, KrylovIIF3 scheme, $\Delta t = 0.5h$, final time $T = 1.0$

$N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
40×40	1.47×10^{-5}		1.13		1.12		0.09	
80×80	9.18×10^{-7}	4.00	7.45	6.60	7.39	6.59	0.28	3.22
160×160	5.74×10^{-8}	4.00	62.08	8.33	61.58	8.34	1.21	4.37
320×320	3.59×10^{-9}	4.00	504.81	8.13	500.40	8.13	4.90	4.06
640×640	2.35×10^{-10}	3.94	3,743.59	7.42	3,696.45	7.39	17.63	3.60
1280×1280	1.25×10^{-11}	4.23	33,080.77	8.84	32,580.07	8.81	80.09	4.54

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2}$ mesh

$M = 25$, the number of operations at one time step for the KrylovIIF2 scheme is estimated to be $831N^3 + 300N^2$. Since three dimensional problem has N^3 grid points, the computational complexity of the KrylovIIF2 scheme is linear, while the computational complexity of the cIIF2 scheme is not linear. Again as that for the two dimensional problem, their computational efficiency depends on the size of the problem. We compute the problem until the final time $T = 1$. The numerical errors, accuracy orders, CPU times for a complete simulation, for time evolution part and for one time step, and the ratios of corresponding CPU times on an $N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2}$ mesh are listed in Tables 5 and 6 for the cIIF2 scheme and the KrylovIIF2 scheme. Both methods give the same numerical errors and the second order accuracy. For a three dimensional time dependent problem with $\Delta t = h/3$, a linear computational complexity method should have the CPU time ratio be 16 for a complete time evolution and the ratio 8 for one time step. The KrylovIIF2 scheme's CPU time ratios shown in Table 6 verify its linear computational complexity. However, the cIIF2 scheme is more efficient than KrylovIIF2 scheme on $10 \times 10 \times 10$, $20 \times 20 \times 20$, $40 \times 40 \times 40$, $80 \times 80 \times 80$, and $160 \times 160 \times 160$ meshes, because the cIIF2 scheme has a much smaller coefficient 3 in its leading operation amount than the KrylovIIF2 whose leading operation amount coefficient is 831. On the most refined mesh $320 \times 320 \times 320$, the KrylovIIF2 scheme is more efficient than the cIIF2. We can also see that the cIIF2 scheme needs slightly additional CPU times

Table 5 Example 1: 3D case, cIIF2 scheme, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	2.24×10^{-3}		0.0061		0.0057		0.00054	
$20 \times 20 \times 20$	5.79×10^{-4}	1.95	0.21	34.99	0.21	36.76	0.010	19.12
$40 \times 40 \times 40$	1.87×10^{-4}	1.63	6.93	32.67	6.90	32.96	0.18	17.05
$80 \times 80 \times 80$	5.50×10^{-5}	1.77	230.83	33.33	230.60	33.42	2.99	16.94
$160 \times 160 \times 160$	1.53×10^{-5}	1.85	8,792.19	38.09	8,790.15	38.12	55.13	18.42
$320 \times 320 \times 320$	4.06×10^{-6}	1.91	367,739.27	41.83	367,712.22	41.83	1242.62	22.54

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

Table 6 Example 1: 3D case, KrylovIIF2 scheme, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	2.24×10^{-3}		0.22		0.22		0.02	
$20 \times 20 \times 20$	5.79×10^{-4}	1.95	3.06	14.15	3.06	14.15	0.15	7.02
$40 \times 40 \times 40$	1.87×10^{-4}	1.63	50.54	16.49	50.54	16.49	1.30	8.51
$80 \times 80 \times 80$	5.50×10^{-5}	1.77	850.24	16.82	850.24	16.82	11.06	8.53
$160 \times 160 \times 160$	1.53×10^{-5}	1.85	13,637.13	16.04	13,637.13	16.04	89.28	8.07
$320 \times 320 \times 320$	4.06×10^{-6}	1.91	225,543.28	16.54	225,543.28	16.54	735.62	8.24

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

to compute a few $N \times N$ matrix exponentials before the time evolution and at the last time step.

Example 2 (*A problem with nonlinear reaction*). We consider a reaction–diffusion problem with nonlinear reaction

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - u^2 + e^{-2t}\cos^2(\pi x)\cos^2(\pi y) + (2\pi^2 - 1)e^{-t}\cos(\pi x)\cos(\pi y).$$

The PDE is defined on the two dimensional domain $(x, y) \in (0, 1) \times (0, 1)$, subject to no-flux boundary conditions. The initial condition is $u(x, y, 0) = \cos(\pi x)\cos(\pi y)$. The exact solution of the problem is $u(x, y, t) = e^{-t}\cos(\pi x)\cos(\pi y)$. We compute the problem until the final time $T = 1$ by the cIIF2 scheme and the KrylovIIF2 scheme. Again we test the L^∞ errors, numerical accuracy orders and CPU times on successively refined meshes to compare the two approaches for such a nonlinear reaction–diffusion problem. In the cIIF2 scheme and the KrylovIIF2 scheme, a local nonlinear equation needed to be solved at every grid point, due to the implicit treatment for the reaction term. Here the local nonlinear equation is solved by fixed-point iterations as that in [37]. The results are reported in Tables 7 and 8. We can see that both methods give the second order accuracy and they have comparable numerical errors, while KrylovIIF2 has smaller numerical errors on refined meshes 640×640 and 1280×1280. The ratios of corresponding CPU times on an $N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2}$

Table 7 Example 2: cIIF2 scheme, $\Delta t = 0.5h$, final time $T = 1.0$

$N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
40×40	2.81×10^{-3}		0.56		0.49		0.0062	
80×80	7.19×10^{-4}	1.97	6.35	11.30	5.73	11.67	0.036	5.78
160×160	1.82×10^{-4}	1.98	82.36	12.97	76.56	13.35	0.24	6.61
320×320	4.56×10^{-5}	1.99	1,202.63	14.60	1,146.50	14.98	1.80	7.56
640×640	1.14×10^{-5}	2.00	18,055.74	15.01	17,598.19	15.35	13.72	7.63
1280×1280	2.86×10^{-6}	2.00	375,400.69	20.79	371,035.11	21.08	142.81	10.41

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2}$ mesh

mesh show that the KrylovIIF2 scheme has a linear computational complexity. Similar as the last example, the cIIF2 scheme is more efficient than the KrylovIIF2 scheme on meshes 40×40, 80×80, 160×160 and 320×320. On more refined meshes 640×640 and 1280×1280, the KrylovIIF2 scheme is more efficient than the cIIF2 scheme.

3.1.2 Diffusion Problems with Cross-Derivatives

Example 3 (*A 3D problem with constant diffusion coefficients*). We consider a three-dimensional reaction–diffusion problem with constant diffusion coefficients

$$\begin{aligned} u_t = &(0.1u_{xx} - 0.15u_{xy} + 0.1u_{yy}) + (0.1u_{xx} + 0.2u_{xz} + 0.2u_{zz}) \\ &+ (0.2u_{yy} + 0.15u_{yz} + 0.1u_{zz}) + 0.8u, \end{aligned}$$

where $(x, y, z) \in \Omega = \{0 < x < 2\pi, 0 < y < 2\pi, 0 < z < 2\pi\}$ with periodic boundary conditions. The initial condition is $u(x, y, z, 0) = \sin(x + y + z)$. The exact solution of the problem is

$$u(x, y, z, t) = e^{-0.2t} \sin(x + y + z).$$

This problem was used in [49] for testing the AcIIF2 scheme. We compute the problem until the final time $T = 1$ by the KrylovIIF2 scheme (17) with the convection term $F_a = 0$, and the AcIIF2 scheme (42), (69) with the convection term $F_a = 0$. For the AcIIF2 scheme, we implement it in two different ways. One way is to directly compute the matrix exponentials in (69). As that shown in the following numerical results, it is still very expensive in terms of both CPU times and computer memory to directly calculate such $N^2 \times N^2$ matrices' exponentials. A more efficient way to implement AcIIF schemes is to apply Krylov subspace approximations of Sect. 2.2.1 in computations of these $N^2 \times N^2$ matrices' exponentials. We call such method AcIIF schemes with Krylov subspace approximations. Again we test the L^∞ errors, numerical accuracy orders and CPU times on successively refined meshes to compare the KrylovIIF2 scheme, the direct AcIIF2 scheme, and the AcIIF2 scheme with Krylov subspace approximations for this problem. The results are reported in Tables 9, 10 and 11. We can see that all of methods give the same numerical errors and the second order accuracy. However, the direct AcIIF2 scheme is computationally expensive as shown in Table 10, in both CPU times and computer memory costs. The significant CPU time and computer memory costs for the direct AcIIF2 scheme come from the direct computations

Table 8 Example 2: KrylovIIF2 scheme, $\Delta t = 0.5h$, final time $T = 1.0$

$N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
40×40	2.81×10^{-3}		3.85		3.85		0.05	
80×80	7.19×10^{-4}	1.97	26.50	6.88	26.50	6.88	0.17	3.51
160×160	1.81×10^{-4}	1.99	198.52	7.49	198.52	7.49	0.61	3.63
320×320	4.45×10^{-5}	2.03	1,621.66	8.17	1,621.66	8.17	2.54	4.13
640×640	7.65×10^{-6}	2.54	12,822.76	7.91	12,822.76	7.91	9.92	3.91
1280×1280	1.90×10^{-6}	2.01	104,679.46	8.16	104,679.46	8.16	40.07	4.04

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2}$ mesh

Table 9 Example 3: KrylovIIF2 scheme, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	4.21×10^{-2}		0.15		0.15		0.03	
$20 \times 20 \times 20$	1.11×10^{-2}	1.92	2.09	13.51	2.08	13.52	0.21	6.76
$40 \times 40 \times 40$	2.79×10^{-3}	2.00	33.11	15.88	33.09	15.89	1.65	7.95
$80 \times 80 \times 80$	6.97×10^{-4}	2.00	538.81	16.27	538.70	16.28	13.69	8.27
$160 \times 160 \times 160$	1.74×10^{-4}	2.00	8,413.74	15.62	8,412.93	15.62	109.56	8.00
$320 \times 320 \times 320$	4.36×10^{-5}	2.00	132,359.95	15.73	132,353.57	15.73	866.21	7.91

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

and stores of several $N^2 \times N^2$ matrices' exponentials. In fact, the computations on the $160 \times 160 \times 160$ mesh can *not* be performed due to memory restrictions of our workstation. Direct large $N^2 \times N^2$ matrix-vector multiplications require a large amount of CPU time for refined meshes as shown in Table 10 the one time step CPU times. On the other hand, if we use the Krylov approach to approximate these $N^2 \times N^2$ matrices' exponentials in the AcIIF2 scheme, the computational efficiency can be improved dramatically. This is shown in Table 11. An interesting case is that for the coarse meshes such as $10 \times 10 \times 10$ and $20 \times 20 \times 20$, the one time step CPU time for the direct AcIIF2 scheme is less than that for the AcIIF2 scheme with Krylov subspace approximations due to the relative small sizes of $N^2 \times N^2$ matrix-vector multiplications. However, the total CPU time for the direct AcIIF2 scheme still costs more due to the expensive direct evaluations of $N^2 \times N^2$ matrices' exponentials. In Table 9, we report results for the KrylovIIF2 scheme. The efficiency of the KrylovIIF2 scheme is impressive. In fact, the KrylovIIF2 scheme is the most efficient one among all three approaches here on all meshes. We can also see that both the KrylovIIF2 scheme and the AcIIF2 scheme with Krylov subspace approximations have linear computational complexity as shown by the CPU time ratios in Tables 9 and 11.

Example 4 (*A 4D problem with constant diffusion coefficients*). We test a higher dimensional problem, the four-dimensional reaction–diffusion problem with constant diffusion coefficients

Table 10 Example 3: Direct AcIIF2 scheme, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	4.21×10^{-2}		2.16		1.08		0.01	
$20 \times 20 \times 20$	1.11×10^{-2}	1.92	143.85	66.60	73.36	67.75	0.28	31.73
$40 \times 40 \times 40$	2.79×10^{-3}	2.00	11,831.05	82.24	5,214.92	71.09	8.88	32.26
$80 \times 80 \times 80$	6.97×10^{-4}	2.00	1,601,309.44	135.35	753,295.70	144.45	485.98	54.73
$160 \times 160 \times 160$	–	–	–	–	–	–	–	–

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh. The symbol "–" means no enough memory for computation

Table 11 Example 3: AcIIF2 scheme with Krylov subspace approximations, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	4.21×10^{-2}		1.17		1.17		0.23	
$20 \times 20 \times 20$	1.11×10^{-2}	1.92	8.66	7.41	8.66	7.41	0.87	3.70
$40 \times 40 \times 40$	2.79×10^{-3}	2.00	96.87	11.18	96.86	11.18	4.85	5.59
$80 \times 80 \times 80$	6.97×10^{-4}	2.00	1,352.48	13.96	1,352.37	13.96	34.70	7.15
$160 \times 160 \times 160$	1.74×10^{-4}	2.00	21,221.14	15.69	21,220.33	15.69	275.57	7.94
$320 \times 320 \times 320$	4.36×10^{-5}	2.00	339,245.16	15.99	339,238.81	15.99	2,217.32	8.05

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

Table 12 Example 4: KrylovIIF2 scheme, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10 \times 10$	1.16×10^{-1}		1.60		1.59		0.32	
$20 \times 20 \times 20 \times 20$	2.92×10^{-2}	1.99	49.34	30.89	49.30	30.92	4.93	15.49
$40 \times 40 \times 40 \times 40$	7.24×10^{-3}	2.01	1,596.13	32.35	1,595.56	32.37	79.79	16.19
$80 \times 80 \times 80 \times 80$	1.81×10^{-3}	2.00	70,569.13	44.21	70,560.68	44.22	1,929.45	24.18

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

$$\begin{aligned} u_t = {} & (0.1u_{x_1x_1} - 0.15u_{x_1x_2} + 0.1u_{x_2x_2}) + (0.1u_{x_1x_1} + 0.2u_{x_1x_3} + 0.2u_{x_3x_3}) \\ & + (0.1u_{x_1x_1} + 0.2u_{x_1x_4} + 0.2u_{x_4x_4}) + (0.1u_{x_2x_2} + 0.2u_{x_2x_3} + 0.2u_{x_3x_3}) \\ & + (0.1u_{x_2x_2} + 0.2u_{x_2x_4} + 0.2u_{x_4x_4}) + (0.2u_{x_3x_3} + 0.15u_{x_3x_4} + 0.1u_{x_4x_4}) + 2u, \end{aligned} \tag{51}$$

where $(x_1, x_2, x_3, x_4) \in \Omega = \{0 < x_1 < 2\pi, 0 < x_2 < 2\pi, 0 < x_3 < 2\pi, 0 < x_4 < 2\pi\}$ with periodic boundary condition. The initial condition is $u(x_1, x_2, x_3, x_4, 0) = \sin(x_1 + x_2 + x_3 + x_4)$. The exact solution of the problem is

$$u(x_1, x_2, x_3, x_4, t) = e^{-0.5t} \sin(x_1 + x_2 + x_3 + x_4).$$

Table 13 Example 4: Direct AcIIF2 scheme, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10 \times 10$	1.16×10^{-1}		5.20		3.04		0.19	
$20 \times 20 \times 20 \times 20$	2.92×10^{-2}	1.99	398.91	76.75	258.66	85.06	11.84	63.91
$40 \times 40 \times 40 \times 40$	7.24×10^{-3}	2.01	38,341.37	96.12	25,777.97	99.66	799.41	67.50
$80 \times 80 \times 80 \times 80$	–	–	–	–	–	–	–	–

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh. On the $80 \times 80 \times 80 \times 80$ mesh, the computations can *not* be performed due to computation time restrictions of our workstation

Table 14 Example 4: AcIIF2 scheme with Krylov subspace approximations, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10 \times 10$	1.16×10^{-1}		23.70		23.69		4.73	
$20 \times 20 \times 20 \times 20$	2.92×10^{-2}	1.99	346.17	14.61	346.13	14.61	34.59	7.31
$40 \times 40 \times 40 \times 40$	7.24×10^{-3}	2.01	7,779.73	22.47	7,779.17	22.47	389.45	11.26
$80 \times 80 \times 80 \times 80$	1.81×10^{-3}	2.00	217,356.07	27.94	217,347.68	27.94	5,573.58	14.31

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

We compute the problem until the final time $T = 1$ by the KrylovIIF2 scheme (17) with the convection term $F_a = 0$, and the AcIIF2 scheme (42), (72), (73) with the convection term $F_a = 0$. For the AcIIF2 scheme, we also implement it in two different ways, i.e., the direct computations of $N^2 \times N^2$ matrices' exponentials and the Krylov subspace approximations of them. The numerical results are reported in Tables 12, 13 and 14. We obtain the same conclusion as the 3D problem (Example 3). All of methods give the same numerical errors and the second order accuracy. However, the direct AcIIF2 scheme is computationally the most expensive one among three approaches for relatively refined meshes such as $40 \times 40 \times 40 \times 40$. We count the total numbers of multiplication and division operations at one time step. The direct AcIIF2 scheme needs $6N^6 + 2N^4$ operations, where N is the number of grid points in each spatial direction. The computational complexity is *not* linear and CPU time ratio is expected to be around $2^6 = 64$ when the spatial mesh is refined once. This is verified in Table 13. As a result of the significant increase of computation time with mesh refinement, CPU time has reached the maximum computation time restriction of our workstation and the computation on $80 \times 80 \times 80 \times 80$ can *not* be performed. The computational efficiency is improved dramatically when the Krylov approach is used to approximate these $N^2 \times N^2$ matrices' exponentials in the AcIIF2 scheme, as shown in Table 14. Again, the KrylovIIF2 scheme is the most efficient one among all three approaches here on all meshes as shown in Table 12. In terms of total numbers of multiplication and division operations at one time step, the KrylovIIF2 scheme needs $(M^2 + 28M + 4)N^4 + O(M^3)$ operations, and the AcIIF2 scheme with Krylov subspace approximations needs $(6M^2 + 66M + 14)N^4 + O(N^2)$ operations. M is the dimension of the Krylov subspace and $M = 25$ in this example. Hence they have linear computational complexity.

Table 15 Example 5: KrylovIIF2 scheme, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	2.15×10^{-1}		0.17		0.17		0.03	
$20 \times 20 \times 20$	5.29×10^{-2}	2.02	2.20	13.06	2.19	13.10	0.22	6.56
$40 \times 40 \times 40$	1.34×10^{-2}	1.99	35.05	15.94	35.00	15.98	1.75	7.89
$80 \times 80 \times 80$	3.34×10^{-3}	2.00	551.57	15.73	551.17	15.75	14.13	8.07
$160 \times 160 \times 160$	8.34×10^{-4}	2.00	8,992.13	16.30	8,989,12	16.31	115.99	8.21
$320 \times 320 \times 320$	2.09×10^{-4}	2.00	153,195.14	17.04	153,171.55	17.04	958.75	8.27

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

Table 16 Example 5: Direct AcIIF2 scheme, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	2.12×10^{-1}		13.79		6.77		0.02	
$20 \times 20 \times 20$	5.19×10^{-2}	2.03	1,723.95	125.01	852.12	125.81	0.54	27.17
$40 \times 40 \times 40$	1.31×10^{-2}	1.99	328,908.44	190.79	145,345.87	170.57	20.18	37.45
$80 \times 80 \times 80$	–	–	–	–	–	–	–	–

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh. The symbol "–" means no enough memory for computation

Example 5 (*A 3D problem with variable diffusion coefficients*). In this example, we test the three-dimensional reaction–diffusion problem with variable diffusion coefficients

$$\begin{aligned} u_t &= 0.5u_{xx} - 0.5\sin(x+y)u_{xy} + 0.5u_{yy} \\ &\quad + 0.5u_{xx} - \frac{1}{3}\cos y u_{xz} + \frac{1}{3}u_{zz} \\ &\quad + 0.5(1+\cos x)u_{yy} - 0.5(1+\cos x)u_{yz} + \frac{1}{3}(1+\cos x)u_{zz} + f(x,y,z,u), \end{aligned} \tag{52}$$

where $(x, y, z) \in \Omega = \{0 < x < 2\pi, 0 < y < 2\pi, 0 < z < 2\pi\}$ with periodic boundary conditions. The initial condition is $u(x, y, z, 0) = \sin(x + y + z)$. The source term $f(x, y, z, u) = \left(1.3 + \frac{2}{3} - 0.5\sin(x + y) + \frac{1}{3}(\cos x - \cos y)\right)u$. The exact solution of this problem is

$$u(x, y, z, t) = e^{-0.2t}\sin(x + y + z).$$

This problem was used in [49] for testing the AcIIF2 scheme. We compute the problem until the final time $T = 1$. The KrylovIIF2 scheme (17) with the convection term $F_a = 0$, and the AcIIF2 scheme (42), (70), (71) with the convection term $F_a = 0$ are tested. Two different ways to implement the AcIIF2 scheme, i.e., direct computations of $N^2 \times N^2$ matrices' exponentials and Krylov subspace approximations of them, are performed. The numerical results are reported in Tables 15, 16 and 17. We obtain the same conclusion as Example 3 and Example 4. All of methods achieve similar numerical errors and the second order accuracy. Again, the direct AcIIF2 scheme is computationally the most expensive one among three approaches

Table 17 Example 5: AcIIF2 scheme with Krylov subspace approximations, $\Delta t = h/3$, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	2.12×10^{-1}		1.99		1.99		0.40	
$20 \times 20 \times 20$	5.19×10^{-2}	2.03	14.87	7.45	14.86	7.45	1.49	3.73
$40 \times 40 \times 40$	1.31×10^{-2}	1.99	165.34	11.12	165.28	11.12	8.26	5.56
$80 \times 80 \times 80$	3.27×10^{-3}	2.00	2,299.09	13.91	2,298.70	13.91	58.91	7.13
$160 \times 160 \times 160$	8.17×10^{-4}	2.00	35,181.50	15.30	35,178.49	15.30	456.60	7.75
$320 \times 320 \times 320$	2.04×10^{-4}	2.00	577,577.49	16.42	577,553.96	16.42	3,775.65	8.27

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

due to direct computations of quite a few $N^2 \times N^2$ matrices' exponentials. Especially for this problem with variable diffusion coefficients, much more $N^2 \times N^2$ matrices' exponentials need to be computed than that for constant diffusion coefficient problems because such $N^2 \times N^2$ matrices $\mathcal{A}_{12}^{k_3}$, $\mathcal{A}_{13}^{k_2}$ and $\mathcal{A}_{23}^{k_1}$ in (70) and (71) are different at different spatial grid points. Since direct implementation of the AcIIF2 scheme computes and stores these $N^2 \times N^2$ matrices' exponentials before the time evolution, much more computer memory is used to store matrices' exponentials than that by the approach of Krylov subspace approximations, in which multiplications of exponential matrices and vectors are performed in the time evolution process and no matrix's exponential is pre-stored. In fact, the computations on the $80 \times 80 \times 80$ mesh by the direct AcIIF2 scheme can *not* be performed due to memory restrictions of our workstation. Table 16 shows that a complete simulation needs much more CPU times than that of the time evolution part. This verifies that direct computations of these $N^2 \times N^2$ matrices' exponentials require a large amount of CPU resources. Again, the computational efficiency can be improved dramatically by using the Krylov approach to approximate multiplications of $N^2 \times N^2$ matrices' exponentials with vectors in the AcIIF2 scheme, as shown in Table 17. And computations can be performed on much more refined meshes (Table 17) since we do not need to pre-store these $N^2 \times N^2$ matrices' exponentials. The most efficient one is the computations by using the KrylovIIF2 scheme, as shown in Table 15. In terms of total numbers of multiplication and division operations at one time step, the KrylovIIF2 scheme needs $(M^2 + 19M + 7)N^3 + MN^2 + MN + O(M^3)$ operations, and the AcIIF2 scheme needs $5N^5 + 6N^3$ operations. N is the number of grid points in each spatial direction, while the constant M is the dimension of the Krylov subspace and $M = 25$ in this example. Hence the KrylovIIF2 scheme has linear computational complexity as shown by the CPU time ratios in Table 15.

3.1.3 A System with Stiff Reactions from Mathematical Biology

Example 6 We consider an example in mathematical modeling of the dorsal-ventral patterning in *Drosophila* embryos, a regulatory system involving several zygotic genes [35]. Among them, decapentaplegic (Dpp) promotes dorsal cell fates such as amnioserosa and inhibits development of the ventral central nervous system; and another gene Sog promotes central nervous system development. In this system, Dpp is produced only in the dorsal region while Sog is produced only in the ventral region. For the wild-type, the Dpp activity has a sharp peak around the mid-line of the dorsal with the presence of its "inhibitor" Sog.

Intriguingly, mutation of Sog results in a loss of ventral structure as expected, but, in addition, the amnioserosa is reduced as well. It appears that the Dpp antagonist, Sog, is required for maximal Dpp signaling [2]. Motivated by experimental study on over-expression of the cell receptors along the anterior–posterior axis of the embryo [35], a two-dimensional reaction diffusion model was developed [27] to exam the Dpp activities outside the area of elevated receptors in a *Drosophila* embryo. The model has stiff reaction terms due to largely different biochemical reaction rates in the system [38]. Here we compare the computational efficiency of compact IIF method and Krylov IIF method for solving this example.

Let $[L]$, $[S]$, $[LS]$, $[LR]$ denote the concentration of Dpp, Sog, Dpp-Sog complex, and Dpp-receptor complex, respectively. The dynamics of the Dpp-Sog system is governed by the following reaction diffusion system [27]:

$$\begin{aligned}
\frac{\partial [L]}{\partial t} &= D_L \left(\frac{\partial^2 [L]}{\partial x^2} + \frac{\partial^2 [L]}{\partial y^2} \right) - k_{on}[L]\,(R(x,y) - [LR]) + k_{off}[LR] \\
&\quad - j_{on}[L][S] + (j_{off} + \tau j_{deg})[LS] + V_L(x,y) \\
\frac{\partial [LR]}{\partial t} &= k_{on}[L]\,(R(x,y) - [LR]) - (k_{off} + k_{deg})[LR] \\
\frac{\partial [LS]}{\partial t} &= D_{LS} \left(\frac{\partial^2 [LS]}{\partial x^2} + \frac{\partial^2 [LS]}{\partial y^2} \right) + j_{on}[L][S] - (j_{off} + j_{deg})[LS] \\
\frac{\partial [S]}{\partial t} &= D_S \left(\frac{\partial^2 [S]}{\partial x^2} + \frac{\partial^2 [S]}{\partial y^2} \right) - j_{on}[L][S] + j_{off}[LS] + V_S(x,y)
\end{aligned} \tag{53}$$

in the domain $0 < x < X_{\max}, 0 < y < Y_{\max}$, where

$$R(x,y) = \begin{cases} R_h, & x \le X_h, \\ R_0, & x > X_h. \end{cases} \tag{54}$$

$$V_L(X,Y) = \begin{cases} v_L, & y < \frac{1}{2} Y_{\max}, \\ 0, & y \ge \frac{1}{2} Y_{\max}. \end{cases} \tag{55}$$

$$V_S(X,Y) = \begin{cases} 0, & y < \frac{1}{2} Y_{\max}, \\ v_S, & y \ge \frac{1}{2} Y_{\max}. \end{cases} \tag{56}$$

The boundary conditions for $[L]$, $[LS]$, and $[S]$ are no-flux at $x = 0$ and $x = X_{\max}$, and periodic at $y = 0$ and $y = Y_{\max}$. $R(x,y)$ is the concentration of the initially available receptor in space; $x = X_h$ is the boundary between the two regions with different level of receptors; $V_L(x,y)$ and $V_S(x,y)$ are the production rates for Dpp and Sog, respectively; D_L, D_{LS}, D_S are diffusion coefficients; τ is the cleavage rate for Sog, and other coefficients are on, off and degradation rate constants for the corresponding biochemical reactions. The initial concentrations of all morphogen molecules are zeros. Both $X_{\max}$ and $Y_{\max}$ are taken to be 0.055cm, based on the embryo size of *Drosophila* at its certain developmental stage [35]. We study the cell receptor over-expression experiments in [35] by setting $R_h = 9\,\mu$M in the region $0 < x \le X_h = 0.02$cm, and $R_0 = 3\,\mu$M in the rest part of the domain. The second order Krylov IIF (Krylov IIF2) scheme and the second order compact IIF (cIIF2) scheme are used to simulate the system. The numerical solutions for the concentrations of Dpp, Dpp-receptor, Dpp-Sog and Sog are presented in Figs. 1 and 2. Similar results are obtained for these two methods. Simulations by both methods confirm that the over-expression of receptor induces a local boost of Dpppreceptor activities near the boundary of two different concentration regions of receptors, similar to the experimental observations in [35]. However

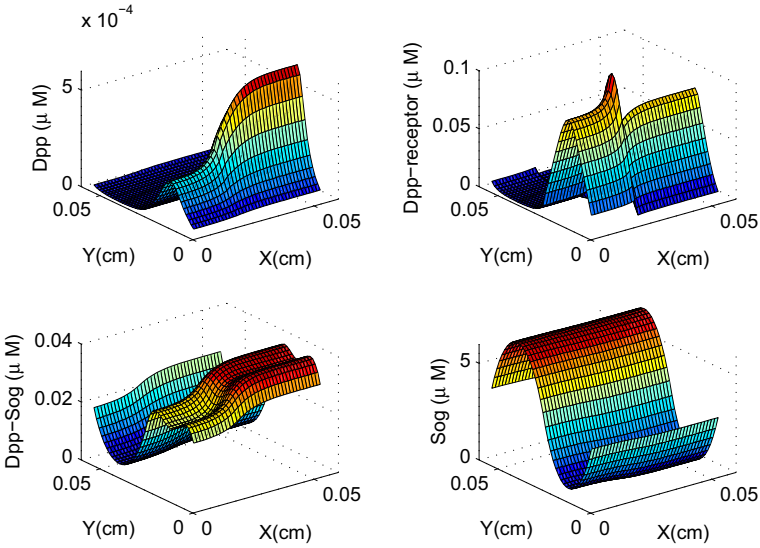

Fig. 1 Numerical solutions of Example 6 using the Krylov IIF2 scheme: concentrations of $[L]$, $[LR]$, $[LS]$, $[S]$ at $T = 100$ s for the Dpp-Sog system when receptors are over-expressed. $\Delta t = h_x = h_y = 0.001375$ in the simulation. Parameters are $D_L = D_{LS} = D_S = 85\,\mu\text{m}^2\,\text{s}^{-1}$; $v_L = 1\,\text{nM}\,\text{s}^{-1}$; $v_S = 80\,\text{nM}\,\text{s}^{-1}$; $k_{on} = 0.4\,\mu\text{M}^{-1}\,\text{s}^{-1}$; $k_{off} = 4 \times 10^{-6}\,\text{s}^{-1}$; $k_{deg} = 5 \times 10^{-4}\,\text{s}^{-1}$; $j_{on} = 95\,\mu\text{M}^{-1}\,\text{s}^{-1}$; $j_{off} = 4 \times 10^{-6}\,\text{s}^{-1}$; $j_{deg} = 0.54\,\text{s}^{-1}$; $\tau = 1$; $R_h = 9\,\mu\text{M}$; $R_0 = 3\,\mu\text{M}$

the computational efficiency of these two methods are different. It takes 871.26 seconds CPU time for the cIIF2 scheme to finish the simulation, while it costs 8152.50 seconds CPU time for the Krylov IIF2 scheme. Again, consistent observations with previous examples are obtained. For this example which has diffusion terms without cross-derivatives, compact approach is more efficient than the Krylov approach.

3.2 Convection–Diffusion Problems

In this section, we test these schemes for dealing with high dimensional convection–diffusion problems with WENO discretizations for convection terms.

Example 7 (*A 4D convection–diffusion equation with anisotropic diffusion and constant diffusion coefficients*) We consider a four-dimensional convection–diffusion equation with cross-derivative diffusion terms and constant diffusion coefficients

$$\begin{aligned} u_t &+ \left(\frac{1}{2}u^2\right)_{x_1} + \left(\frac{1}{2}u^2\right)_{x_2} + \left(\frac{1}{2}u^2\right)_{x_3} + \left(\frac{1}{2}u^2\right)_{x_4} \\ &= (0.1u_{x_1x_1} - 0.15u_{x_1x_2} + 0.1u_{x_2x_2}) + (0.1u_{x_1x_1} + 0.2u_{x_1x_3} + 0.2u_{x_3x_3}) \\ &\quad + (0.1u_{x_1x_1} + 0.2u_{x_1x_4} + 0.2u_{x_4x_4}) + (0.1u_{x_2x_2} + 0.2u_{x_2x_3} + 0.2u_{x_3x_3}) \\ &\quad + (0.1u_{x_2x_2} + 0.2u_{x_2x_4} + 0.2u_{x_4x_4}) + (0.2u_{x_3x_3} + 0.15u_{x_3x_4} \\ &\quad + 0.1u_{x_4x_4}) + S(x_1, x_2, x_3, x_4, t), \end{aligned} \tag{57}$$

where $(x_1, x_2, x_3, x_4) \in \Omega = \{0 < x_1 < 2\pi, 0 < x_2 < 2\pi, 0 < x_3 < 2\pi, 0 < x_4 < 2\pi\}$ with periodic boundary condition. The initial condition is $u(x_1, x_2, x_3, x_4, 0) = \sin(x_1 + x_2 + x_3 + x_4)$. The exact solution is

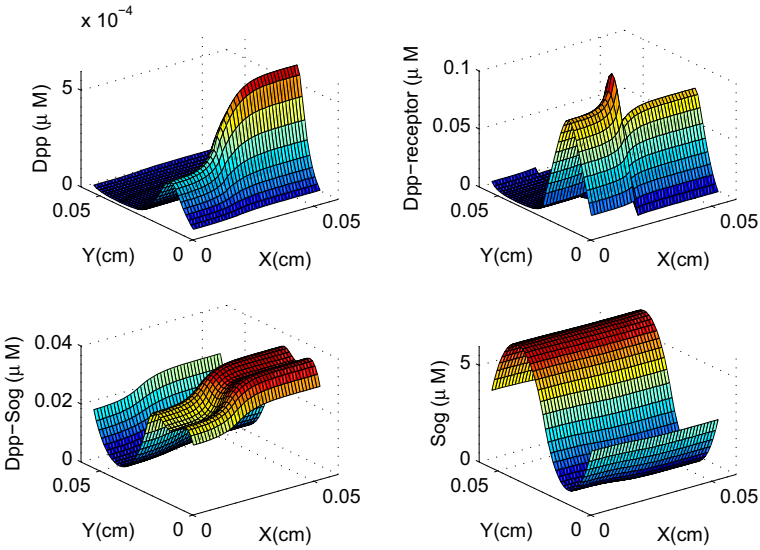

Fig. 2 Numerical solutions of Example 6 using the cIIF2 scheme: concentrations of $[L]$, $[LR]$, $[LS]$, $[S]$ at $T = 100$ s for the Dpp-Sog system when receptors are over-expressed. $\Delta t = h_x = h_y = 0.001375$ in the simulation. Parameters are $D_L = D_{LS} = D_S = 85\,\mu\text{m}^2\,\text{s}^{-1}$; $v_L = 1\,\text{nM}\,\text{s}^{-1}$; $v_S = 80\,\text{nM}\,\text{s}^{-1}$; $k_{on} = 0.4\,\mu\text{M}^{-1}\,\text{s}^{-1}$; $k_{off} = 4 \times 10^{-6}\,\text{s}^{-1}$; $k_{deg} = 5 \times 10^{-4}\,\text{s}^{-1}$; $j_{on} = 95\,\mu\text{M}^{-1}\,\text{s}^{-1}$; $j_{off} = 4 \times 10^{-6}\,\text{s}^{-1}$; $j_{deg} = 0.54\,\text{s}^{-1}$; $\tau = 1$; $R_h = 9\,\mu\text{M}$; $R_0 = 3\,\mu\text{M}$

$$u(x_1, x_2, x_3, x_4) = e^{-0.5t} \sin(x_1 + x_2 + x_3 + x_4).$$

The source term

$$S(x_1, x_2, x_3, x_4, t) = (4e^{-0.5t} \cos(x_1 + x_2 + x_3 + x_4) + 2)e^{-0.5t} \sin(x_1 + x_2 + x_3 + x_4).$$

We compute the problem until the final time $T = 1$. The KrylovIIF2-WENO scheme (17) and the AcIIF2-WENO scheme (42), (72), (73) with Krylov subspace approximations to matrix exponentials in (72) and (73) are used. Here time step sizes are determined only by the convection (hyperbolic) part of the equation with CFL number 0.1. Numerical results are reported in Tables 18 and 19. We can see that both schemes achieve the same numerical errors and second order accuracy. However, the KrylovIIF2-WENO scheme is much more efficient than the AcIIF2-WENO scheme with Krylov subspace approximations for this example.

Example 8 (*A 3D convection–diffusion equation with anisotropic diffusion and variable diffusion coefficients*) We add convection terms to the example 5 and consider the following three-dimensional convection–diffusion equation with cross-derivative diffusion terms and variable diffusion coefficients

$$\begin{aligned} u_t + (\frac{1}{2}u^2)_x + (\frac{1}{2}u^2)_y + (\frac{1}{2}u^2)_z &= 0.5u_{xx} - 0.5\sin(x+y)u_{xy} \\ &+ 0.5u_{yy} + 0.5u_{xx} - \frac{1}{3}\cos(y)u_{xz} + \frac{1}{3}u_{zz} + 0.5(1+\cos x)u_{yy} \\ &- 0.5(1+\cos x)u_{yz} + \frac{1}{3}(1+\cos x)u_{zz} + S(x, y, z, t), \end{aligned} \tag{58}$$

Table 18 Example 7: KrylovIIF2-WENO scheme, final time $T = 1.0$

$N \times N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10 \times 10$	2.27×10^{-2}		6.55		6.52		0.65	
$20 \times 20 \times 20 \times 20$	1.01×10^{-2}	1.18	242.15	36.97	241.64	37.08	10.51	16.13
$40 \times 40 \times 40 \times 40$	3.30×10^{-3}	1.61	8,013.72	33.09	8,005.98	33.13	166.82	15.87
$80 \times 80 \times 80 \times 80$	9.00×10^{-4}	1.87	316,945.98	39.55	316,803.84	39.57	3,084.58	18.49

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

Table 19 Example 7: AcIIF2-WENO scheme with Krylov subspace approximations, final time $T = 1.0$

$N \times N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10 \times 10$	2.27×10^{-2}		110.57		110.54		11.07	
$20 \times 20 \times 20 \times 20$	1.01×10^{-2}	1.18	2,290.56	20.72	2,290.05	20.72	99.25	8.96
$40 \times 40 \times 40 \times 40$	3.30×10^{-3}	1.61	60,778.28	26.53	60,770.28	26.54	1,269.46	12.79
$80 \times 80 \times 80 \times 80$	9.00×10^{-4}	1.87	1,812,641.33	29.82	1,812,266.39	29.82	17,984.97	14.17

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

where $(x, y, z) \in \Omega = \{0 < x < 2\pi, 0 < y < 2\pi, 0 < z < 2\pi\}$ with periodic boundary conditions. The initial condition is $u(x, y, z, 0) = \sin(x + y + z)$. The exact solution of this equation is

$$u(x, y, z, t) = e^{-0.2t} \sin(x + y + z).$$

And the source term $S(x, y, z, t)$ is

$$S(x, y, z, t) = \Big(3e^{-0.2t} \cos(x + y + z) + \frac{59}{30} - 0.5 \sin(x + y) + \frac{1}{3}(\cos(x) - \cos(y))\Big) e^{-0.2t} \sin(x + y + z).$$

We compute the problem until the final time $T = 1$. The KrylovIIF2-WENO scheme (17), and the AcIIF2-WENO scheme (42), (70), (71) with Krylov subspace approximations to matrix exponentials in (70) and (71) are tested. Time step sizes are determined only by the convection (hyperbolic) part of the equation with CFL number 0.1. Numerical results are reported in Tables 20 and 21. The same observations as the last example are obtained. Both schemes achieve almost the same numerical errors and second order accuracy. The KrylovIIF2-WENO scheme is much more efficient than the AcIIF2-WENO scheme with Krylov subspace approximations for this convection–diffusion example with anisotropic diffusion and variable diffusion coefficients.

Example 9 (*A convection-dominated problem*) In this example, we test the performance of the schemes for convection-dominated case. Consider the two-dimensional nonlinear viscous

Table 20 Example 8: KrylovIIF2-WENO scheme, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	1.37×10^{-1}		1.22		1.22		0.09	
$20 \times 20 \times 20$	2.99×10^{-2}	2.19	21.24	17.35	21.21	17.37	0.76	8.11
$40 \times 40 \times 40$	5.28×10^{-3}	2.50	393.16	18.51	392.95	18.52	6.89	9.10
$80 \times 80 \times 80$	1.09×10^{-3}	2.28	8,463.98	21.53	8,462.12	21.53	61.55	8.93
$160 \times 160 \times 160$	2.62×10^{-4}	2.05	95,558.44	11.29	95,545.98	11.29	413.83	6.72

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

Table 21 Example 8: AcIIF2-WENO scheme with Krylov subspace approximations, final time $T = 1.0$

$N \times N \times N$	L^∞ error	Order	CPU (s)	R1	CPU1 (s)	R2	CPU2 (s)	R3
$10 \times 10 \times 10$	1.37×10^{-1}		10.05		10.05		0.78	
$20 \times 20 \times 20$	2.99×10^{-2}	2.19	81.47	8.10	81.44	8.10	2.91	3.73
$40 \times 40 \times 40$	5.28×10^{-3}	2.50	936.26	11.49	936.05	11.49	16.43	5.64
$80 \times 80 \times 80$	1.09×10^{-3}	2.28	11,024.66	11.78	11,022.74	11.78	91.27	5.56
$160 \times 160 \times 160$	2.61×10^{-4}	2.06	215,299.80	19.53	215,287.37	19.53	936.02	10.26

CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds. R1, R2 and R3 are the ratios of corresponding CPU times on an $N \times N \times N$ mesh to that on a $\frac{N}{2} \times \frac{N}{2} \times \frac{N}{2}$ mesh

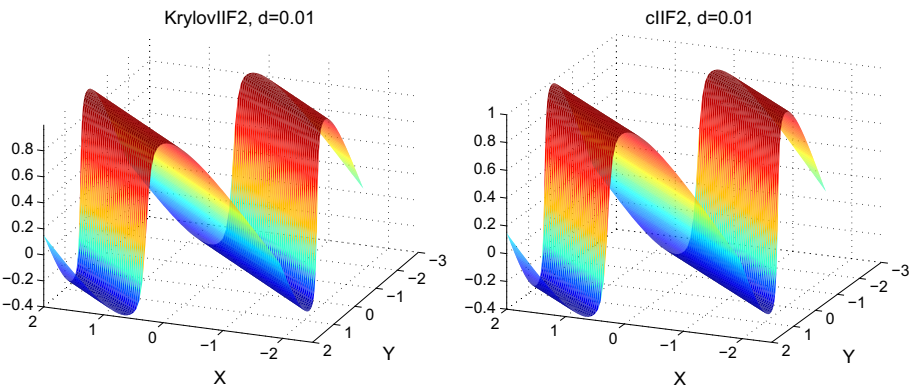

Fig. 3 Numerical solutions of nonlinear viscous Burgers' equation on a 80×80 mesh by the Krylov IIF2-WENO scheme and the cIIF2-WENO scheme. Time $T = 5/\pi^2$. *Left picture* result of Krylov IIF2-WENO; *right picture* result of cIIF2-WENO

Table 22 CPU time comparisons for solving the nonlinear viscous Burgers' equation

$N \times N$	KrylovIIF2 CPU (s)	cIIF2 CPU (s)
40×40	0.52	0.10
80×80	5.83	1.20
160×160	29.81	15.15
320×320	378.36	290.28
640×640	3,067.96	4,344.81

The PDE is solved until the final time $T = 5/\pi^2$. KrylovIIF2 CPU: CPU time of the Krylov IIF2-WENO scheme; cIIF2 CPU: CPU time of the cIIF2-WENO scheme

Table 23 Numerical errors and accuracy orders for the KrylovIIF2 scheme and the AcIIF2 scheme with Krylov subspace approximations for the 3D Fokker–Planck equation (63)

KrylovIIF2-WENO		
Time step size	L^∞error	Order
$\triangle t$	1.56×10^{-8}	
$\triangle t/2$	3.90×10^{-9}	2.00
$\triangle t/4$	1.00×10^{-9}	1.96
AcIIF2-WENO with Krylov subspace approx.		
Time step	L^∞ error	Order
$\triangle t$	1.56×10^{-8}	
$\triangle t/2$	3.90×10^{-9}	2.00
$\triangle t/4$	1.00×10^{-9}	1.96

The third order WENO scheme is used for the convection terms. Final time $T = 5$. $\triangle t = 0.017$

Burgers' equation

$$\begin{cases} u_t + (\frac{u^2}{2})_x + (\frac{u^2}{2})_y = 0.01\Delta u, \quad -2 \le x \le 2, \ -2 \le y \le 2, \\ u(x, y, 0) = 0.3 + 0.7 \sin(\frac{\pi}{2}(x + y)), \end{cases} \tag{59}$$

with periodic boundary condition. Since the viscous coefficient is much smaller than the convection coefficient, a sharp gradient (the shock wave) is developed along with the time evolution. The Krylov IIF2-WENO scheme and the cIIF2-WENO scheme are used to solve the PDE to $T = 5/\pi^2$. The numerical results are reported in Fig. 3. We can observe that the WENO scheme plays an important role here to obtain a sharp, non-oscillatory shock transition region. The time step size is only restricted by the hyperbolic part of the PDE with CFL number 0.5. We compare the CPU times of the Krylov IIF2-WENO scheme and the cIIF2-WENO scheme on different meshes. The results are reported in Table 22. Consistent observations with previous examples are obtained. For this example which has diffusion terms without cross-derivatives, compact approach is more efficient than the Krylov approach, except the case with a very refined mesh.

Example 10 (*Fokker–Planck equations*)The Fokker–Planck equation (FPE) [9,39] describes in a statistical sense how a collection of initial data evolves in time, e.g., in describing Brownian motion. It is a N-dimensional convection–diffusion equation and has been applied in computing statistical properties in many systems. In [49], AcIIF schemes with second order

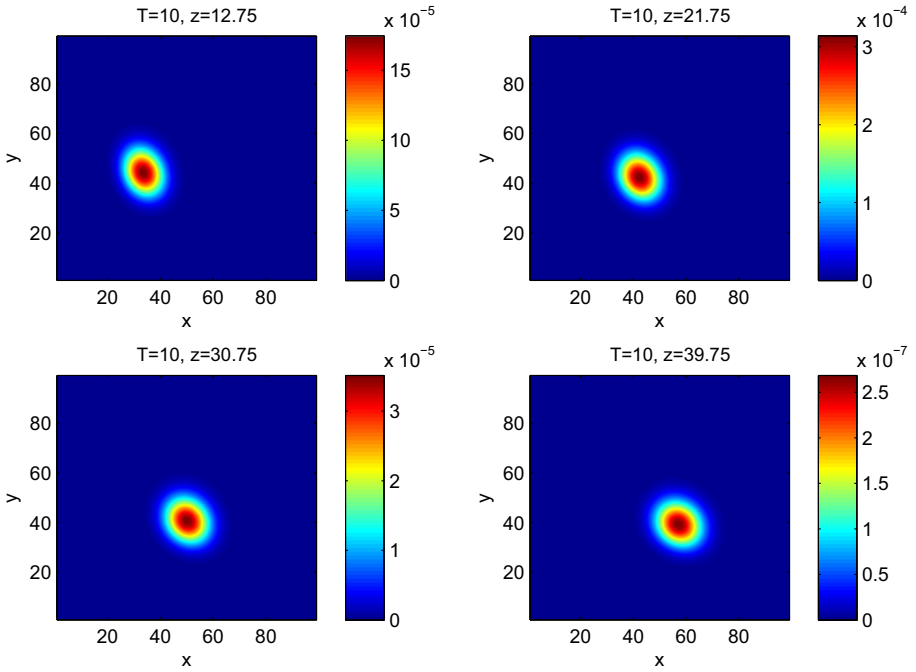

Fig. 4 Distribution of molecular species A and B with $E_A = 12.75, 21.75, 30.75, 39.75$. Numerical solutions of (63) using the KrylovIIF2-WENO scheme. Final time $T = 10$. $\triangle t = 0.017$. The numbers of spatial grid points are $N_A = 120$, $N_B = 120$, $N_{E_A} = 60$

central difference spatial discretizations for the diffusion terms were applied in solving FPEs which describe the time evolution of the probability density function of stochastic systems [40]. The general form of FPEs is

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t} = -\sum_{r=1}^{R}\left\{\sum_{i=1}^{N} n_{ri}\frac{\partial}{\partial x_i}\left(q_r(\boldsymbol{x},t) - \frac{1}{2}\sum_{j=1}^{N} n_{rj}\frac{\partial q_r(\boldsymbol{x},t)}{\partial x_j}\right)\right\}, \tag{60}$$

where $p(\boldsymbol{x},t)$ is the probability density of the system at the state $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ and time t. In the context of bio-chemical reactions, R denotes the total number of chemical reactions in the system, N the total number of species involving in the reaction, and x_i denotes the copy number of ith reactant. n_{ri} is the change of x_i when the rth reaction occurs once. $q_r(\boldsymbol{x},t)$ is defined by $q_r(\boldsymbol{x},t) = w_r(\boldsymbol{x})p(\boldsymbol{x},t)$, where $w_r(\boldsymbol{x},t)$ is the reaction propensity function for rth reaction at state $\boldsymbol{x}$ and time t. In this section, we study computational efficiency of Krylov IIF-WENO scheme and AcIIF-WENO scheme for solving high dimensional FPE. Since IIF schemes in this paper are multistep methods, numerical values at the first time step are needed to start computations for solving convection–diffusion equations. We use a third order Runge–Kutta scheme for the first step time evolution. Then the second order Krylov IIF scheme and AcIIF scheme are used to continue the time evolution.

(1) A three dimensional Fokker–Planck equation

We first compare the computational efficiency of the KrylovIIF2-WENO scheme (17) and the AcIIF2-WENO scheme (42), (70), (71) with Krylov subspace approximations for a three

Table 24 CPU time for KrylovIIF2 scheme and the AcIIF2 scheme with Krylov subspace approximations for the 3D Fokker–Planck equation (63)

	CPU	CPU1	CPU2
KrylovIIF2-WENO	44,568.7	44,562.3	75.24
AcIIF2-WENO with Krylov	183,126.0	183,120.0	309.38

The third order WENO scheme is used for the convection terms. Final time $T = 10$. $\triangle t = 0.017$. CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds

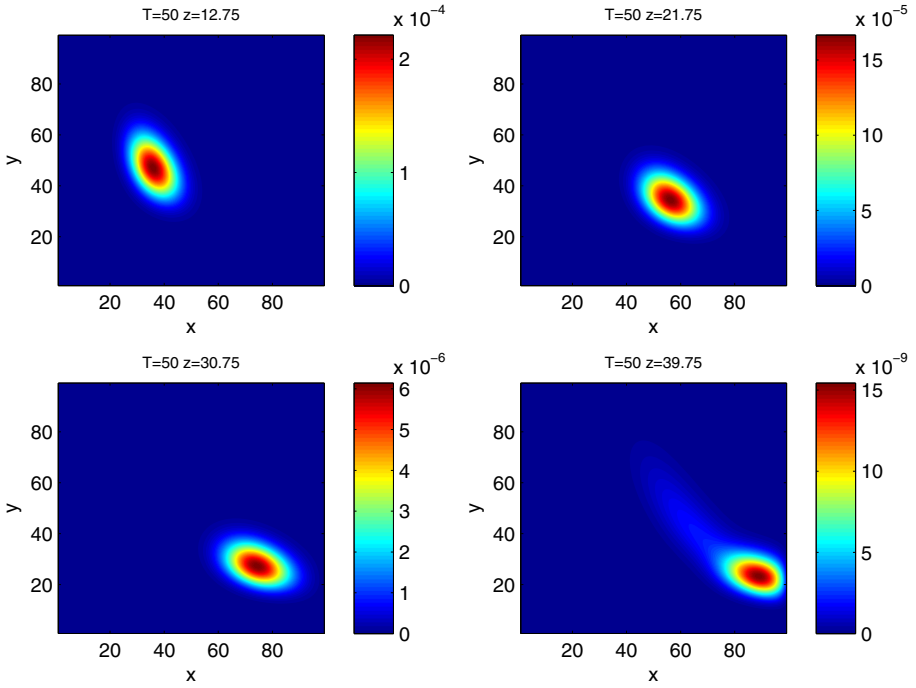

Fig. 5 Distribution of molecular species A and B with $E_A = 12.75, 21.75, 30.75, 39.75$. Numerical solutions of (63) using the KrylovIIF2-WENO scheme. Final time $T = 50$. $\triangle t = 0.017$. The numbers of spatial grid points are $N_A = 120$, $N_B = 120$, $N_{E_A} = 60$

dimensional Fokker–Planck equation [44] which involves two metabolites A and B and one enzyme E_A. The reactions are described as following (here $\emptyset$ means that there is no reactant or product in the reaction):

$$\begin{aligned} &\emptyset \xrightarrow{\frac{k_A[E_A]}{1+[A]/K_I}} A, \quad \emptyset \xrightarrow{k_B} B, \\ &A + B \xrightarrow{k[A][B]} \emptyset, \\ &A \xrightarrow{\mu[A]} \emptyset, \quad B \xrightarrow{\mu[B]} \emptyset, \\ &\emptyset \xrightarrow{\frac{k_{E_A}}{1+[A]/K_R}} E_A, \quad E_A \xrightarrow{\mu[E_A]} \emptyset. \end{aligned} \tag{61}$$

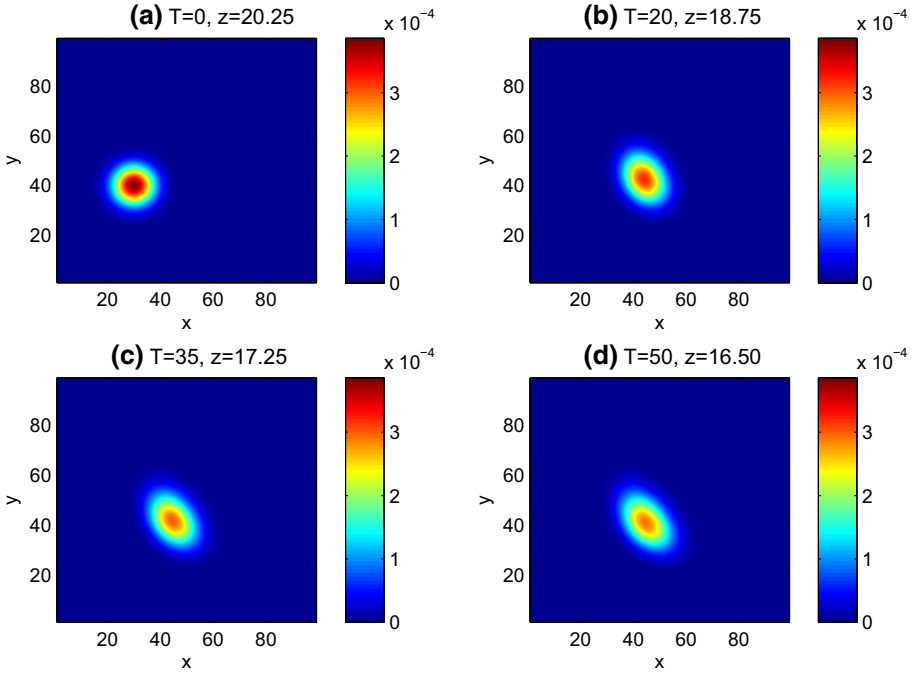

Fig. 6 Distribution of molecular species A and B with different E_A values, at time $T = 0, 20, 35, 50$. Numerical solutions of (63) using the KrylovIIF2-WENO scheme. $\triangle t = 0.017$. The numbers of spatial grid points are $N_A = 120$, $N_B = 120$, $N_{E_A} = 60$

In this system, the total number of reactions R is 7, and the total number of chemical species N is 3. The vectors $\boldsymbol{n_r} = (n_{r1}, n_{r2}, n_{r3})$ are $\boldsymbol{n_1} = (1, 0, 0), \boldsymbol{n_2} = (0, 1, 0), \boldsymbol{n_3} = (-1, -1, 0)$, $\boldsymbol{n_4} = (-1, 0, 0)$, $\boldsymbol{n_5} = (0, -1, 0)$, $\boldsymbol{n_6} = (0, 0, 1)$, $\boldsymbol{n_7} = (0, 0, -1)$. We denote the system state $\boldsymbol{x}$ by $\boldsymbol{x} = (x_1, x_2, x_3)$ which is $([A], [B], [E_A])$ in this case. Then the propensity functions $w_r(\boldsymbol{x})$ are

$$\begin{aligned} &w_1 = \frac{k_A x_3}{1 + x_1/K_I}, \quad w_2 = k_B, \quad w_3 = k x_1 x_2, \\ &w_4 = \mu x_1, \quad w_5 = \mu x_2, \quad w_6 = \frac{k_{E_A}}{1 + x_1/K_R}, \quad w_7 = \mu x_3, \end{aligned} \tag{62}$$

where $k_A = 0.3\,\mathrm{s}^{-1}$, $k_B = 2\,\mathrm{s}^{-1}$, $K_I = 30$, $k = 0.001\,\mathrm{s}^{-1}$, $\mu = 0.004\,\mathrm{s}^{-1}$, $K_R = 30$ and $k_{E_A} = 1\,\mathrm{s}^{-1}$ [44]. Then the FPE can be written as

$$\frac{\partial p(\boldsymbol{x}, t)}{\partial t} = -(L_1 + L_2 + L_3 + L_4 + L_5 + L_6 + L_7), \tag{63}$$

where L_r represents the operator for the rth reaction. Specifically,

$$L_1 = \frac{\partial q_1(\boldsymbol{x}, t)}{\partial x_1} - \frac{1}{2}\frac{\partial^2 q_1(\boldsymbol{x}, t)}{\partial x_1^2},$$

$$L_2 = \frac{\partial q_2(\boldsymbol{x}, t)}{\partial x_2} - \frac{1}{2}\frac{\partial^2 q_2(\boldsymbol{x}, t)}{\partial x_2^2},$$

$$
\begin{aligned}
L_3 &= -\frac{\partial q_3(\boldsymbol{x},t)}{\partial x_1} - \frac{\partial q_3(\boldsymbol{x},t)}{\partial x_2} - \frac{1}{2}\Big(\frac{\partial^2 q_3(\boldsymbol{x},t)}{\partial x_1^2} + \frac{\partial^2 q_3(\boldsymbol{x},t)}{\partial {x_2}^2} + 2\frac{\partial^2 q_3(\boldsymbol{x},t)}{\partial x_1 \partial x_2}\Big),\\
L_4 &= -\frac{\partial q_4(\boldsymbol{x},t)}{\partial x_1} - \frac{1}{2}\frac{\partial^2 q_4(\boldsymbol{x},t)}{\partial x_1^2},\\
L_5 &= -\frac{\partial q_5(\boldsymbol{x},t)}{\partial x_2} - \frac{1}{2}\frac{\partial^2 q_5(\boldsymbol{x},t)}{\partial x_2^2},\\
L_6 &= \frac{\partial q_6(\boldsymbol{x},t)}{\partial x_3} - \frac{1}{2}\frac{\partial^2 q_6(\boldsymbol{x},t)}{\partial x_3^2},\\
L_7 &= -\frac{\partial q_7(\boldsymbol{x},t)}{\partial x_3} - \frac{1}{2}\frac{\partial^2 q_7(\boldsymbol{x},t)}{\partial x_3^2}.
\end{aligned} \tag{64}
$$

The computational domain is $\Omega = [0, 100] \times [0, 100] \times [0, 45]$, which covers nearly all the possible states of the chemical reactions, since the probability of $[A] > 100$, $[B] > 100$, and $[E_A] > 45$ is sufficiently small. The initial condition in our simulation is a Gaussian distribution centered at point (30, 40, 20) with standard deviation $\sqrt{30}$. Zero Dirichlet boundary conditions are used.

For spatial discretizations, we use the third order WENO scheme for the convection terms and the second order central difference scheme for the diffusion terms. And we compare the second order Krylov IIF scheme and the second order AcIIF scheme with Krylov subspace approximations. For simulation results shown in the figures here, the time step size $\triangle t$ is 0.017 (corresponding to the CFL number 0.4 for the convection part) and the numbers of spatial grid points are $N_A = 120$, $N_B = 120$, $N_{E_A} = 60$. In Table 23, we list the errors and accuracy orders for both schemes, and the same numerical errors and second order accuracy are obtained. Since there is no explicit form for the exact solution in this example, we focus on testing the schemes' temporal accuracy. So the spatial resolution is fixed to be $120 \times 120 \times 60$, and numerical errors for a time step size $\triangle t$ are obtained by calculating the difference of numerical values for $\triangle t$ and $\triangle t/2$. We compare the computational efficiency of these two schemes and list CPU times of using them to solve the problem until the final time $T = 10$ with $\triangle t = 0.017$, in Table 24. The CPU times in Table 24 show that the KrylovIIF2-WENO scheme is more efficient than the AcIIF2-WENO scheme with Krylov subspace approximations, for this example. In Figs. 4, 5 and 6, we show contour plots of numerical solutions by the KrylovIIF2-WENO scheme on two dimensional domain of molecular species A and B, with different values of the third dimension E_A. Contour plots of numerical solutions by the AcIIF2-WENO scheme with Krylov subspace approximations are presented in Figs. 7, 8 and 9. We see that both methods generate similar numerical solutions.

(2) A four dimensional Fokker–Planck equation

We further test the methods for a higher dimensional problem, i.e., a four dimensional FPE which involves two metabolites A and B and two enzymes E_A and E_B. The reactions are described as following (here $\emptyset$ means that there is no reactant or product in the reaction):

$$
\begin{aligned}
&\emptyset \xrightarrow{\frac{k_A[E_A]}{1+[A]/K_I}} A, \quad \emptyset \xrightarrow{\frac{k_B[E_B]}{1+[B]/K_I}} B,\\
&A + B \xrightarrow{k[A][B]} \emptyset,\\
&A \xrightarrow{\mu[A]} \emptyset, \quad B \xrightarrow{\mu[B]} \emptyset,
\end{aligned}
$$

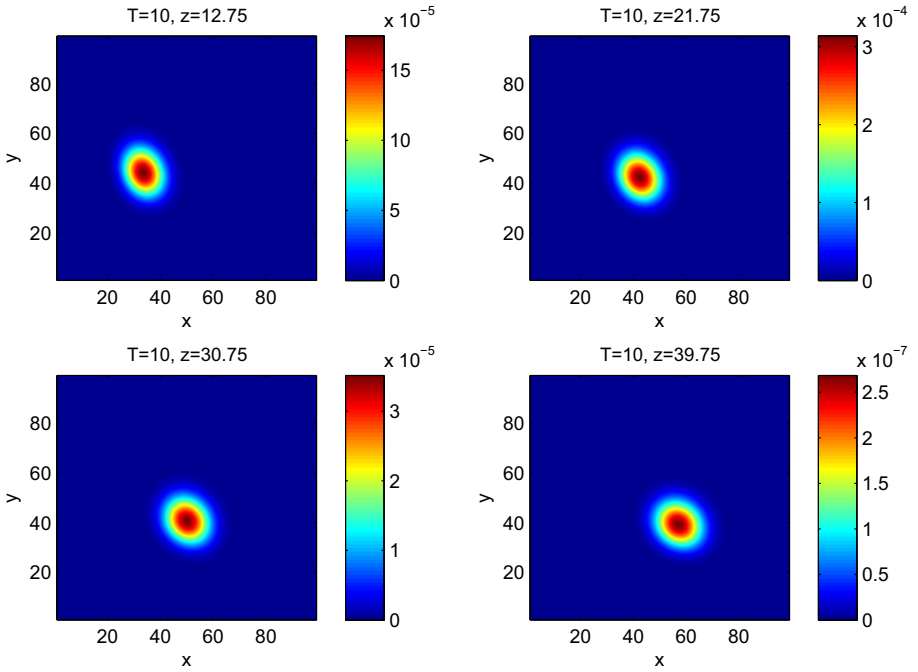

Fig. 7 Distribution of molecular species A and B with $E_A = 12.75, 21.75, 30.75, 39.75$. Numerical solutions of (63) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time $T = 10$. $\triangle t = 0.017$. The numbers of spatial grid points are $N_A = 120$, $N_B = 120$, $N_{E_A} = 60$

$$\begin{aligned} \emptyset &\xrightarrow{\frac{k_{E_A}}{1+[A]/K_R}} E_A, \quad E_A \xrightarrow{\mu[E_A]} \emptyset, \\ \emptyset &\xrightarrow{\frac{k_{E_B}}{1+[B]/K_R}} E_B, \quad E_B \xrightarrow{\mu[E_B]} \emptyset. \end{aligned} \tag{65}$$

In this system, the total number of reactions R is 9, and the total number of chemical species N is 4. The vectors $\boldsymbol{n_r} = (n_{r1}, n_{r2}, n_{r3}, n_{r4})$ are $\boldsymbol{n_1} = (1, 0, 0, 0)$, $\boldsymbol{n_2} = (0, 1, 0, 0)$, $\boldsymbol{n_3} = (-1, -1, 0, 0)$, $\boldsymbol{n_4} = (-1, 0, 0, 0)$, $\boldsymbol{n_5} = (0, -1, 0, 0)$, $\boldsymbol{n_6} = (0, 0, 1, 0)$, $\boldsymbol{n_7} = (0, 0, -1, 0)$, $\boldsymbol{n_8} = (0, 0, 0, 1)$, $\boldsymbol{n_9} = (0, 0, 0, -1)$. We denote the system state $\boldsymbol{x}$ by $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ which is $([A], [B], [E_A], [E_B])$ in this case. Then the propensity functions $w_r(\boldsymbol{x})$ are

$$\begin{aligned} &w_1 = \frac{k_A x_3}{1 + x_1/K_I}, \quad w_2 = \frac{k_B x_4}{1 + x_2/K_I}, \quad w_3 = k x_1 x_2, \quad w_4 = \mu x_1, \\ &w_5 = \mu x_2, \quad w_6 = \frac{k_{E_A}}{1 + x_1/K_R}, \quad w_7 = \mu x_3, \quad w_8 = \frac{k_{E_B}}{1 + x_2/K_R}, \quad w_9 = \mu x_4, \end{aligned} \tag{66}$$

where $k_A = 0.3\,\mathrm{s}^{-1}$, $k_B = 0.3\,\mathrm{s}^{-1}$, $K_I = 60$, $k = 0.001\,\mathrm{s}^{-1}$, $\mu = 0.002\,\mathrm{s}^{-1}$, $K_R = 30$, $k_{E_A} = 0.02\,\mathrm{s}^{-1}$ and $k_{E_B} = 0.02\,\mathrm{s}^{-1}$ [44]. Then the FPE can be written as

$$\frac{\partial p(\boldsymbol{x}, t)}{\partial t} = -(L_1 + L_2 + L_3 + L_4 + L_5 + L_6 + L_7 + L_8 + L_9), \tag{67}$$

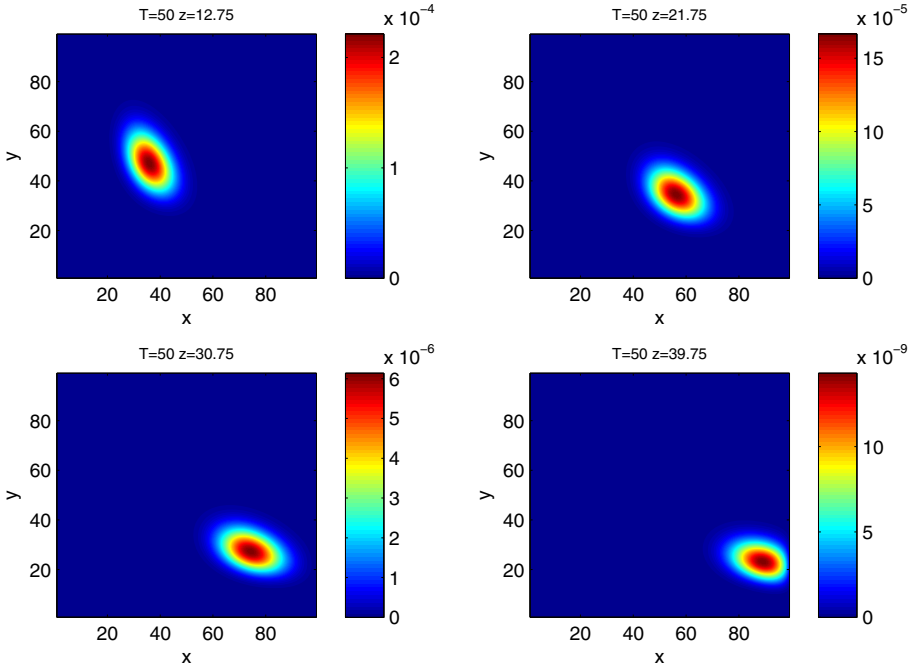

Fig. 8 Distribution of molecular species A and B with $E_A = 12.75, 21.75, 30.75, 39.75$. Numerical solutions of (63) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time $T = 50$. $\triangle t = 0.017$. The numbers of spatial grid points are $N_A = 120$, $N_B = 120$, $N_{E_A} = 60$

where L_r represents the operator for the rth reaction. Specifically,

$$L_1 = \frac{\partial q_1(\boldsymbol{x},t)}{\partial x_1} - \frac{1}{2}\frac{\partial^2 q_1(\boldsymbol{x},t)}{\partial x_1^2},$$

$$L_2 = \frac{\partial q_2(\boldsymbol{x},t)}{\partial x_2} - \frac{1}{2}\frac{\partial^2 q_2(\boldsymbol{x},t)}{\partial x_2^2},$$

$$L_3 = -\frac{\partial q_3(\boldsymbol{x},t)}{\partial x_1} - \frac{\partial q_3(\boldsymbol{x},t)}{\partial x_2} - \frac{1}{2}\Big(\frac{\partial^2 q_3(\boldsymbol{x},t)}{\partial x_1^2} + \frac{\partial^2 q_3(\boldsymbol{x},t)}{\partial {x_2}^2} + 2\frac{\partial^2 q_3(\boldsymbol{x},t)}{\partial x_1 \partial x_2}\Big),$$

$$L_4 = -\frac{\partial q_4(\boldsymbol{x},t)}{\partial x_1} - \frac{1}{2}\frac{\partial^2 q_4(\boldsymbol{x},t)}{\partial x_1^2},$$

$$L_5 = -\frac{\partial q_5(\boldsymbol{x},t)}{\partial x_2} - \frac{1}{2}\frac{\partial^2 q_5(\boldsymbol{x},t)}{\partial x_2^2},$$

$$L_6 = \frac{\partial q_6(\boldsymbol{x},t)}{\partial x_3} - \frac{1}{2}\frac{\partial^2 q_6(\boldsymbol{x},t)}{\partial x_3^2},$$

$$L_7 = -\frac{\partial q_7(\boldsymbol{x},t)}{\partial x_3} - \frac{1}{2}\frac{\partial^2 q_7(\boldsymbol{x},t)}{\partial x_3^2},$$

$$L_8 = \frac{\partial q_8(\boldsymbol{x},t)}{\partial x_4} - \frac{1}{2}\frac{\partial^2 q_8(\boldsymbol{x},t)}{\partial x_4^2},$$

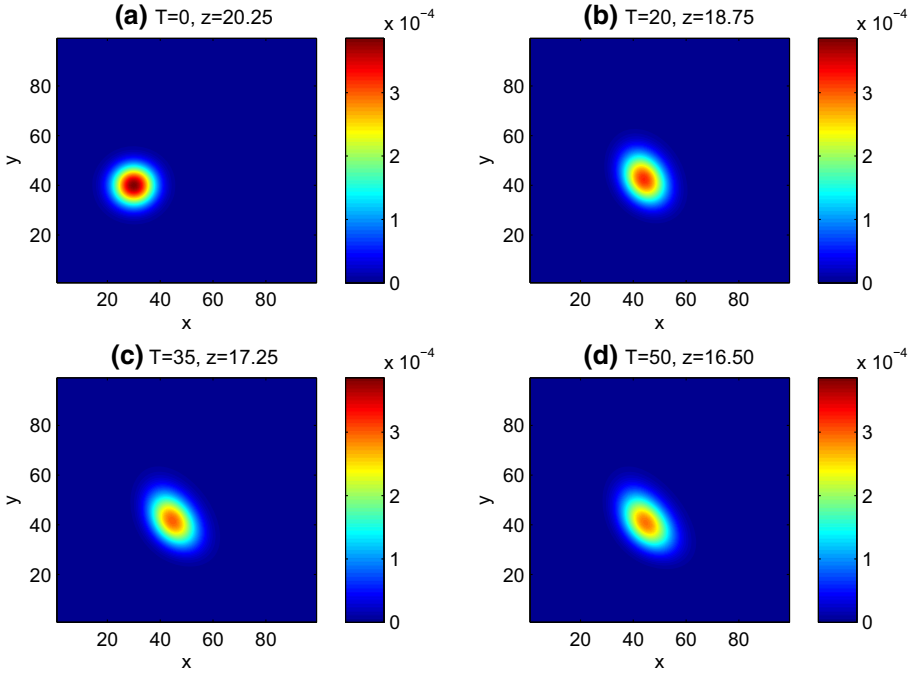

Fig. 9 Distribution of molecular species A and B with different E_A values, at time $T = 0, 20, 35, 50$. Numerical solutions of (63) using the AcIIF2-WENO scheme with Krylov subspace approximations. $\triangle t = 0.017$. The numbers of spatial grid points are $N_A = 120$, $N_B = 120$, $N_{E_A} = 60$

Table 25 Numerical errors and accuracy orders for the KrylovIIF2 scheme and the AcIIF2 scheme with Krylov subspace approximations for the 4D Fokker–Planck equation (67)

KrylovIIF2-WENO		
Time step size	L^∞ error	Order
$\triangle t$	1.03×10^{-8}	
$\triangle t/2$	2.58×10^{-9}	2.00
$\triangle t/4$	6.47×10^{-10}	2.00
AcIIF2-WENO with Krylov subspace approx.		
Time step	L^∞ error	Order
$\triangle t$	1.03×10^{-8}	
$\triangle t/2$	2.58×10^{-9}	2.00
$\triangle t/4$	6.47×10^{-10}	2.00

The third order WENO scheme is used for the convection terms. Final time $T = 5$. $\triangle t = 0.1$

$$L_9 = -\frac{\partial q_9(\boldsymbol{x}, t)}{\partial x_4} - \frac{1}{2}\frac{\partial^2 q_9(\boldsymbol{x}, t)}{\partial x_4^2}. \tag{68}$$

The computational domain is $\Omega = [0, 80] \times [0, 80] \times [0, 30] \times [0, 30]$. The initial condition in our simulation is a Gaussian distribution centered at point $(30, 40, 15, 12)$ with standard deviation $\sqrt{40}$. Zero Dirichlet boundary conditions are used.

Same as that for the three dimensional problem, for spatial discretizations we use the third order WENO scheme for the convection terms and the second order central difference scheme

Table 26 CPU time for KrylovIIF2 scheme and the AcIIF2 scheme with Krylov subspace approximations for the 4D Fokker–Planck equation (67)

	CPU	CPU1	CPU2
KrylovIIF2-WENO	3831.98	3826.48	38.09
AcIIF2-WENO with Krylov	93320.7	93315.6	924.16

The third order WENO scheme is used for the convection terms. Final time $T = 10$. $\triangle t = 0.1$. CPU: CPU time for a complete simulation. CPU1: CPU time for time evolution part. CPU2: CPU time for one time step. CPU time unit: seconds

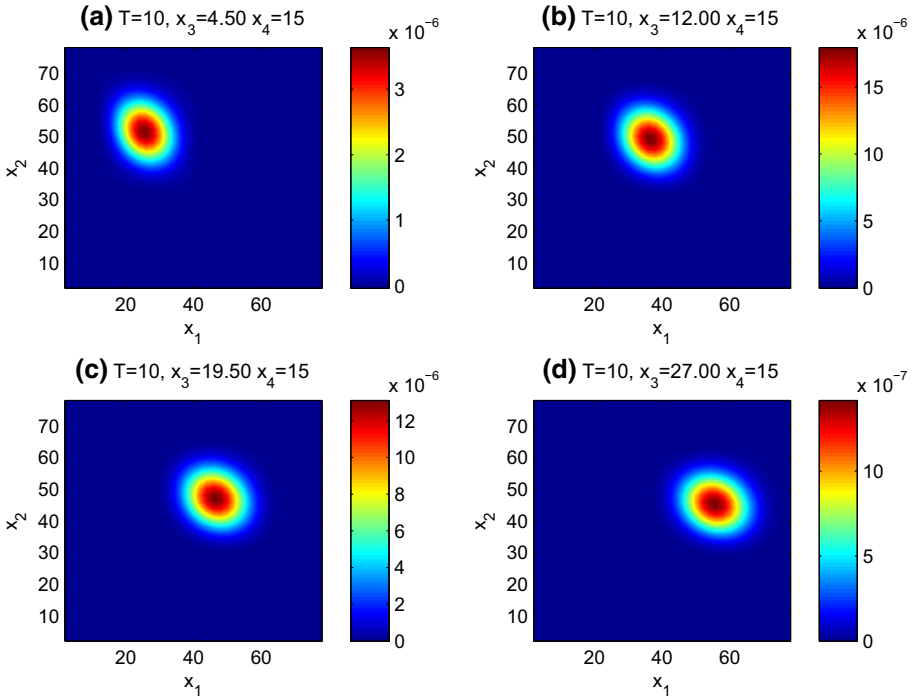

Fig. 10 Distribution of molecular species A and B with $E_A = 4.5, 12, 19.5, 27$ and $E_B = 15$. Numerical solutions of (67) using the KrylovIIF2-WENO scheme. Final time $T = 10$. $\triangle t = 0.1$. The numbers of spatial grid points are $N_A = 40$, $N_B = 40$, $N_{E_A} = 20$, $N_{E_B} = 20$

for the diffusion terms. We compare the computational efficiency of the second order Krylov IIF scheme (17) and the second order AcIIF scheme (42), (74)–(77) with Krylov subspace approximations. For simulation results shown in the figures here, the time step size $\triangle t$ is 0.1 (corresponding to the CFL number 0.6 for the convection part) and the numbers of spatial grid points are $N_A = 40$, $N_B = 40$, $N_{E_A} = 20$, $N_{E_B} = 20$. In Table 25, we list the errors and accuracy orders for both schemes, and the same numerical errors and second order accuracy are obtained. We compare the computational efficiency of these two schemes and list CPU times of using them to solve the problem until the final time $T = 10$ with $\triangle t = 0.1$, in Table 26. We obtain the same conclusion as that for the three dimensional problem. The CPU times in Table 26 show that the KrylovIIF2-WENO scheme is more efficient than the AcIIF2-WENO scheme with Krylov subspace approximations, for this four dimensional example. In

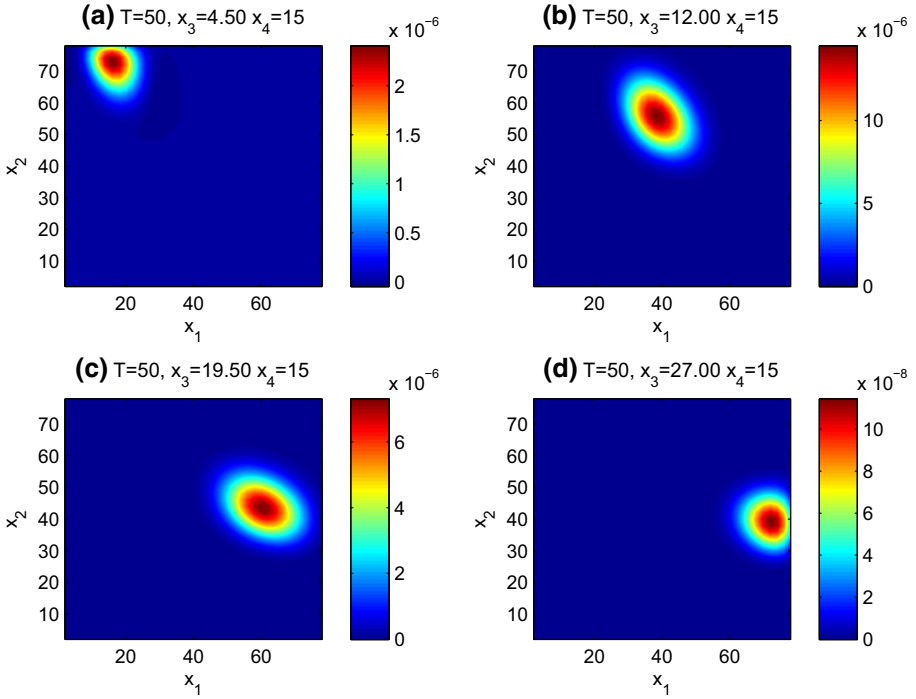

Fig. 11 Distribution of molecular species A and B with $E_A = 4.5, 12, 19.5, 27$ and $E_B = 15$. Numerical solutions of (67) using the KrylovIIF2-WENO scheme. Final time $T = 50$. $\triangle t = 0.1$. The numbers of spatial grid points are $N_A = 40$, $N_B = 40$, $N_{E_A} = 20$, $N_{E_B} = 20$

Figs. 10, 11 and 12, we show contour plots of numerical solutions by the KrylovIIF2-WENO scheme on two dimensional domain of molecular species A and B, with different values of the third and the fourth dimension E_A and E_B. Contour plots of numerical solutions by the AcIIF2-WENO scheme with Krylov subspace approximations are presented in Figs. 13, 14 and 15. We see that both methods generate similar numerical solutions.

4 Conclusions and Discussions

In this paper, we systematically perform numerical comparison and computational complexity analysis to study two different approaches in dealing with solving high spatial dimension diffusion and convection–diffusion PDE problems by integration factor WENO methods. Specifically, one approach is the cIIF/AcIIF method, and the other one is the Krylov IIF method, i.e., direct application of Krylov subspace approximations in efficiently calculating large matrix exponentials in integration factor methods. Via extensive numerical experiments and analysis of the results for various high spatial dimension problems, we find that both the cIIF/AcIIF method and the Krylov IIF method have their own advantages for different type of problems. The Krylov IIF method has linear computational complexity. For the numerical examples tested in this paper, it is shown that on not very refined meshes, the cIIF/AcIIF method is more efficient than the Krylov IIF method for problems whose diffusion terms do *not* have cross-derivatives. The Krylov IIF method is more efficient on such problems for very

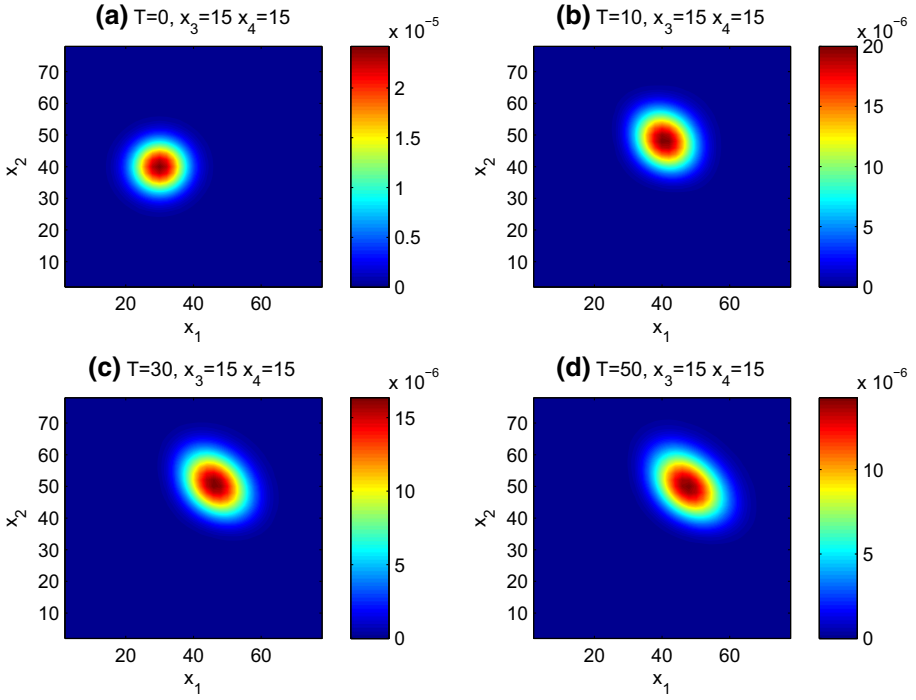

Fig. 12 Distribution of molecular species A and B with $E_A = 15$ and $E_B = 15$, at time $T = 0, 10, 30, 50$. Numerical solutions of (67) using the KrylovIIF2-WENO scheme. $\triangle t = 0.1$. The numbers of spatial grid points are $N_A = 40$, $N_B = 40$, $N_{E_A} = 20$, $N_{E_B} = 20$

refined meshes due to its linear computational complexity property. For high dimensional problems whose diffusion terms have no cross-derivatives, the cIIF/AcIIF method only needs to compute matrix exponentials with sizes as that for one spatial dimension problems (i.e., $N \times N$ matrices and N is the number of spatial grid points in one spatial direction). Hence it is very efficient. For high dimensional problems whose diffusion terms have cross-derivatives, the AcIIF method can reduce $N^d \times N^d$ matrices' exponentials to a series of $N^2 \times N^2$ matrices' exponentials. However, computations of these $N^2 \times N^2$ matrices' exponentials are still costly in CPU time and computer memory, especially for a not very coarse mesh. Applications of Krylov subspace approximations to these $N^2 \times N^2$ matrices' exponentials in the AcIIF method can significantly improve its computational efficiency. We compare three approaches including the AcIIF method, the AcIIF method with Krylov subspace approximation, and the direct Krylov IIF method for problems whose diffusion terms have cross-derivatives, and find that the most efficient method for such problems is the direct Krylov IIF method, as that shown in the numerical experiments. Certainly the efficiency of the Krylov IIF method depends on the dimension size M of Krylov subspace used in computation. In the development of Krylov IIF schemes for solving high spatial dimension convection–diffusion–reaction PDEs [5,20,21], M is taken to be 25 and Krylov subspace approximation errors are much smaller than truncation errors of the numerical schemes which discretize the PDEs, for different problems and matrices' sizes. Following the literature, for all examples in this paper, we choose $M = 25$ and obtain correct accuracy orders of the numerical schemes, even for very large $N^4 \times N^4$ matrices from the four spatial dimension PDEs. It will be interesting to study

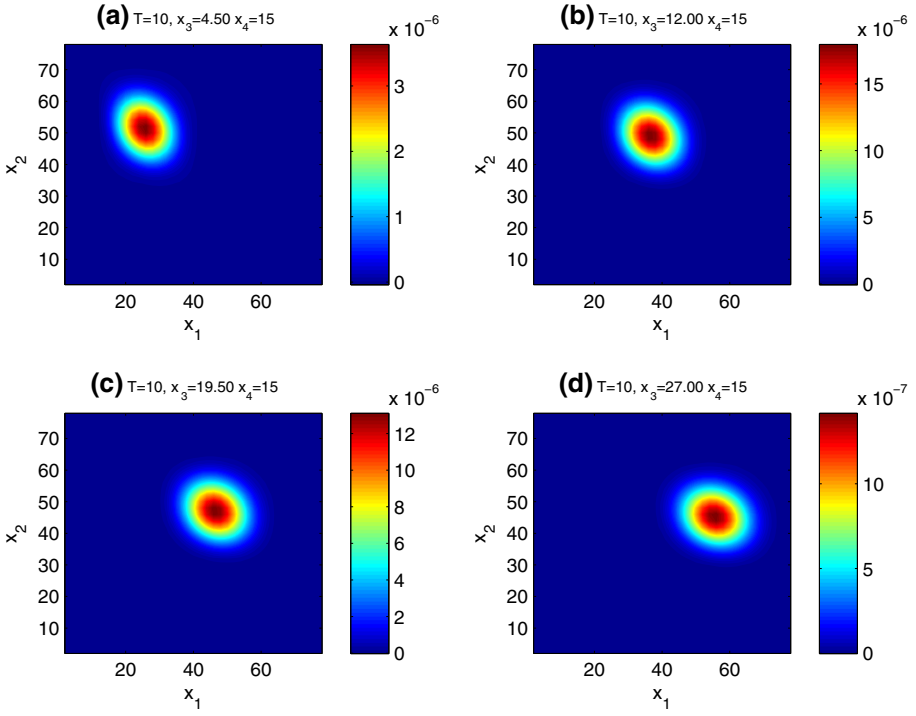

Fig. 13 Distribution of molecular species A and B with $E_A = 4.5, 12, 19.5, 27$ and $E_B = 15$. Numerical solutions of (67) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time $T = 10$. $\triangle t = 0.1$. The numbers of spatial grid points are $N_A = 40$, $N_B = 40$, $N_{E_A} = 20$, $N_{E_B} = 20$

possible dependence of the parameter M on different types of PDEs (different differential operators) and problems, which is one of our future work.

To further improve efficiency of IIF methods for high dimensional problems, both the AcIIF method and the Krylov IIF method have been implemented on sparse grids [31,50]. In [50], the AcIIF method was applied on sparse grids to solve high dimensional reaction–diffusion systems of dimensionality up to 6, and significantly higher efficiency was achieved. For higher spatial dimension problems with cross-derivative diffusion terms, the sparse grid Krylov IIF method in [31] could be very efficient as that shown in this paper. While both AcIIF and Krylov IIF methods on sparse grids [31,50] can be applied to higher spatial dimension problems, there are many interesting questions to address. For example, how could a computation on sparse grids achieve similar accuracy level as that on single grids? For higher spatial dimension problems, demands on computer memory increase significantly, and parallel computing may be needed. These interesting questions will be studied in our future research.

Another recent interesting work on the IIF method is to apply it in solving stochastic reaction–diffusion equations in [46]. Stochastic reaction–diffusion equations have broad applications in modeling biological or physical systems which are subjected to noises and environmental perturbations. Stiffness in stochastic reaction–diffusion equations may occur in the deterministic and/or the stochastic terms. In [46], the stiff deterministic diffusion and reaction terms were treated by the IIF approach, and the stochastic term was dealt with explicitly. Nice stability properties and efficiency of the original IIF method were preserved well. It

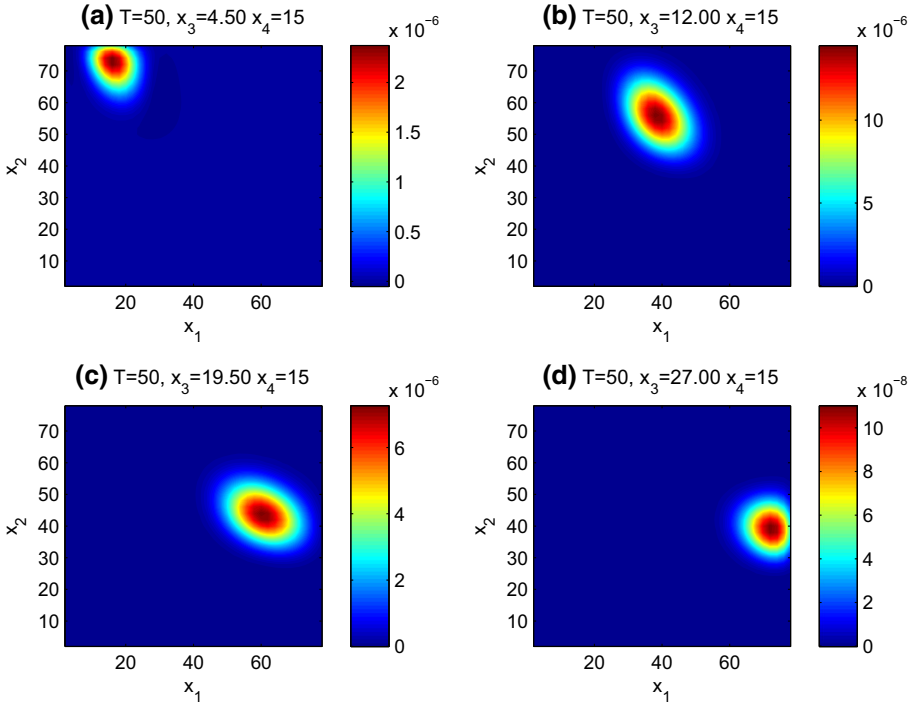

Fig. 14 Distribution of molecular species A and B with $E_A = 4.5, 12, 19.5, 27$ and $E_B = 15$. Numerical solutions of (67) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time $T = 50$. $\triangle t = 0.1$. The numbers of spatial grid points are $N_A = 40$, $N_B = 40$, $N_{E_A} = 20$, $N_{E_B} = 20$

provides an efficient new approach for solving stochastic reaction–diffusion equations with stiff deterministic terms. For such problems with high spatial dimensions, both Krylov IIF and AcIIF methods discussed in this paper can be straightforwardly applied in dealing with the large matrix exponential challenge arising from the stiff deterministic diffusion. We expect to see the effectiveness of the Krylov IIF and AcIIF methods in solving high spatial dimension stochastic problems, as that discussed in this paper. This is one of our future research.

Appendix: Detailed Formulae for AcIIF-WENO Schemes

(1) For the three dimensional CDR equation, if $\mathcal{L}_{12}$, $\mathcal{L}_{13}$ and $\mathcal{L}_{23}$ commute with each other, then

$$\begin{aligned}
\boldsymbol{\Theta_1} &= \bigotimes_{1\le k_1\le N_1} e^{\mathcal{A}_{23}\triangle t_n}\Bigg(\bigotimes_{1\le k_2\le N_2} e^{\mathcal{A}_{13}\triangle t_n}\Bigg(\bigotimes_{1\le k_3\le N_3} e^{\mathcal{A}_{12}\triangle t_n} V_1(:,:,k_3)\Bigg)(:,k_2,:)\Bigg)(k_1,:,:),\\
\boldsymbol{\Theta_2} &= \bigotimes_{1\le k_1\le N_1} e^{\mathcal{A}_{23}(\triangle t_n+\triangle t_{n-1})}\Bigg(\bigotimes_{1\le k_2\le N_2} e^{\mathcal{A}_{13}(\triangle t_n+\triangle t_{n-1})}\\
&\quad\Bigg(\bigotimes_{1\le k_3\le N_3} e^{\mathcal{A}_{12}(\triangle t_n+\triangle t_{n-1})} V_2(:,:,k_3)\Bigg)(:,k_2,:)\Bigg)(k_1,:,:). \qquad (69)
\end{aligned}$$

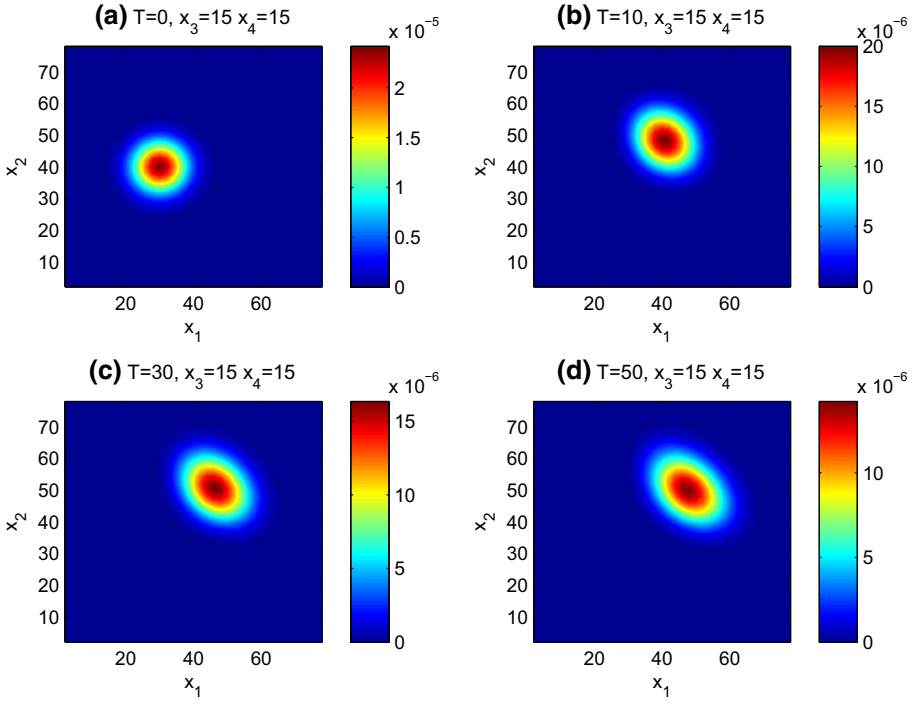

Fig. 15 Distribution of molecular species A and B with $E_A = 15$ and $E_B = 15$, at time $T = 0, 10, 30, 50$. Numerical solutions of (67) using the AcIIF2-WENO scheme with Krylov subspace approximations. $\triangle t = 0.1$. The numbers of spatial grid points are $N_A = 40$, $N_B = 40$, $N_{E_A} = 20$, $N_{E_B} = 20$

If $\mathcal{L}_{12}$, $\mathcal{L}_{13}$ and $\mathcal{L}_{23}$ do not commute with each other, then

$$\begin{aligned}
\mathbf{\Theta_1} &= \bigotimes_{1\le k_3\le N_3} e^{\mathcal{A}_{12}^{k_3}\frac{\triangle t_n}{2}}\left(\bigotimes_{1\le k_2\le N_2} e^{\mathcal{A}_{13}^{k_2}\frac{\triangle t_n}{2}} V_1^*(:,k_2,:)\right)(:,:,k_3),\\
V_1^* &= \bigotimes_{1\le k_1\le N_1} e^{\mathcal{A}_{23}^{k_1}\triangle t_n}\left(\bigotimes_{1\le k_2\le N_2} e^{\mathcal{A}_{13}^{k_2}\frac{\triangle t_n}{2}}\left(\bigotimes_{1\le k_3\le N_3} e^{\mathcal{A}_{12}^{k_3}\frac{\triangle t_n}{2}} V_1(:,:,k_3)\right)(:,k_2,:)\right)(k_1,:,:);
\end{aligned} \tag{70}$$

and

$$\begin{aligned}
\mathbf{\Theta_2} &= \bigotimes_{1\le k_3\le N_3} e^{\mathcal{A}_{12}^{k_3}\frac{(\triangle t_n+\triangle t_{n-1})}{2}}\left(\bigotimes_{1\le k_2\le N_2} e^{\mathcal{A}_{13}^{k_2}\frac{(\triangle t_n+\triangle t_{n-1})}{2}} V_2^*(:,k_2,:)\right)(:,:,k_3),\\
V_2^* &= \bigotimes_{1\le k_1\le N_1} e^{\mathcal{A}_{23}^{k_1}(\triangle t_n+\triangle t_{n-1})}\left(\bigotimes_{1\le k_2\le N_2} e^{\mathcal{A}_{13}^{k_2}\frac{(\triangle t_n+\triangle t_{n-1})}{2}}\right.\\
&\quad\left.\left(\bigotimes_{1\le k_3\le N_3} e^{\mathcal{A}_{12}^{k_3}\frac{(\triangle t_n+\triangle t_{n-1})}{2}} V_2(:,:,k_3)\right)(:,k_2,:)\right)(k_1,:,:).
\end{aligned} \tag{71}$$

(2) For the four dimensional CDR equation, if $\mathcal{L}_{12}$, $\mathcal{L}_{13}$, $\mathcal{L}_{14}$, $\mathcal{L}_{23}$, $\mathcal{L}_{24}$ and $\mathcal{L}_{34}$ commute with each other, then

$$\begin{aligned}\boldsymbol{\Theta_1} = \bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_2\le N_2}} e^{\mathcal{A}_{34}\Delta t_n}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{24}\Delta t_n}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{23}\Delta t_n}\Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{14}\Delta t_n}\Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{13}\Delta t_n}\\ \Bigg(\bigotimes_{\substack{1\le k_3\le N_3\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{12}\Delta t_n} V_1(:,:,k_3,k_4)\Bigg)(:,k_2,:,k_4)\Bigg)(:,k_2,k_3,:)\Bigg)(k_1,:,:,k_4)\Bigg)(k_1,:,k_3,:)\Bigg)\\ (k_1,k_2,:,:),\end{aligned} \tag{72}$$

$$\begin{aligned}\boldsymbol{\Theta_2} = \bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_2\le N_2}} e^{\mathcal{A}_{34}(\Delta t_n+\Delta t_{n-1})}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{24}(\Delta t_n+\Delta t_{n-1})}\\ \Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{23}(\Delta t_n+\Delta t_{n-1})}\Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{14}(\Delta t_n+\Delta t_{n-1})}\\ \Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{13}(\Delta t_n+\Delta t_{n-1})}\Bigg(\bigotimes_{\substack{1\le k_3\le N_3\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{12}(\Delta t_n+\Delta t_{n-1})} V_2(:,:,k_3,k_4)\Bigg)(:,k_2,:,k_4)\Bigg)\\ (:,k_2,k_3,:)\Bigg)(k_1,:,:,k_4)\Bigg)(k_1,:,k_3,:)\Bigg)(k_1,k_2,:,:).\end{aligned} \tag{73}$$

If $\mathcal{L}_{12}$, $\mathcal{L}_{13}$, $\mathcal{L}_{14}$, $\mathcal{L}_{23}$, $\mathcal{L}_{24}$ and $\mathcal{L}_{34}$ do not commute with each other, then

$$\begin{aligned}\boldsymbol{\Theta_1} = \bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_2\le N_2}} e^{\mathcal{A}_{34}^{k_1,k_2}\frac{\Delta t_n}{2}}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{24}^{k_1,k_3}\frac{\Delta t_n}{2}}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{23}^{k_1,k_4}\frac{\Delta t_n}{2}}\Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{14}^{k_2,k_3}\frac{\Delta t_n}{2}}\\ \Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{13}^{k_2,k_4}\frac{\Delta t_n}{2}} V_1^*(:,k_2,:,k_4)\Bigg)(:,k_2,k_3,:)\Bigg)(k_1,:,:,k_4)\Bigg)(k_1,:,k_3,:)\Bigg)(k_1,k_2,:,:),\end{aligned} \tag{74}$$

$$\begin{aligned}V_1^* = \bigotimes_{\substack{1\le k_3\le N_3\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{12}^{k_3,k_4}\Delta t_n}\Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{13}^{k_2,k_4}\frac{\Delta t_n}{2}}\Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{14}^{k_2,k_3}\frac{\Delta t_n}{2}}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{23}^{k_1,k_4}\frac{\Delta t_n}{2}}\\ \Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{24}^{k_1,k_3}\frac{\Delta t_n}{2}}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_2\le N_2}} e^{\mathcal{A}_{34}^{k_1,k_2}\frac{\Delta t_n}{2}} V_1(k_1,k_2,:,:)\Bigg)\end{aligned} \tag{75}$$

$$(k_1,:,k_3,:)\Bigg)(k_1,:,:,k_4)\Bigg)(:,k_2,k_3,:)\Bigg)(:,k_2,:,k_4)\Bigg)(:,:,k_3,k_4).$$

And

$$\begin{aligned}\boldsymbol{\Theta_2} = \bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_2\le N_2}} e^{\mathcal{A}_{34}^{k_1,k_2}\frac{(\Delta t_n+\Delta t_{n-1})}{2}}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{24}^{k_1,k_3}\frac{(\Delta t_n+\Delta t_{n-1})}{2}}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{23}^{k_1,k_4}\frac{(\Delta t_n+\Delta t_{n-1})}{2}}\\ \Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{14}^{k_2,k_3}\frac{(\Delta t_n+\Delta t_{n-1})}{2}}\Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{13}^{k_2,k_4}\frac{(\Delta t_n+\Delta t_{n-1})}{2}} V_2^*(:,k_2,:,k_4)\Bigg)\\ (:,k_2,k_3,:)\Bigg)(k_1,:,:,k_4)\Bigg)(k_1,:,k_3,:)\Bigg)(k_1,k_2,:,:),\end{aligned} \tag{76}$$

$$V_2^* = \bigotimes_{\substack{1\le k_3\le N_3\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{12}^{k_3,k_4}(\triangle t_n+\triangle t_{n-1})}\Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{13}^{k_2,k_4}\frac{(\triangle t_n+\triangle t_{n-1})}{2}}\Bigg(\bigotimes_{\substack{1\le k_2\le N_2\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{14}^{k_2,k_3}\frac{(\triangle t_n+\triangle t_{n-1})}{2}}$$
$$\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_4\le N_4}} e^{\mathcal{A}_{23}^{k_1,k_4}\frac{(\triangle t_n+\triangle t_{n-1})}{2}}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_3\le N_3}} e^{\mathcal{A}_{24}^{k_1,k_3}\frac{(\triangle t_n+\triangle t_{n-1})}{2}}\Bigg(\bigotimes_{\substack{1\le k_1\le N_1\\ 1\le k_2\le N_2}} e^{\mathcal{A}_{34}^{k_1,k_2}\frac{(\triangle t_n+\triangle t_{n-1})}{2}}$$
$$V_2(k_1,k_2,:,:)\Bigg)(k_1,:,k_3,:)\Bigg)(k_1,:,:,k_4)\Bigg)(:,k_2,k_3,:)\Bigg)(:,k_2,:,k_4)\Bigg)(:,:,k_3,k_4). \quad (77)$$

References

1. Ascher, U., Ruuth, S., Wetton, B.: Implicit–explicit methods for time-dependent PDE's. SIAM J. Numer. Anal. **32**, 797–823 (1995)
2. Ashe, H.L., Levine, M.: Local inhibition and long-range enhancement of Dpp signal transduction by Sog. Nature **398**, 427–431 (1999)
3. Beylkin, G., Keiser, J.M., Vozovoi, L.: A new class of time discretization schemes for the solution of nonlinear PDEs. J. Comput. Phys. **147**, 362–387 (1998)
4. Bourlioux, A., Layton, A.T., Minion, M.L.: High-order multi-implicit spectral deferred correction methods for problems of reactive flow. J. Comput. Phys. **189**, 651–675 (2003)
5. Chen, S., Zhang, Y.-T.: Krylov implicit integration factor methods for spatial discretization on high dimensional unstructured meshes: application to discontinuous Galerkin methods. J. Comput. Phys. **230**, 4336–4352 (2011)
6. Christlieb, A., Ong, B., Qiu, J.-M.: Integral deferred correction methods constructed with high order Runge–Kutta integrators. Math. Comput. **79**, 761–783 (2010)
7. Cox, S.M., Matthews, P.C.: Exponential time differencing for stiff systems. J. Comput. Phys. **176**, 430–455 (2002)
8. Dutt, A., Greengard, L., Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations. BIT **40**(2), 241–266 (2000)
9. Fokker, A.D.: Die mittlere energie rotierender elektrischer dipole im strahlungsfeld. Ann. Phys. **348**, 810–820 (1914)
10. Gallopoulos, E., Saad, Y.: Efficient solution of parabolic equations by Krylov approximation methods. SIAM J. Sci. Stat. Comput. **13**(5), 1236–1264 (1992)
11. Gottlieb, S., Shu, C.-W.: Total variation diminishing Runge–Kutta schemes. Math. Comput. **67**, 73–85 (1998)
12. Gottlieb, S., Shu, C.-W., Tadmor, E.: Strong stability preserving high order time discretization methods. SIAM Rev. **43**, 89–112 (2001)
13. Harten, A., Engquist, B., Osher, S., Chakravarthy, S.: Uniformly high order essentially non-oscillatory schemes, III. J. Comput. Phys. **71**, 231–303 (1987)
14. Higham, N.J.: The scaling and squaring method for the matrix exponential revisited. SIAM Rev. **51**(4), 747–764 (2009)
15. Hu, C., Shu, C.-W.: Weighted essentially non-oscillatory schemes on triangular meshes. J. Comput. Phys. **150**, 97–127 (1999)
16. Huang, J., Jia, J., Minion, M.: Arbitrary order Krylov deferred correction methods for differential algebraic equations. J. Comput. Phys. **221**(2), 739–760 (2007)
17. Hochbruck, M., Lubich, C.: On Krylov subspace approximations to the matrix exponential operator. SIAM J. Numer. Anal. **34**, 1911–1925 (1997)
18. Hundsdorfer, W., Verwer, J.: Numerical Solution of Time-Dependent Advection–Diffusion–Reaction Equations. Springer, Berlin (2003)
19. Jiang, G.-S., Shu, C.-W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. **126**, 202–228 (1996)
20. Jiang, T., Zhang, Y.-T.: Krylov implicit integration factor WENO methods for semilinear and fully nonlinear advection–diffusion–reaction equations. J. Comput. Phys. **253**, 368–388 (2013)
21. Jiang, T., Zhang, Y.-T.: Krylov single-step implicit integration factor WENO methods for advection–diffusion–reaction equations. J. Comput. Phys. **311**, 22–44 (2016)
22. Ju, L., Zhang, J., Zhu, L., Du, Q.: Fast explicit integration factor methods for semilinear parabolic equations. J. Sci. Comput. **62**, 431–455 (2015)

23. Kanevsky, A., Carpenter, M.H., Gottlieb, D., Hesthaven, J.S.: Application of implicit–explicit high order Runge–Kutta methods to discontinuous-Galerkin schemes. J. Comput. Phys. **225**(2), 1753–1781 (2007)
24. Kassam, A.-K., Trefethen, L.N.: Fourth-order time stepping for stiff PDEs. SIAM J. Sci. Comput. **26**(4), 1214–1233 (2005)
25. Kennedy, C.A., Carpenter, M.H.: Additive Runge–Kutta schemes for convection–diffusion–reaction equations. Appl. Numer. Math. **44**, 139–181 (2003)
26. Layton, A.T., Minion, M.L.: Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics. J. Comput. Phys. **194**(2), 697–715 (2004)
27. Lander, A., Nie, Q., Wan, F., Zhang, Y.-T.: Localized ectopic expression of Dpp receptors in a Drosophila embryo. Stud. Appl. Math. **123**, 175–214 (2009)
28. Liu, X.-D., Osher, S., Chan, T.: Weighted essentially non-oscillatory schemes. J. Comput. Phys. **115**, 200–212 (1994)
29. Liu, X.F., Nie, Q.: Compact integration factor methods for complex domains and adaptive mesh refinement. J. Comput. Phys. **229**(16), 5692–5706 (2010)
30. Liu, Y., Zhang, Y.-T.: A robust reconstruction for unstructured WENO schemes. J. Sci. Comput. **54**, 603–621 (2013)
31. Lu, D., Zhang, Y.-T.: Krylov integration factor method on sparse grids for high spatial dimension convection–diffusion equations. J. Sci. Comput. **69**, 736–763 (2016)
32. Lu, J., Fang, J., Tan, S., Shu, C.-W., Zhang, M.: Inverse Lax–Wendroff procedure for numerical boundary conditions of convection–diffusion equations. J. Comput. Phys. **317**, 276–300 (2016)
33. Maday, Y., Patera, A.T., Ronquist, E.M.: An operator-integration-factor splitting method for time-dependent problems: application to incompressible fluid flow. J. Sci. Comput. **5**, 263–292 (1990)
34. Minion, M.L.: Semi-implicit spectral deferred correction methods for ordinary differential equations. Commun. Math. Sci. **1**(3), 471–500 (2003)
35. Mizutani, C.M., Nie, Q., Wan, F., Zhang, Y.-T., Vilmos, P., Sousa-Neves, R., Bier, E., Marsh, L., Lander, A.: Formation of the BMP activity gradient in the Drosophila embryo. Dev. Cell **8**, 915–924 (2005)
36. Moler, C., Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. **45**, 3–49 (2003)
37. Nie, Q., Zhang, Y.-T., Zhao, R.: Efficient semi-implicit schemes for stiff systems. J. Comput. Phys. **214**, 521–537 (2006)
38. Nie, Q., Wan, F., Zhang, Y.-T., Liu, X.-F.: Compact integration factor methods in high spatial dimensions. J. Comput. Phys. **227**, 5238–5255 (2008)
39. Planck, M.: Sitzber. Preuss. Akad. Wiss., p. 324 (1917)
40. Risken, H.: The Fokker–Planck Equation: Methods of Solutions and Applications. Springer, Berlin (1996)
41. Shu, C.-W.: TVD time discretizations. SIAM J. Sci. Stat. Comput. **9**, 1073–1084 (1988)
42. Shu, C.-W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes. J. Comput. Phys. **77**, 439–471 (1988)
43. Shu, C.-W.: Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In: Cockburn, B., Johnson, C., Shu, C.-W., Tadmor, E., Quarteroni, A. (eds.) Advanced Numerical Approximation of Nonlinear Hyperbolic Equations. Lecture Notes in Mathematics, vol. 1697. Springer (1998)
44. Sjoberg, P., Lotstedt, P., Elf, J.: Fokker–Planck approximation of the master equation in molecular biology. Comput. Vis. Sci. **12**, 37–50 (2009)
45. Strang, G.: On the construction and comparison of difference schemes. SIAM J. Numer. Anal **8**(3), 506–517 (1968)
46. Ta, C., Wang, D., Nie, Q.: An integration factor method for stochastic and stiff reaction–diffusion systems. J. Comput. Phys. **v295**, 505–522 (2015)
47. Trefethen, L.N., Bau, D.: Numerical Linear Algebra. SIAM, Philadelphia (1997)
48. Verwer, J.G., Sommeijer, B.P., Hundsdorfer, W.: RKC time-stepping for advection–diffusion–reaction problems. J. Comput. Phys. **201**, 61–79 (2004)
49. Wang, D., Zhang, L., Nie, Q.: Array-representation integration factor method for high-dimensional systems. J. Comput. Phys. **258**, 585–600 (2014)
50. Wang, D., Chen, W., Nie, Q.: Semi-implicit integration factor methods on sparse grids for high-dimensional systems, J. Comput. Phys. **v292**, 43–55 (2015)
51. Zhang, Y.-T., Shu, C.-W.: High order WENO schemes for Hamilton–Jacobi equations on triangular meshes. SIAM J. Sci. Comput. **24**, 1005–1030 (2003)
52. Zhang, Y.-T., Shu, C.-W.: Third order WENO scheme on three dimensional tetrahedral meshes. Commun. Comput. Phys. **5**, 836–848 (2009)
53. Zhong, X.: Additive semi-implicit Runge–Kutta methods for computing high-speed nonequilibrium reactive flows. J. Comput. Phys. **128**, 19–31 (1996)