J Sci Comput (2016) 69:736–763
DOI 10.1007/s10915-016-0216-7

Krylov Integration Factor Method on Sparse Grids for High Spatial Dimension Convection–Diffusion Equations

Dong Lu[1] · **Yong-Tao Zhang**[1]

Received: 21 September 2015 / Revised: 29 March 2016 / Accepted: 28 April 2016 /
Published online: 19 May 2016

Abstract Krylov implicit integration factor (IIF) methods were developed in Chen and Zhang (J Comput Phys 230:4336–4352, 2011) for solving stiff reaction–diffusion equations on high dimensional unstructured meshes. The methods were further extended to solve stiff advection–diffusion–reaction equations in Jiang and Zhang (J Comput Phys 253:368–388, 2013). Recently we studied the computational power of Krylov subspace approximations on dealing with high dimensional problems. It was shown that the Krylov integration factor methods have linear computational complexity and are especially efficient for high dimensional convection–diffusion problems with anisotropic diffusions. In this paper, we combine the Krylov integration factor methods with sparse grid combination techniques and solve high spatial dimension convection–diffusion equations such as Fokker–Planck equations on sparse grids. Numerical examples are presented to show that significant computational times are saved by applying the Krylov integration factor methods on sparse grids.

Keywords Implicit integration factor methods · Sparse grids · Krylov subspace approximation · High spatial dimensions · Convection–diffusion equations

1 Introduction

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 [1,5,14–16,20]. In [22], a class of efficient implicit integration factor (IIF) methods were developed for solving systems with

✉ Yong-Tao Zhang
yzhang10@nd.edu

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

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 [22] from implicit exponential time differencing (ETD) methods in [1].

For problems with high spatial dimensions, the major computational challenge in applying the IIF methods is how to deal with matrix exponentials 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 [23]. 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 [28] 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 [4], Krylov subspace approximation is directly applied to the IIF methods for solving high dimensional reaction–diffusion problems. The Krylov IIF methods were further extended to solve high dimensional stiff convection–diffusion–reaction (CDR) equations in [12,13]. Recently in [19] we studied the computational power of Krylov subspace approximations on dealing with high spatial dimension convection–diffusion problems. Systematical numerical comparison and complexity analysis are carried out for the computational efficiency of the two different approaches. It was shown that the Krylov integration factor methods have linear computational complexity and are especially efficient for solving high dimensional convection–diffusion problems with anisotropic diffusions, such as high dimensional Fokker–Planck equations [6,24].

In this paper, we aim at achieving more efficient computations of Krylov IIF schemes than the existing work in the literature by developing the Krylov IIF schemes on sparse grids for high spatial dimension problems. In recent years, sparse-grid has become a major approximation tool for high-dimensional problems. It has been successfully used in many scientific and engineering applications. Discretizations on sparse grids involve $O(N \cdot (\log N)^{d-1})$ degrees of freedom only, where d denotes the underling problem's dimensionality and N is the number of grid points in one coordinate direction. A detailed review on sparse-grid technique can be found in [3]. Sparse-grid techniques were introduced by Zenger [29] in 1991 to reduce the number of degrees of freedom in finite-element calculations. The sparse-grid combination technique, which was introduced in 1992 by Griebel et al. [8], can be seen as a practical implementation of the sparse-grid technique. In the sparse-grid combination technique, the final solution is a linear combination of solutions on semi-coarsened grids, where the coefficients of the combination are chosen such that there is a canceling in leading-order error terms and the accuracy order can be kept to be the same as that on single full grids [8,17,18]. The rest of the paper is organized as following. In Sect. 2, we review the Krylov IIF schemes. Krylov IIF methods on sparse grids are developed in Sect. 3. In Sect. 4, we perform extensive numerical experiments to test the sparse-grid Krylov IIF methods and show significant savings in computational costs by comparisons with single-grid computations. Conclusions are given in Sect. 5.

2 Krylov IIF Methods for CDR Equations

In this section, we briefly review the Krylov IIF methods for high spatial dimension convection–diffusion-reaction equations which were developed in [12]. We consider convection–diffusion-reaction equations

$$u_t + \sum_{i=1}^{d} f_i(u)_{x_i} = \nabla \cdot (\mathbf{D}\nabla u) + r(u), \tag{1}$$

where u is the unknown, $\vec{f_i}$, $i = 1, \ldots, d$ are flux functions in d spatial dimensions respectively, $\mathbf{D}$ is the diffusion matrix and it could be non-constant and have variable entries, and r is the reaction term. Often the CDR models in applications have nonlinear convection and reaction terms, but a linear diffusion term $\nabla \cdot (\mathbf{D}\nabla u)$, where $\mathbf{D}$ is independent of u but could be functions of spatial variables. We use central differences schemes to discretize the diffusion terms, and nonlinear stable weighted essentially non-oscillatory (WENO) [11] or upwind schemes for convection terms. After spatial discretizations, a semi-discretized ODE system

$$\frac{d\vec{U}}{dt} = \vec{F}_d(\vec{U}) + \vec{F}_a(\vec{U}) + \vec{R}(\vec{U}) \tag{2}$$

is obtained. Here $\vec{U} = (u_i)_{1\le i\le N}$, $\vec{F}_d(\vec{U}) = (\hat{F}_{d\,i}(\vec{U}))_{1\le i\le N}$, $\vec{F}_a(\vec{U}) = (\hat{F}_{a\,i}(\vec{U}))_{1\le i\le N}$, $\vec{R} = (r(u_i))_{1\le i\le N}$. 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 its each component $\hat{F}_{d\,i}$ 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 schemes. $\vec{F}_a(\vec{U})$ is the approximation for the nonlinear advection terms by WENO scheme or upwind scheme, and its each component $\hat{F}_{a\,i}$ is a nonlinear function of several numerical values on the WENO or upwind approximation stencil. $\vec{R}(\vec{U})$ is the nonlinear reaction term, and its each component $r(u_i)$ is a nonlinear function which only depends on numerical value at one grid point. In [12], 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, we focus on the problems with linear diffusions of either constant or variable diffusion coefficients, i.e., $\vec{F}_d(\vec{U}) = C\vec{U}$. IIF methods for (2) are constructed by exactly integrating the linear part of the system. Directly multiply (2) 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

$$\begin{aligned}\vec{U}(t_{n+1}) = {} & e^{C\Delta t_n}\vec{U}(t_n) + e^{C\Delta t_n}\int_0^{\Delta t_n} e^{-C\tau}\vec{F}_a(\vec{U}(t_n+\tau))d\tau \\ & + e^{C\Delta t_n}\int_0^{\Delta t_n} e^{-C\tau}\vec{R}(\vec{U}(t_n+\tau))d\tau. \end{aligned} \tag{3}$$

Two of the nonlinear terms in (3) 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 approximations 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 nonlinear convection term $e^{-C\tau}\vec{F}_a(\vec{U}(t_n+\tau))$ is approximated explicitly by an $(r-1)$-th order Lagrange polynomial with interpolation points at $t_n, t_{n-1}, \ldots, t_{n+1-r}$. The r-th order IIF scheme for CDR equations is obtained as

$$\vec{U}_{n+1} = e^{C\Delta t_n}\vec{U}_n + \Delta t_n \left\{ \alpha_{n+1}\vec{R}(\vec{U}_{n+1}) + \sum_{i=2-r}^{0} \alpha_{n+i} e^{C(\Delta t_n - \tau_i)}\vec{R}(\vec{U}_{n+i}) \right. \\ \left. + \sum_{i=1-r}^{0} \beta_{n+i} e^{C(\Delta t_n - \tau_i)}\vec{F}_a(\vec{U}_{n+i}) \right\}, \tag{4}$$

where the coefficients

$$\alpha_{n+i} = \frac{1}{\Delta t_n}\int_0^{\Delta t_n} \prod_{j=2-r, j\neq i}^{1} \frac{\tau - \tau_j}{\tau_i - \tau_j} d\tau, \qquad i = 1, 0, -1, \ldots, 2-r; \tag{5}$$

$$\beta_{n+i} = \frac{1}{\Delta t_n}\int_0^{\Delta t_n} \prod_{j=1-r, j\neq i}^{0} \frac{\tau - \tau_j}{\tau_i - \tau_j} d\tau, \qquad i = 0, -1, -2, \ldots, 1-r. \tag{6}$$

$\tau_1 = \Delta t_n, \tau_0 = 0, \tau_i = -\sum_{k=i}^{-1} \Delta t_{n+k}$ 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

$$\vec{U}_{n+1} = e^{C\Delta t_n}\vec{U}_n + \Delta t_n \left\{ \alpha_{n+1}\vec{R}(\vec{U}_{n+1}) + \alpha_n e^{C\Delta t_n}\vec{R}(\vec{U}_n) \right. \\ \left. + \beta_{n-1} e^{C(\Delta t_n + \Delta t_{n-1})}\vec{F}_a(\vec{U}_{n-1}) + \beta_n e^{C\Delta t_n}\vec{F}_a(\vec{U}_n) \right\}, \tag{7}$$

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

$$\vec{U}_{n+1} = e^{C\Delta t_n}\vec{U}_n + \Delta t_n \left\{ \alpha_{n+1}\vec{R}(\vec{U}_{n+1}) + \alpha_n e^{C\Delta t_n}\vec{R}(\vec{U}_n) + \alpha_{n-1} e^{C(\Delta t_n + \Delta t_{n-1})}\vec{R}(\vec{U}_{n-1}) \right. \\ \left. + \beta_{n-2} e^{C(\Delta t_n + \Delta t_{n-1} + \Delta t_{n-2})}\vec{F}_a(\vec{U}_{n-2}) + \beta_{n-1} e^{C(\Delta t_n + \Delta t_{n-1})}\vec{F}_a(\vec{U}_{n-1}) + \beta_n e^{C\Delta t_n}\vec{F}_a(\vec{U}_n) \right\}, \tag{8}$$

where

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

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

$$\alpha_{n-1} = -\frac{\Delta t_n^2}{6\Delta t_{n-1}(\Delta t_{n-1} + \Delta t_n)},$$

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

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

$$\beta_{n-2} = \frac{1}{\Delta t_{n-2}(\Delta t_{n-1} + \Delta t_{n-2})}\left(\frac{\Delta t_n^2}{3} + \frac{\Delta t_n \Delta t_{n-1}}{2}\right).$$

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 (4). 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. Here we review Krylov subspace approximation method which was applied to IIF schemes in [4] and for solving CDR equations in [12]. The computational power of Krylov subspace approximation method for IIF schemes on high spatial dimension problems has been studied in [19] which shows its high efficiency. 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 (4). Follow the literature (e.g. [7,21]), 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

$$K_M = \text{span}\{v, Cv, C^2v, \ldots, C^{M-1}v\}. \tag{9}$$

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 [27] as the following.

```
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 $K_M = \text{span}\{v_1, v_2, \ldots, v_j\}$, 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

$$H_M = V_M^T C V_M. \tag{10}$$

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 closest approximation to $e^{C\Delta t}v$ from 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. Use the fact of (10), we have the approximation

$$e^{C\Delta t} v \simeq \beta V_M e^{H_M \Delta t} e_1. \tag{11}$$

Thus the large $e^{C\Delta t}$ matrix exponential problem is replaced with a much smaller $e^{H_M \Delta t}$ problem. The small matrix exponential $e^{H_M \Delta t}$ will be computed using a scaling and squaring algorithm with a Padé approximation with only a small computational cost, see [7,10,21]. Then the Krylov approximations are directly applied in schemes (4), (7) or (8) to obtain Krylov IIF schemes for CDR equations [12]. The r-th order Krylov IIF scheme for CDR equations has the following form

$$\begin{aligned}\vec{U}_{n+1} = {} & \Delta t_n \alpha_{n+1} \vec{R}(\vec{U}_{n+1}) + \gamma_{0,n} V_{M,0,n} e^{H_{M,0,n}\Delta t_n} e_1 \\ & + \Delta t_n \left(\beta_{n+1-r} \gamma_{1-r,n} V_{M,1-r,n} e^{H_{M,1-r,n}(\Delta t_n - \tau_{1-r})} e_1 \right. \\ & \left. + \sum_{i=2-r}^{-1} \gamma_{i,n} V_{M,i,n} e^{H_{M,i,n}(\Delta t_n - \tau_i)} e_1 \right),\end{aligned} \tag{12}$$

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

$$\begin{aligned}\vec{U}_{n+1} = {} & \frac{1}{2} \Delta t_n \vec{R}(\vec{U}_{n+1}) + \gamma_{0,n} V_{M,0,n} e^{H_{M,0,n}\Delta t_n} e_1 \\ & - \frac{(\Delta t_n)^2}{2\Delta t_{n-1}} \left(\gamma_{-1,n} V_{M,-1,n} e^{H_{M,-1,n}(\Delta t_n + \Delta t_{n-1})} e_1 \right),\end{aligned} \tag{13}$$

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 (Krylov IIF3) scheme has the form

$$\begin{aligned}\vec{U}_{n+1} = {} & \frac{2\Delta t_n + 3\Delta t_{n-1}}{6(\Delta t_n + \Delta t_{n-1})} \Delta t_n \vec{R}(\vec{U}_{n+1}) + \gamma_{0,n} V_{M,0,n} e^{H_{M,0,n}\Delta t_n} e_1 \\ & + \Delta t_n \left(\frac{2(\Delta t_n)^2 + 3\Delta t_n \Delta t_{n-1}}{6\Delta t_{n-2}(\Delta t_{n-1} + \Delta t_{n-2})} \gamma_{-2,n} V_{M,-2,n} e^{H_{M,-2,n}(\Delta t_n + \Delta t_{n-1} + \Delta t_{n-2})} e_1 \right. \\ & \left. + \gamma_{-1,n} V_{M,-1,n} e^{H_{M,-1,n}(\Delta t_n + \Delta t_{n-1})} e_1 \right),\end{aligned} \tag{14}$$

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. (8) for values of $\alpha_n, \beta_n, \alpha_{n-1}, \beta_{n-1}$.

As that pointed out in [12], 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 Theoretical analysis including linear stability and error analysis of the IIF schemes for convection–diffusion-reaction equations is given in [12]. As an example, we present the linear stability analysis of the second order IIF scheme for convection–diffusion-reaction equations in the Appendix section of this paper.

3 Krylov IIF Schemes on Sparse Grids

To achieve further efficiency in solving the CDR Eq. (1) on high spatial dimensions by Krylov IIF schemes, we present the Krylov IIF schemes on sparse grids by sparse-grid combination technique. The basic idea of sparse-grid combination technique is that by combining several solutions on different semi-coarsened grids (sparse grids), a final solution on the most refined mesh is obtained. The most refined mesh is corresponding to the usual single full grid. Since the PDEs are solved on semi-coarsened grids which have much fewer grid points than the single full grid, computation costs are saved a lot. The final solution obtained by sparse-grid combination technique is required to have the similar accuracy as that by solving the PDEs directly on a single full grid. For example see [8,17,18].

Here we use two dimensional (2D) case as the example to illustrate the idea. Higher dimensional cases are similar. Consider a 2D domain $[a,b]^2$. The construction of semi-coarsened grids is as follows. We first partition the domain into the coarsest mesh, which is called a root grid $\Omega^{0,0}$ with N_r cells in each direction. The root grid mesh size is $H = \frac{b-a}{N_r}$. The multi-level refinement on the root grid is performed to obtain a family of semi-coarsened grids $\{\Omega^{l_1,l_2}\}$. The semi-coarsened grid $\{\Omega^{l_1,l_2}\}$ has mesh sizes $h_{l_1} = 2^{-l_1}H$ in the x direction and $h_{l_2} = 2^{-l_2}H$ in the y direction, where $l_1 = 0, 1, \ldots, N_L$, $l_2 = 0, 1, \ldots, N_L$, see Fig. 1. Superscripts l_1, l_2 indicate the level of refinement relative to the root grid $\Omega^{0,0}$, and N_L indicates the finest level. Therefore, our finest grid is Ω^{N_L,N_L} with mesh size $h = 2^{-N_L}H$ for both x and y directions.

To solve Eq. (1), we will use the second order Krylov IIF (Krylov IIF2) method (13) or the third order Krylov IIF (Krylov IIF3) scheme (14) for time discretization. Spatial discretizations are the classical second or fourth order central schemes for diffusion terms, and the third order WENO scheme or the upwind scheme for convection terms. Following the spare-grid combination techniques, rather than on a single full grid, the PDE (1) is solved on the following $(2N_L+1)$ sparse grids $\{\Omega^{l_1,l_2}\}_I$:

$$\left\{\Omega^{0,N_L}, \Omega^{1,N_L-1}, \ldots, \Omega^{N_L-1,1}, \Omega^{N_L,0}\right\} \text{ and } \left\{\Omega^{0,N_L-1}, \Omega^{1,N_L-2}, \ldots, \Omega^{N_L-2,1}, \Omega^{N_L-1,0}\right\}.$$

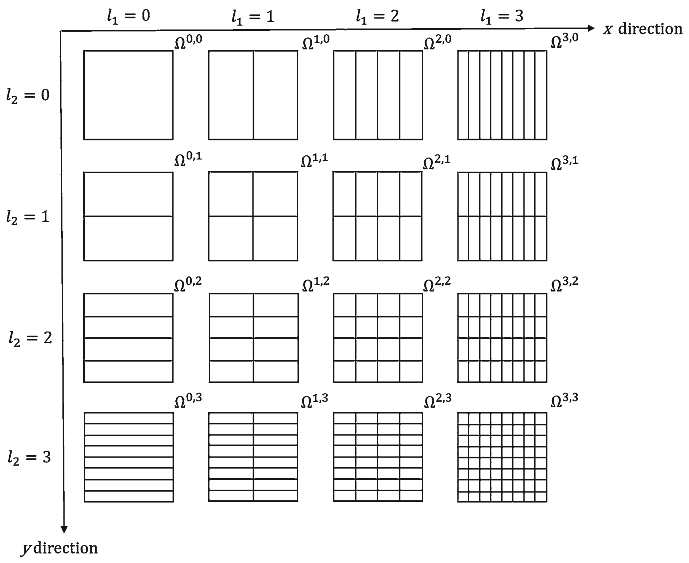

Fig. 1 Semi-coarsened sparse grids $\{\Omega^{l_1,l_2}\}$ with the finest level $N_L = 3$

And I denotes the index set

$$I = \left\{(l_1, l_2) | l_1 + l_2 = N_L \quad \text{or} \quad l_1 + l_2 = N_L - 1\right\}.$$

By carrying out time marching of the PDE using Krylov IIF schemes on these $(2N_L + 1)$ sparse grids, we obtain $(2N_L + 1)$ sets of numerical solutions $\{U^{l_1,l_2}\}_I$ (one set of numerical solution is obtained on each sparse grid). The next step is to combine solutions on sparse grids to obtain the final solution on the finest grid Ω^{N_L,N_L}. The key point here is that the PDE is never solved directly on Ω^{N_L,N_L} in order to save computational costs. To extend numerical solutions on sparse grids to that on the finest grid, we apply a prolongation operator P^{N_L,N_L} (defined in the spare-grid combination techniques [8,17,18]) on each sparse grid solution U^{l_1,l_2} to obtain $(2N_L + 1)$ solutions on the finest grid Ω^{N_L,N_L}. And finally, these solutions are combined to form the final solution $\hat{U}^{N_L,N_L}$ on Ω^{N_L,N_L}.

Next we provide details on the prolongation operator P^{N_L,N_L}. Prolongation operator P^{N_L,N_L} maps numerical solutions $\{U^{l_1,l_2}\}_I$ on sparse grids onto the finest grid Ω^{N_L,N_L}. And a prolongation operator is basically an interpolation operator. For example, U^{l_1,l_2} is numerical solution on Ω^{l_1,l_2}, then $P^{N_L,N_L} U^{l_1,l_2}$ gives numerical values on the most refined mesh Ω^{N_L,N_L}. For the 2D case, first in one direction(e.g. the x direction), we construct $(N_r 2^{l_1-1})$ quadratic interpolation polynomials $P_i^2(x)$, $i = 1, \ldots, N_r 2^{l_1-1}$, by the third order Lagrange interpolation. Each interpolation uses three adjacent grid points to construct a quadratic polynomial. Note that a higher order interpolation is needed for comparable numerical accuracy as that of the numerical schemes, if higher order accuracy numerical schemes are used to solve PDEs on sparse grids (see [8,17,18]). Then we evaluate $P_i^2(x)$ on the grid points of Ω^{N_L,l_2}, which is the most refined meshes in the x direction. Next, in the other direction (e.g. the y direction), we construct $(N_r 2^{l_2-1})$ quadratic interpolation polynomials $P_j^2(y)$, $j = 1, \ldots, N_r 2^{l_2-1}$, and evaluate them on the grid points of Ω^{N_L,N_L}. At last we get $P^{N_L,N_L} U^{l_1,l_2}$, defined on the finest grid Ω^{N_L,N_L}. We summarize the algorithm of Krylov IIF scheme on sparse grids as following.

Algorithm: Krylov IIF Scheme with Sparse-Grid Combination Technique

- Step 1: Restrict the initial condition $u(x, y, 0)$ to $(2N_L + 1)$ sparse grids $\{\Omega^{l_1,l_2}\}_I$ defined above;
- Step 2: On each sparse grid Ω^{l_1,l_2}, solve the Eq. (1) by Krylov IIF scheme to reach the final time T. Then we get $(2N_L + 1)$ sets of solutions $\{U^{l_1,l_2}\}_I$;
- Step 3: At the final time T,
  - on each grid Ω^{l_1,l_2}, apply prolongation operator P^{N_L,N_L} on U^{l_1,l_2}. Then we get $P^{N_L,N_L}U^{l_1,l_2}$, defined on the most refined mesh Ω^{N_L,N_L}.
  - do the combination to get the final solution

$$\hat{U}^{N_L,N_L} = \sum_{l_1+l_2=N_L} P^{N_L,N_L}U^{l_1,l_2} - \sum_{l_1+l_2=N_L-1} P^{N_L,N_L}U^{l_1,l_2}. \tag{15}$$

For three dimensional (3D) or higher dimensional problems, the algorithm is similar although prolongation operations are performed in additional spatial directions. The sparse-grid combination formula for higher dimensional cases can be found in the literature, e.g. in [8]. Specifically the 3D formula is

$$\begin{aligned}\hat{U}^{N_L,N_L,N_L} = &\sum_{l_1+l_2+l_3=N_L} P^{N_L,N_L,N_L}U^{l_1,l_2,l_3} - 2\sum_{l_1+l_2+l_3=N_L-1} P^{N_L,N_L,N_L}U^{l_1,l_2,l_3}\\ &+ \sum_{l_1+l_2+l_3=N_L-2} P^{N_L,N_L,N_L}U^{l_1,l_2,l_3}.\end{aligned} \tag{16}$$

4 Numerical Experiments

In this section, we use various numerical examples to show the computational efficiency of Krylov IIF schemes with sparse-grid combination technique on sparse grids, by comparing to the same schemes on regular grids. Examples include reaction–diffusion equations without convection, and convection–diffusion problems. Equations with different types of diffusions are tested, namely, equations with constant diffusion coefficients, with variable diffusion coefficients, and with/without cross derivatives. We test examples with an exact solution and a three dimensional Fokker–Planck equation which has broad applications. For each example, we compute numerical accuracy errors and convergence orders of the schemes, and record CPU times. We also list the ratios of corresponding CPU times on an $N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2}$ mesh, to study the computational complexity of the schemes on sparse grids and on regular single grids. Here in the data Tables and texts of this section, $N_h \times N_h$ (or a coarser one $\frac{N_h}{2} \times \frac{N_h}{2}$ in the text description) denotes the most refined mesh in sparse grids or a regular mesh in single grid computations. Since Krylov IIF schemes remove time step size constraint of stiff diffusion and reaction terms, the time step sizes can be taken as that for a pure hyperbolic problem, i.e., proportional to the spatial grid sizes. For computations on sparse grids, PDEs are evolved on different semi-coarsened sparse grids. How to choose time step sizes for each individual time evolution is an interesting question. Via numerical experiments, we found that for the Example 1, which is a relatively simple constant diffusion problem without cross derivatives and convection terms, if the grids are uniform, the time step sizes are taken to be proportional to the minimum spatial grid size of each spatial direction on each individual semi-coarsened sparse grid Ω^{l_1,l_2}, i.e. $\triangle t = c \times \min(h_{l_1}, h_{l_2})$. c is a constant. Hence time step sizes may take different values for solving the PDE on different semi-coarsened sparse grid, although each individual time evolution reaches the same final time. The resulting numerical accuracy orders keep the desired values. However for more complicated problems such as Examples 2, 3, 4 and 5, time step sizes on all semi-coarsened sparse grids need to take the same value. It is determined by the spatial grid size h of the most refined grid Ω^{N_L,N_L}, namely, it is proportional to h with $\triangle t = c \times h$. Numerical experiments show that the desired numerical accuracy orders are reached with time step sizes taken this way. Hence for a general problem, the numerical experiments in this paper suggest that time step sizes on all semi-coarsened sparse grids should be determined by the spatial grid size h of the most refined grid Ω^{N_L,N_L}. All of the numerical simulations in this paper are performed on a 2.3 GHz, 16GB RAM Linux workstation.

Example 1 **(Isotropic diffusion problems).** We consider a reaction–diffusion problem with isotropic diffusion

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

First we test the two dimensional case defined on the domain $\Omega = \{0 < x < 2\pi, 0 < y < 2\pi\}$, subject to periodic boundary conditions, i.e.,

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

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 till final time $T = 1$ by the second order Krylov IIF scheme (Krylov IIF2) (13) on both single grids and sparse grids, and compare their computational efficiency. We present the L^∞ errors, L^2 errors, the corresponding numerical accuracy orders, and CPU times on successively refined meshes to show the efficiency of computations on sparse grids. There are two different ways to refine meshes for computations on sparse grids. One way is to refine the root grid $\Omega^{0,0}$, and keep the number of semi-coarsened sparse-grid levels (total $N_L + 1$ levels) unchanged. For example, sparse-grid with a 10×10 root grid and $N_L = 3$ has the finest mesh 80×80. If the root grid is refined once to be 20×20, with $N_L = 3$ unchanged we can obtain the finest mesh 160×160. The other way is to increase the number of levels (refine level), and keep the root grid $\Omega^{0,0}$ unchanged. For example, if we increase $N_L = 3$ to $N_L = 4$ with a 10×10 root grid, the finest mesh which is 80×80 for the $N_L = 3$ case is refined to be 160×160 for the $N_L = 4$ case. The numerical errors, accuracy orders, and CPU times are listed in Table 1 for computations by the Krylov IIF2 scheme on single-grid and sparse-grid. The computations on single-grid, and sparse grids with two different mesh refinement methods achieve the similar numerical errors and the second order accuracy. However, computations on sparse-grid are much more efficient than those on single-grid. Comparing the CPU times in Table 1, we can see that for computations on sparse grids with the first mesh refinement method (i.e., refine root grids), more than 50 % computation time can be saved, especially on more refined meshes. Moreover, the CPU time savings are even more significant for computations on sparse grids with the second mesh refinement method (i.e., refine level). As that shown in Table 1, 92 % CPU time can be saved for the computation on a 640×640 mesh. We also list the ratios of corresponding CPU times on an $N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2}$, to study the computational complexity of the methods. For this two dimensional time dependent parabolic problem, we achieve large time step size computation $\triangle t = O(h)$ by using the Krylov IIF method. A linear computational complexity method should have the CPU time ratio be 8 for a complete time evolution. The CPU time ratios shown in Table 1 for computations on single-grid verify its linear computational complexity. For computations on sparse grids, the CPU time ratio is around 8 for the refining root grid case, and around 4 for

Table 1 Example 1, 2D case, Krylov IIF2 scheme, comparison of numerical errors and CPU times for computations on single-grid and sparse-grid

		$N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Single-grid								
		80×80	1.86×10^{-4}		8.74×10^{-5}		3.56	
		160×160	4.66×10^{-5}	2.00	2.25×10^{-5}	1.96	27.34	7.68
		320×320	1.16×10^{-5}	2.00	5.71×10^{-6}	1.98	219.15	8.02
		640×640	2.91×10^{-6}	2.00	1.44×10^{-6}	1.99	1828.21	8.34
N_r	N_L	$N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Sparse-grid, refine root grids								
10	3	80×80	1.83×10^{-4}		9.15×10^{-5}		2.50	
20	3	160×160	4.57×10^{-5}	2.00	2.29×10^{-5}	2.00	14.74	5.91
40	3	320×320	1.14×10^{-5}	2.00	5.71×10^{-6}	2.00	104.47	7.09
80	3	640×640	2.86×10^{-6}	2.00	1.43×10^{-6}	2.00	817.09	7.82
Sparse-grid, refine level								
10	3	80×80	1.83×10^{-4}		9.15×10^{-5}		2.50	
10	4	160×160	4.57×10^{-5}	2.00	2.29×10^{-5}	2.00	9.33	3.74
10	5	320×320	1.08×10^{-5}	2.09	5.38×10^{-6}	2.09	36.03	3.86
10	6	640×640	2.68×10^{-6}	2.00	1.34×10^{-6}	2.00	142.53	3.96

Final time $T = 1.0$. For single-grid computations, $\triangle t = 0.5\,h$. For sparse-grid computations, $\triangle t = 0.5\min(h_{l_1}, h_{l_2})$ on each semi-coarsened sparse grid Ω^{l_1,l_2}. N_r: number of cells in each spatial direction of a root grid. N_L: the finest level in a sparse-grid computation. CPU: CPU time for a complete simulation. "Ratio" is the ratio of corresponding CPU times on an $N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2}$ mesh. CPU time unit: seconds

the refining level case. Hence the computational complexity on sparse-grid is also linear for the first mesh refinement method, and much better than linear for the second mesh refinement method.

We perform the same test for the third order scheme. The third order Krylov IIF scheme (Krylov IIF3) (14) on single-grid and the same scheme with sparse-grid combination technique are used to compute this two-dimensional problem till final time $T = 1$. Again we use two different ways to refine meshes on sparse grids. The numerical results are reported in Table 2. Comparable numerical errors and fourth order accuracy order are obtained for all three different approaches. The fourth order accuracy order here is due to the fourth order central difference scheme to discretize the diffusion terms. It is obvious here that the spatial errors dominate and are larger than the temporal errors. Again, computations on sparse-grid are more efficient than those on single-grid as that shown in Table 2. Especially for the the second mesh refinement method (i.e., refine level), 82 % CPU time can be saved for the computation on a 640×640 mesh. In terms of computational complexity, the Krylov IIF3 scheme shows a linear computational complexity on single-grid as that for the second order scheme. The computational complexity of the Krylov IIF3 scheme on sparse-grid is also linear for the first mesh refinement method, and much better than linear for the second mesh refinement method.

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 $u(x, y, z, 0) =$

Table 2 Example 1, 2D case, Krylov IIF3 scheme, comparison of numerical errors and CPU times for computations on single-grid and sparse-grid

		$N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Single-grid								
		80×80	8.82×10^{-7}		4.41×10^{-7}		7.45	
		160×160	5.63×10^{-8}	4.00	2.82×10^{-8}	3.97	62.08	8.33
		320×320	3.56×10^{-9}	4.00	1.78×10^{-9}	3.98	504.81	8.13
		640×640	2.33×10^{-10}	3.94	1.17×10^{-10}	3.93	3743.59	7.42
N_r	N_L	$N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Sparse-grid, refine root grids								
10	3	80×80	8.82×10^{-7}		4.41×10^{-7}		7.85	
20	3	160×160	5.63×10^{-8}	3.97	2.82×10^{-8}	3.97	48.09	6.13
40	3	320×320	3.56×10^{-9}	3.98	1.78×10^{-9}	3.98	356.76	7.42
80	3	640×640	2.26×10^{-10}	3.98	1.13×10^{-10}	3.98	2850.46	7.99
Sparse-grid, refine level								
10	3	80×80	8.82×10^{-7}		4.41×10^{-7}		7.85	
10	4	160×160	5.63×10^{-8}	3.97	2.82×10^{-8}	3.97	34.26	4.36
10	5	320×320	3.56×10^{-9}	3.98	1.78×10^{-9}	3.98	152.83	4.46
10	6	640×640	2.26×10^{-10}	3.98	1.13×10^{-10}	3.98	688.69	4.51

Final time $T = 1.0$. For single-grid computations, $\Delta t = 0.5\ h$. For sparse-grid computations, $\Delta t = 0.5 \min(h_{l_1}, h_{l_2})$ on each semi-coarsened sparse grid Ω^{l_1,l_2}. N_r: number of cells in each spatial direction of a root grid. N_L: the finest level in a sparse-grid computation. CPU: CPU time for a complete simulation. "Ratio" is the ratio of corresponding CPU times on an $N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2}$ mesh. CPU time unit: seconds

$\cos(x) + \cos(y) + \cos(z)$. The exact solution is $u(x, y, z, t) = e^{-0.1t}(\cos(x) + \cos(y) + \cos(z))$. We compute the problem till final time $T = 1$. The numerical errors, accuracy orders, CPU times for a complete simulation and the ratios of CPU times on an $N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2}$ mesh are listed in Table 3 for the Krylov IIF2 scheme on single-grid and on sparse-grid with two different mesh refinement approaches. The computation on the $640 \times 640 \times 640$ single-grid can *not* be performed due to the computer memory restriction. Computer memory is saved significantly by using sparse-grid and computations can be successfully done for the $640 \times 640 \times 640$ mesh case. We observe that all computations give comparable 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 single-grid computation, the Krylov IIF2 scheme's CPU time ratios shown in Table 3 verify its linear computational complexity. We also observe that the Krylov IIF2 scheme on sparse-grid with the first mesh refinement method (refining root grid) has CPU time ratio be around 16, so it also has linear computational complexity. And computations on sparse-grid with the second mesh refinement method (i.e., refining level) has CPU time ratio be around 5 as that shown in Table 3, hence its computational complexity is much better than linear. In terms of computational efficiency, the savings of CPU times and improvement of the efficiency for solving this three dimensional problem on sparse-grid are more significant than that for two dimensional problems, as that shown in Table 3. For example, we compare the CPU times for

Table 3 Example 1, 3D case, Krylov IIF2 scheme, comparison of numerical errors and CPU times for computations on single-grid and sparse-grid

		$N_h \times N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Single-grid								
		$80 \times 80 \times 80$	5.50×10^{-5}		2.43×10^{-5}		850.24	
		$160 \times 160 \times 160$	1.53×10^{-5}	1.85	6.58×10^{-6}	1.88	13,637.13	16.04
		$320 \times 320 \times 320$	4.06×10^{-6}	1.91	1.71×10^{-6}	1.94	225,543.28	16.54
N_r	N_L	$N_h \times N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Sparse-grid, refine root grids								
10	3	$80 \times 80 \times 80$	5.40×10^{-5}		2.39×10^{-5}		89.35	
20	3	$160 \times 160 \times 160$	1.50×10^{-5}	1.85	6.49×10^{-6}	1.88	1494.67	16.73
40	3	$320 \times 320 \times 320$	3.99×10^{-6}	1.91	1.69×10^{-6}	1.94	25,244.30	16.89
80	3	$640 \times 640 \times 640$	1.03×10^{-6}	1.95	4.30×10^{-7}	1.97	422,502.00	16.74
Sparse-grid, refine level								
10	3	$80 \times 80 \times 80$	5.40×10^{-5}		2.39×10^{-5}		89.35	
10	4	$160 \times 160 \times 160$	1.54×10^{-5}	1.81	6.68×10^{-6}	1.84	453.00	5.07
10	5	$320 \times 320 \times 320$	4.21×10^{-6}	1.87	1.78×10^{-6}	1.91	2321.95	5.13
10	6	$640 \times 640 \times 640$	1.09×10^{-6}	1.95	4.53×10^{-7}	1.97	12,730.90	5.48

Final time $T = 1.0$. For single-grid computations, $\triangle t = h/3$. For sparse-grid computations, $\triangle t = 1/3 \min(h_{l_1}, h_{l_2}, h_{l_3})$ on each semi-coarsened sparse grid Ω^{l_1,l_2,l_3}. N_r: number of cells in each spatial direction of a root grid. N_L: the finest level in a sparse-grid computation. CPU: CPU time for a complete simulation. "Ratio" is the ratio of corresponding CPU times on an $N_h \times N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2} \times \frac{N_h}{2}$ mesh. CPU time unit: seconds

computations on a $320 \times 320 \times 320$ mesh. With the number of cells in each spatial direction of a root grid $N_r = 40$ and the finest level $N_L = 3$, the CPU time for the computation on sparse-grid (25,244.30 s) is about 1/10 of that on a single-grid (225,543.28 s). And with a coarser root grid $N_r = 10$ and the finest level $N_L = 5$, the CPU time for the computation on sparse-grid (2321.95 s) is about 1/100 of that on a single-grid (225,543.28 s), so 99 % CPU time is saved. Since higher dimensional problems generally demand much more computational time than low dimensional ones, the efficiency achieved here verifies advantages of Krylov IIF schemes designed on sparse-grid for solving higher dimensional problems.

We use the Krylov IIF2 scheme here as an example to further analyze the computational complexity on singe-grid and sparse-grid. We estimate the number of multiplication and division operations in one time step for computations on single-grid and sparse-grid for the 2D case. The number of operations is $(M^2 + 14M + 9)[(1 + 1.5N_L)2^{N_L} N_r^2 + (6 \cdot 2^{N_L} - 4)N_r + 2N_L + 1]$ for the computation on sparse-grid with an $N_r \times N_r$ root grid and N_L fine levels. For the computation on an $N_h \times N_h$ single-grid, the number of operations is $(M^2+14M+9)N_h^2$. M is the dimension of Krylov subspace, and $M = 25$ here. In Table 4, we list the number of operations for these grids used in this example. It shows that computations on sparse-grid need fewer operations than that on single-grid, especially the savings of operations are very significant for computations on sparse-grid with the second mesh refinement method (i.e., refining level).

It is interesting to compare the computational efficiency of Krylov IIF method on singe-grid and sparse-grid studied in this paper with a fully implicit scheme with an advanced linear sys-

Table 4 Example 1, 2D case, Krylov IIF2 scheme, comparison of the number of multiplication and division operations in one time step for computations on single-grid and sparse-grid

$N_h \times N_h$	Single-grid	Sparse-grid (refine root grid)	Sparse-grid (refine level)
80×80	6,297,600	4,769,448	4,769,448
160×160	25,190,400	18,191,208	11,934,936
320×320	100,761,600	71,012,328	28,625,544
640×640	403,046,400	280,564,968	66,727,992

tem solver such as a multigrid method. As an example, we apply the Crank-Nicolson scheme [9] in discretizing the 2D case here. A multigrid solver (the Two-Grid correction scheme) [2] is implemented to solve the linear system at every time step. We take the number of relaxation times to be 3 in the Two-Grid correction scheme [2]. The results including numerical errors, accuracy orders and CPU times are reported in Table 5. The Crank–Nicolson scheme with the Two-Grid correction multigrid solver for solving this problem has similar numerical errors and the second order accuracy order as the Krylov IIF2 scheme on singe-grid and sparse-grid (Table 1). In terms of computational efficiency, the Crank–Nicolson scheme with the Two-Grid correction multigrid solver is more efficient on relatively coarse mesh (e.g. the 80×80 mesh) than the Krylov IIF2 scheme. However, on more refined meshes the Krylov IIF2 scheme is more efficient. Especially, the improvement of efficiency is very obvious for computations on sparse-grid. More systematic comparisons of Krylov IIF schemes and fully implicit schemes with efficient multigrid solvers will be carried out in our future research.

Remark The numerical methods with sparse grid combination technique are presented using uniform rectangular meshes in this paper. The approach can be straightforwardly implemented on non-uniform rectangular meshes. Here we test the Krylov IIF2 scheme with sparse grid combination technique on non-uniform rectangular meshes by applying it in solving the 2D case of this example. The non-uniform meshes are obtained by randomly perturbing x-coordinates and y-coordinates of a uniform mesh in the range of $(-0.3h, 0.3h)$. We use five points in one spatial direction to approximate the diffusion terms on non-uniform meshes. Hence the approximations to the diffusion terms are on a centered stencil and the accuracy order for the diffusion terms is 3. The numerical errors, accuracy orders, and CPU times are listed in Table 6 for computations by the Krylov IIF2 scheme on single-grid and sparse-grid. We draw consistent conclusion with computations on uniform meshes. Namely, the computations on single-grid, and sparse grids with two different mesh refinement methods achieve the similar numerical errors, while computations on sparse-grid are much more efficient than those on single-grid.

Example 2 (**A 3D problem with anisotropic diffusion and constant diffusion coefficients**). We consider a three-dimensional reaction–diffusion problem with cross-derivative diffusion terms and constant diffusion coefficients

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

Table 5 Example 1, 2D case, Crank–Nicolson scheme with a multigrid solver (the Two-Grid correction scheme) for the linear systems

$N_h \times N_h$	L^∞ Error	Order	L^2 error	Order	CPU(s)
80×80	1.86×10^{-4}		9.29×10^{-5}		1.81
160×160	4.65×10^{-5}	2.00	2.32×10^{-5}	2.00	24.76
320×320	1.16×10^{-5}	2.00	5.81×10^{-6}	2.00	352.57
640×640	2.90×10^{-6}	2.00	1.45×10^{-6}	2.00	5125.76

Numerical errors, accuracy orders and CPU times are presented. Final time $T = 1.0$. $\Delta t = 0.5\,h$. CPU: CPU time for a complete simulation. CPU time unit: seconds

Table 6 Example 1, 2D case, Krylov IIF2 scheme, Non-uniform grids

		$N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Single-grid								
		80×80	1.17×10^{-6}		3.44×10^{-7}		16.86	
		160×160	8.21×10^{-8}	3.83	2.58×10^{-8}	3.73	190.58	11.30
		320×320	6.75×10^{-9}	3.60	2.96×10^{-9}	3.12	1045.65	5.49
		640×640	1.10×10^{-9}	2.61	5.66×10^{-10}	2.39	8495.37	8.12
N_r	N_L	$N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Sparse-grid, refine root grids								
10	3	80×80	1.17×10^{-6}		3.44×10^{-7}		9.72	
20	3	160×160	8.21×10^{-8}	3.83	2.58×10^{-8}	3.73	65.78	6.77
40	3	320×320	6.75×10^{-9}	3.60	2.96×10^{-9}	3.12	491.27	7.47
80	3	640×640	1.12×10^{-9}	2.60	5.72×10^{-10}	2.37	3852.75	7.84
Sparse-grid, refine level								
10	3	80×80	1.17×10^{-6}		3.44×10^{-7}		9.72	
10	4	160×160	8.21×10^{-8}	3.83	2.58×10^{-8}	3.73	45.06	4.64
10	5	320×320	6.75×10^{-9}	3.60	2.96×10^{-9}	3.12	206.02	4.57
10	6	640×640	1.10×10^{-9}	2.61	5.66×10^{-10}	2.39	936.73	4.55

Comparison of numerical errors and CPU times for computations on single-grid and sparse-grid. Final time $T = 1.0$. $\Delta t = 0.5\,h$. For single-grid computations, h is the smallest grid size in all spatial directions. For sparse-grid computations, h is the smallest grid size of the most refined grid Ω^{N_L,N_L}. N_r: number of cells in each spatial direction of a root grid. N_L: the finest level in a sparse-grid computation. CPU: CPU time for a complete simulation. "Ratio" is the ratio of corresponding CPU times on an $N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2}$ mesh. CPU time unit: seconds

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

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

In [19], we show that for high dimensional problems with anisotropic diffusion terms, Krylov IIF schemes are more efficient than compact IIF methods [23]. It is interesting to test Krylov

Table 7 Example 2, Krylov IIF2 scheme, comparison of numerical errors and CPU times for computations on single-grid and sparse-grid

		$N_h \times N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Single-grid								
		$80 \times 80 \times 80$	6.97×10^{-4}		4.93×10^{-4}		538.81	
		$160 \times 160 \times 160$	1.74×10^{-4}	2.00	1.23×10^{-4}	2.00	8413.74	15.62
		$320 \times 320 \times 320$	4.36×10^{-5}	2.00	3.08×10^{-5}	2.00	132,359.95	15.73
N_r	N_L	$N_h \times N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Sparse-grid, refine root grids								
10	3	$80 \times 80 \times 80$	7.49×10^{-4}		5.13×10^{-4}		118.58	
20	3	$160 \times 160 \times 160$	1.76×10^{-4}	2.09	1.24×10^{-4}	2.05	1817.95	15.33
40	3	$320 \times 320 \times 320$	4.36×10^{-5}	2.10	3.09×10^{-5}	2.01	29,573.60	16.27
80	3	$640 \times 640 \times 640$	1.09×10^{-5}	2.00	7.71×10^{-6}	2.00	465,538.00	15.74
Sparse-grid, refine level								
10	3	$80 \times 80 \times 80$	7.49×10^{-4}		5.13×10^{-4}		118.58	
10	4	$160 \times 160 \times 160$	1.87×10^{-4}	2.00	1.30×10^{-4}	1.98	728.21	6.14
10	5	$320 \times 320 \times 320$	4.73×10^{-5}	1.98	3.28×10^{-5}	1.98	4371.18	6.00
10	6	$640 \times 640 \times 640$	1.20×10^{-5}	1.98	8.30×10^{-6}	1.98	25,736.20	5.89

Final time $T = 1.0$. $\triangle t = h/3$. N_r: number of cells in each spatial direction of a root grid. N_L: the finest level in a sparse-grid computation. CPU: CPU time for a complete simulation. "Ratio" is the ratio of corresponding CPU times on an $N_h \times N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2} \times \frac{N_h}{2}$ mesh. CPU time unit: seconds

IIF scheme on sparse-grid for such problems with anisotropic diffusion terms. We compute the problem till final time $T = 1$ by the Krylov IIF2 scheme (13) on both single-grid and sparse-grid. The L^∞ errors, L^2 errors, the corresponding numerical accuracy orders, and CPU times on successively refined meshes are reported in Table 7.

As that in the last example, the computation on the $640 \times 640 \times 640$ single-grid can *not* be performed due to the computer memory restriction. Computer memory is saved significantly by using sparse-grid and computations can be successfully done for the $640 \times 640 \times 640$ mesh case. We observe that computations on both single-grid and sparse-grid give similar numerical errors and the second order accuracy. Again, it is shown in Table 7 that by preforming computations on sparse grids, a significant amount of CPU time can be saved, especially if we use a relatively large finest level N_L and a small number of cells N_r in each spatial direction of the root grid. For example, we compare the CPU times for computations on a $320 \times 320 \times 320$ mesh. With the number of cells in each spatial direction of a root grid $N_r = 40$ and the finest level $N_L = 3$, the CPU time for the computation on sparse-grid (29,573.60 s) is about 22 % of that on a single-grid (132,359.95 s), and 78 % CPU time is saved. Furthermore, with a coarser root grid $N_r = 10$ and the finest level $N_L = 5$, the CPU time for the computation on sparse-grid (4371.18 s) is only 3.3 % of that on a single-grid (132,359.95 s), and 96.7 % CPU time is saved. We can also observe that if the mesh refinement is done by refining root grids, the Krylov IIF2 scheme on sparse grids has the linear computational complexity as that for the Krylov IIF2 scheme on single-grid, with CPU time ratios around 16. If the mesh refinement is done by refining level, the CPU time ratios are around 6, and the computations on sparse grids have much better than linear computational complexity.

Example 3 (**A 3D problem with anisotropic diffusion and variable diffusion coefficients**). In this example, we consider a three-dimensional reaction–diffusion problem with cross-derivative diffusion terms and variable diffusion coefficients

$$\begin{aligned} u_t &= 0.5u_{xx} - 0.5\sin(x+y)u_{xy} + 0.5u_{yy} \\ &\quad + 0.5u_{xx} - \frac{1}{3}\cos y u_{xz} + \frac{1}{3}u_{zz} \\ &\quad + 0.5(1+\cos x)u_{yy} - 0.5(1+\cos x)u_{yz} + \frac{1}{3}(1+\cos x)u_{zz} + f(x,y,z,u), \end{aligned} \tag{17}$$

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 $f(x, y, z, u) = \big(1.3 + \frac{2}{3} - 0.5\sin(x + y) + \frac{1}{3}(\cos x - \cos y)\big)u$. The exact solution of this problem is

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

As the last example, in [19] we show that for this problem with anisotropic diffusion terms, Krylov IIF schemes are more efficient than compact IIF methods [23]. Here we use this example to show the significant improvement of computational efficiency of Krylov IIF scheme on sparse grids. We compute the problem till final time $T = 1$ by the Krylov IIF2 scheme (13) on both single-grid and sparse-grid. Again we use two different ways to refine meshes for computations on sparse grids. The numerical results are reported in Table 8. We obtain the similar observations and draw the same conclusion as the last example which has constant diffusion coefficients. Computer memory is saved significantly by using sparse-grid and computations can be successfully done for the $640 \times 640 \times 640$ mesh case, for which the computation on single-grid can *not* be performed due to computer memory restriction. Again, applying sparse-grid combination technique in the Krylov IIF scheme brings in a huge benefit in terms of CPU time savings while the similar numerical errors and accuracy orders are kept as that for the single-grid computations. For example, we compare the CPU times for computations on a $320 \times 320 \times 320$ mesh. With the number of cells in each spatial direction of a root grid $N_r = 40$ and the finest level $N_L = 3$, the CPU time for the computation on sparse-grid (55,060.30 s) is about 36 % of that on a single-grid (153, 195.14 s), and 64 % CPU time is saved. Furthermore, with a coarser root grid $N_r = 10$ and the finest level $N_L = 5$, the CPU time for the computation on sparse-grid (8139.66 s) is only 5.3 % of that on a single-grid (153, 195.14 s), and 94.7 % CPU time is saved. We can also observe that if the mesh refinement is done by refining root grids, the Krylov IIF2 scheme on sparse grids has the linear computational complexity as that for the Krylov IIF2 scheme on single-grid, with CPU time ratios around 16. If the mesh refinement is done by refining level, the CPU time ratios are around 6, and the computations on sparse grids have much better than linear computational complexity.

Example 4 (**A convection–diffusion problem**). In this example, we test the method for solving problems with convection terms. Consider a two-dimensional convection-diffusion problem

$$\frac{\partial u}{\partial t} + \left(\frac{1}{2}u^2\right)_x + \left(\frac{1}{2}u^2\right)_y = 0.2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + f(x, y, t),$$

Table 8 Example 3, Krylov IIF2 scheme, comparison of numerical errors and CPU times for computations on single-grid and sparse-grid

		$N_h \times N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Single-grid								
		$80 \times 80 \times 80$	3.34×10^{-3}		1.09×10^{-3}		551.57	
		$160 \times 160 \times 160$	8.34×10^{-4}	2.00	2.71×10^{-4}	2.01	8992.13	16.30
		$320 \times 320 \times 320$	2.09×10^{-4}	2.00	6.79×10^{-5}	2.00	153,195.14	17.04
N_r	N_L	$N_h \times N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Sparse-grid, refine root grids								
10	3	$80 \times 80 \times 80$	3.19×10^{-3}		1.10×10^{-3}		229.37	
20	3	$160 \times 160 \times 160$	8.13×10^{-4}	1.97	2.70×10^{-4}	2.03	3618.00	15.77
40	3	$320 \times 320 \times 320$	2.07×10^{-4}	1.97	6.77×10^{-5}	1.99	55,060.30	15.22
80	3	$640 \times 640 \times 640$	5.21×10^{-5}	1.99	1.70×10^{-5}	2.00	865,203.00	15.71
Sparse-grid, refine level								
10	3	$80 \times 80 \times 80$	3.19×10^{-3}		1.10×10^{-3}		229.37	
10	4	$160 \times 160 \times 160$	7.85×10^{-4}	2.02	2.82×10^{-4}	1.97	1414.63	6.17
10	5	$320 \times 320 \times 320$	1.94×10^{-4}	2.02	7.30×10^{-5}	1.95	8139.66	5.75
10	6	$640 \times 640 \times 640$	4.83×10^{-5}	2.01	1.90×10^{-5}	1.94	46,392.90	5.70

Final time $T = 1.0$. $\Delta t = h/3$. N_r: number of cells in each spatial direction of a root grid. N_L: the finest level in a sparse-grid computation. CPU: CPU time for a complete simulation. "Ratio" is the ratio of corresponding CPU times on an $N_h \times N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2} \times \frac{N_h}{2}$ mesh. CPU time unit: seconds

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

$$u(x, y, t) = e^{-0.1t}(\cos(x) + \sin(y)).$$

The source term $f(x, y, t)$ is

$$f(x, y, t) = \Big(0.1 + e^{-0.1t}(-\sin(x) + \cos(y))\Big)e^{-0.1t}(\cos(x) + \sin(y)).$$

The Krylov IIF2 scheme (13) with the third order WENO approximation for the convection terms is used here. We compute the problem till final time $T = 1$ on both single-grid and sparse-grid. Here the time step sizes are determined only by the convection (hyperbolic) part of the equation since the IIF schemes remove stability constraint of diffusion and reaction terms [12]. The CFL number for the convection terms is taken to be 0.5 in the computations. Numerical errors, numerical accuracy orders, CPU times for a complete simulation, and the ratios of CPU times on an $N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2}$ mesh are reported. Again, two approaches to perform mesh refinement in sparse-grid computations are used, i.e., the refining root grid approach and the refining level approach. In this example, for mesh refinement in sparse-grid computations by the refining root grid approach, we test performance of the method with two different finest levels $N_L = 3$ and $N_L = 4$. Numerical results are reported in Table 9. We observe that the desired second order accuracy due to the second order Krylov IIF scheme is achieved for all methods. About computational efficiency, we observe that in general a big amount of CPU time is saved if computations are performed on sparse grids.

Table 9 Example 4, Krylov IIF2 scheme, comparison of numerical errors and CPU times for computations on single-grid and sparse-grid

		$N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Single-grid								
		80×80	1.67×10^{-4}		6.91×10^{-5}		13.77	
		160×160	2.23×10^{-5}	2.91	1.27×10^{-5}	2.44	104.16	7.56
		320×320	9.54×10^{-6}	1.22	4.61×10^{-6}	1.47	851.93	8.18
		640×640	2.80×10^{-6}	1.77	1.30×10^{-6}	1.82	6958.20	8.17
N_r	N_L	$N_h \times N_h$	L^∞ error	Order	L^2 error	Order	CPU(s)	Ratio
Sparse-grid, refine root grids, $N_L = 3$								
10	3	80×80	3.52×10^{-3}		8.24×10^{-4}		13.24	
20	3	160×160	3.32×10^{-5}	6.72	1.36×10^{-5}	5.92	81.58	6.16
40	3	320×320	9.44×10^{-6}	1.82	4.60×10^{-6}	1.56	601.76	7.38
80	3	640×640	2.80×10^{-6}	1.76	1.30×10^{-6}	1.82	4712.98	7.83
Sparse-grid, refine root grids, $N_L = 4$								
10	4	160×160	4.97×10^{-4}		1.07×10^{-4}		58.22	
20	4	320×320	8.70×10^{-6}	5.84	4.48×10^{-6}	4.58	395.54	6.79
40	4	640×640	2.78×10^{-6}	1.65	1.30×10^{-6}	1.78	3001.72	7.59
80	4	1280×1280	7.46×10^{-7}	1.90	3.43×10^{-7}	1.93	24,183.80	8.06
Sparse-grid, refine level								
10	3	80×80	3.52×10^{-3}		8.24×10^{-4}		13.24	
10	4	160×160	4.97×10^{-4}	2.82	1.07×10^{-4}	2.94	58.22	4.40
10	5	320×320	3.88×10^{-5}	3.68	8.63×10^{-6}	3.63	260.20	4.47
10	6	640×640	5.69×10^{-6}	2.77	1.59×10^{-6}	2.44	1170.34	4.50

CFL number for the hyperbolic terms is 0.5. Final time $T = 1.0$. N_r: number of cells in each spatial direction of a root grid. N_L: the finest level in a sparse-grid computation. CPU: CPU time for a complete simulation. "Ratio" is the ratio of corresponding CPU times on an $N_h \times N_h$ mesh to that on a $\frac{N_h}{2} \times \frac{N_h}{2}$ mesh. CPU time unit: seconds

Specifically, for example for the 640×640 mesh case, computations on sparse grids can save 57 % CPU time (the $N_r = 40$, $N_L = 4$ case), and even 83 % CPU time (the $N_r = 10$, $N_L = 6$ case) comparing with the single-grid computation, and keep comparable numerical errors. See Table 9.

Again we can also observe that if the mesh refinement is done by refining root grids, Krylov IIF2 scheme on sparse grids has the linear computation complexity as that for the Krylov IIF2 scheme on single-grid, with CPU time ratios around 8. If the mesh refinement is done by refining level, the CPU time ratios are around 5, and the computations on sparse grids have better than linear computational complexity.

Example 5 (**Three dimensional Fokker–Planck equations**). The Fokker–Planck equation (FPE) [6,24] 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 [28], Array-representation integration factor scheme was applied in solving FPEs which describe the time evolution of the probability density function of stochastic systems [25]. The general form of FPEs is

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t} = -\sum_{r=1}^{R}\left\{\sum_{i=1}^{N} n_{ri}\frac{\partial}{\partial x_i}\left(q_r(\boldsymbol{x},t) - \frac{1}{2}\sum_{j=1}^{N} n_{rj}\frac{\partial q_r(\boldsymbol{x},t)}{\partial x_j}\right)\right\}, \tag{18}$$

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 i-th reactant. n_{ri} is the change of x_i when the r-th 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 r-th reaction at state $\boldsymbol{x}$ and time t. Here we apply the second order Krylov IIF scheme Krylov IIF2 (13) on both single-grid and sparse-grid in solving a three dimensional Fokker–Planck equation [26] which involves two metabolites A and B and one enzyme E_A and show computational efficiency of the scheme on sparse-grid. Since Krylov IIF2 scheme for solving convection–diffusion equations is a multistep method, numerical values at the first time step are needed to start the computation. We use a third order Runge-Kutta scheme for the first step time evolution. Then the Krylov IIF2 scheme is used to continue the time evolution. The reactions are described as following (here Ø means that there is no reactant or product in the reaction):

$$\begin{aligned}
&\emptyset \xrightarrow{\frac{k_A[E_A]}{1+[A]/K_I}} A, \qquad \emptyset \xrightarrow{k_B} B,\\
&A + B \xrightarrow{k[A][B]} \emptyset,\\
&A \xrightarrow{\mu[A]} \emptyset, \qquad B \xrightarrow{\mu[B]} \emptyset,\\
&\emptyset \xrightarrow{\frac{k_{E_A}}{1+[A]/K_R}} E_A, \qquad E_A \xrightarrow{\mu[E_A]} \emptyset.
\end{aligned} \tag{19}$$

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

$$\begin{aligned}
&w_1 = \frac{k_A x_3}{1 + x_1/K_I}, \quad w_2 = k_B, \quad w_3 = k x_1 x_2,\\
&w_4 = \mu x_1, \quad w_5 = \mu x_2, \quad w_6 = \frac{k_{E_A}}{1 + x_1/K_R}, \quad w_7 = \mu x_3,
\end{aligned} \tag{20}$$

where $k_A = 0.3\text{s}^{-1}$, $k_B = 2\text{s}^{-1}$, $K_I = 30$, $k = 0.001s^{-1}$, $\mu = 0.004\text{s}^{-1}$, $K_R = 30$ and $k_{E_A} = 1\text{s}^{-1}$ [26]. Then the FPE can be written as

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t} = -(L_1 + L_2 + L_3 + L_4 + L_5 + L_6 + L_7), \tag{21}$$

where L_r represents the operator for the r-th reaction. Specifically,

$$
\begin{aligned}
L_1 &= \frac{\partial q_1(\boldsymbol{x},t)}{\partial x_1} - \frac{1}{2}\frac{\partial^2 q_1(\boldsymbol{x},t)}{\partial x_1^2},\\
L_2 &= \frac{\partial q_2(\boldsymbol{x},t)}{\partial x_2} - \frac{1}{2}\frac{\partial^2 q_2(\boldsymbol{x},t)}{\partial x_2^2},\\
L_3 &= -\frac{\partial q_3(\boldsymbol{x},t)}{\partial x_1} - \frac{\partial q_3(\boldsymbol{x},t)}{\partial x_2} - \frac{1}{2}\left(\frac{\partial^2 q_3(\boldsymbol{x},t)}{\partial x_1^2} + \frac{\partial^2 q_3(\boldsymbol{x},t)}{\partial {x_2}^2} + 2\frac{\partial^2 q_3(\boldsymbol{x},t)}{\partial x_1 \partial x_2}\right),\\
L_4 &= -\frac{\partial q_4(\boldsymbol{x},t)}{\partial x_1} - \frac{1}{2}\frac{\partial^2 q_4(\boldsymbol{x},t)}{\partial x_1^2},\\
L_5 &= -\frac{\partial q_5(\boldsymbol{x},t)}{\partial x_2} - \frac{1}{2}\frac{\partial^2 q_5(\boldsymbol{x},t)}{\partial x_2^2},\\
L_6 &= \frac{\partial q_6(\boldsymbol{x},t)}{\partial x_3} - \frac{1}{2}\frac{\partial^2 q_6(\boldsymbol{x},t)}{\partial x_3^2},\\
L_7 &= -\frac{\partial q_7(\boldsymbol{x},t)}{\partial x_3} - \frac{1}{2}\frac{\partial^2 q_7(\boldsymbol{x},t)}{\partial x_3^2}.
\end{aligned}
\tag{22}
$$

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 upwind scheme for the convection terms and the second order central difference scheme for the diffusion terms. For simulations here, the time step size $\triangle t$ is 0.015 (corresponding to the CFL number 0.4 for the convection part) and the numbers of cells in spatial directions are $N_A = 128$, $N_B = 128$, $N_{E_A} = 64$. For the sparse-grid computations, the root grid is $16 \times 16 \times 8$, and the finest level is $N_L = 3$. In Table 10, we list the errors and accuracy orders for both single-grid and sparse-grid computations, and the similar 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 $128 \times 128 \times 64$, 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 the scheme on single and sparse grids and list CPU times of using them to solve the problem till the final time $T = 10$

Table 10 Numerical errors and accuracy orders for the Krylov IIF2 scheme to solve the 3D Fokker–Planck equation on single and sparse grids

Time step	L^∞ error	Accuracy
On single-grid		
$\triangle t$	1.20×10^{-11}	
$\triangle t/2$	3.04×10^{-12}	1.99
$\triangle t/4$	7.61×10^{-13}	2.00
On sparse-grid		
$\triangle t$	1.32×10^{-11}	
$\triangle t/2$	3.40×10^{-12}	1.96
$\triangle t/4$	8.41×10^{-13}	2.01

Final time $T = 5$. $\triangle t = 0.015$

Table 11 CPU time for the Krylov IIF2 scheme to solve the 3D Fokker–Planck equation on single and sparse grids

	CPU
On single-grid	78,745
On sparse-grid	14,218

Final time $T = 10$. $\triangle t = 0.015$. CPU: CPU time for a complete simulation. CPU time unit: seconds

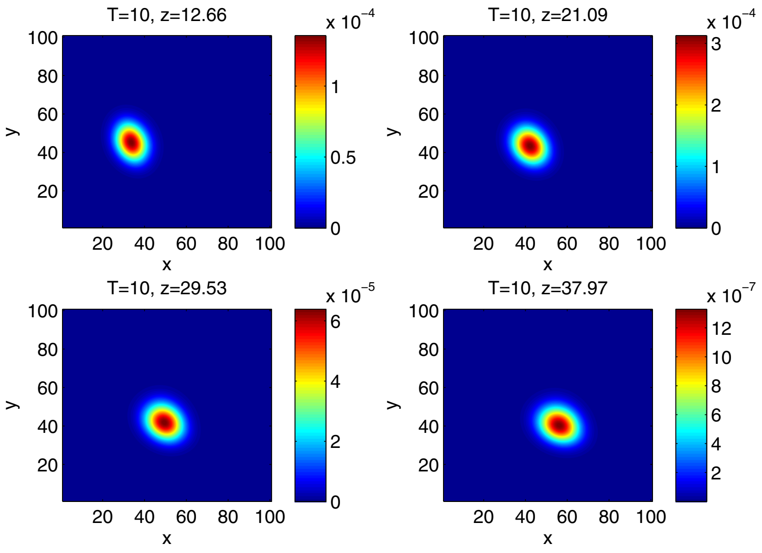

Fig. 2 Numerical solutions of the 3D Fokker–Planck equation using the Krylov IIF2 scheme on single-grid. Final time $T = 10$. $\triangle t = 0.015$. Distribution of molecular species A and B with $E_A = 12.66, 21.09, 29.53, 37.97$

with $\triangle t = 0.015$, in Table 11. The CPU times in Table 11 show that a significant amount of CPU time (82 % CPU time) is saved by using the sparse-grid combination technique. In Figs. 3, 5, and 7, we show contour plots of the numerical solutions by the Krylov IIF2 scheme with sparse-grid combination technique on two dimensional domain of molecular species A and B, with different values of the third dimension E_A. Contour plots of the numerical solutions by the same scheme on single-grid are presented in Figs. 2, 4, and 6. We see that both approaches generate similar numerical solutions.

5 Conclusions

In this paper, we design the Krylov IIF scheme on sparse grids for solving high spatial dimension problems. Our early work shows that the Krylov IIF scheme has linear computational complexity and is especially efficient for solving high dimensional convection–diffusion problems with anisotropic diffusions, for example high dimensional Fokker–Planck equations. With the Krylov IIF scheme on sparse-grid, more efficient algorithm than our previous work is achieved. Numerical experiments are performed for the sparse-grid Krylov IIF method

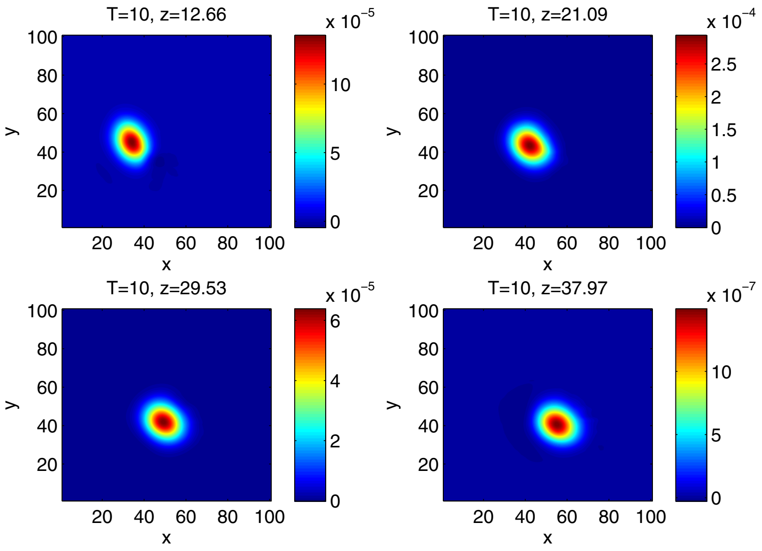

Fig. 3 Numerical solutions of the 3D Fokker–Planck equation using the Krylov IIF2 scheme on sparse-grid. Final time $T = 10$. $\triangle t = 0.015$. Distribution of molecular species A and B with $E_A = 12.66, 21.09, 29.53, 37.97$

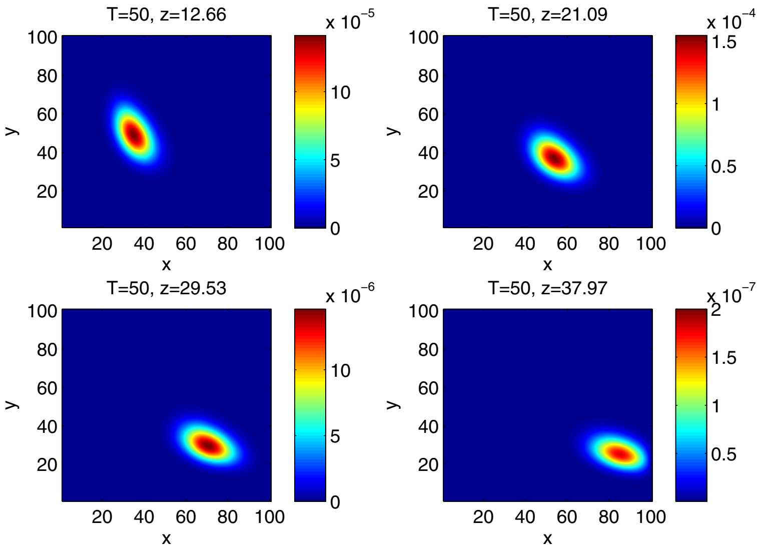

Fig. 4 Numerical solutions of the 3D Fokker–Planck equation using the Krylov IIF2 scheme on single-grid. Final time $T = 50$. $\triangle t = 0.015$. Distribution of molecular species A and B with $E_A = 12.66, 21.09, 29.53, 37.97$

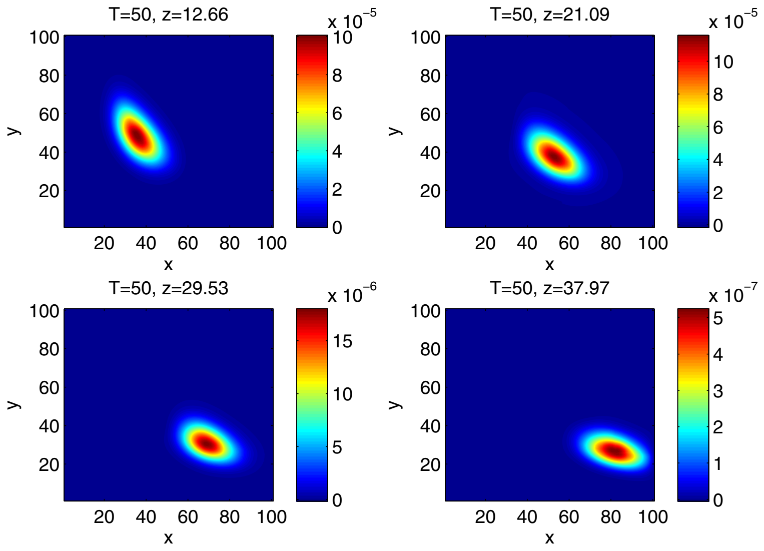

Fig. 5 Numerical solutions of the 3D Fokker–Planck equation using the Krylov IIF2 scheme on sparse-grid. Final time $T = 50$. $\triangle t = 0.015$. Distribution of molecular species A and B with $E_A = 12.66, 21.09, 29.53, 37.97$

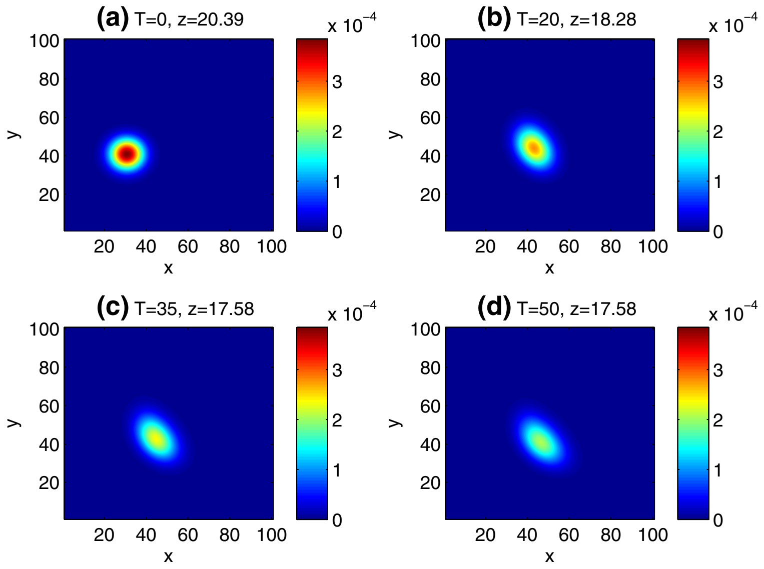

Fig. 6 Numerical solutions of the 3D Fokker–Planck equation using the Krylov IIF2 scheme on single-grid. Distribution of molecular species A and B with different E_A values, at time $T = 0, 20, 35, 50$. $\triangle t = 0.015$

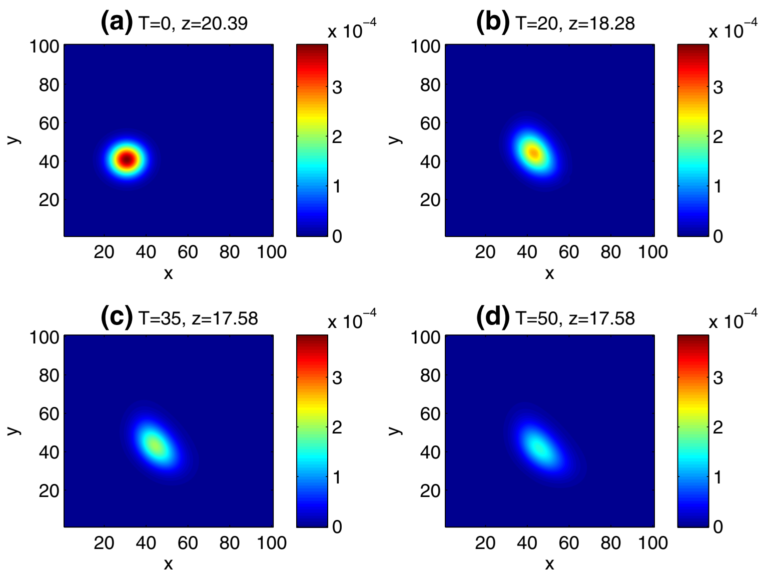

Fig. 7 Numerical solutions of the 3D Fokker–Planck equation using the Krylov IIF2 scheme on sparse-grid. Distribution of molecular species A and B with different E_A values, at time $T = 0, 20, 35, 50$. $\triangle t = 0.015$

to show significant savings in computational costs by comparisons with single-grid computations. It will be interesting to theoretically analyze the errors for the sparse-grid Krylov IIF method in solving both linear and nonlinear problems, and design the sparse-grid combination technique on unstructured triangular meshes. These will be our future work.

6 Appendix: Linear Stability Analysis of the IIF2 Scheme (7) for CDR Equations

To analyze the linear stability of IIF schemes, we use the following scalar linear test equation

$$u_t = au - du + ru, \qquad \text{with } r \in \mathcal{C}, \text{and } a, d \in \mathcal{R}, d > 0. \tag{23}$$

In the context of solving CDR equations, a and d actually represent spatial discretizations for the convection term and the diffusion term respectively. Following the stability analysis approach in [22], we show boundaries of the stability regions in the complex plane for $r\Delta t$, a family of curves for different values of $d\Delta t$ and $a\Delta t$, and indicate the corresponding stability regions. Here we present the analysis of the IIF2 scheme (7) as an example. More details and analysis results can be found in [12].

Applying the IIF2 scheme (7) to the Eq. (23) with a uniform time step size Δt, then substituting $u_n = e^{in\theta}$ into the resulting equation, we obtain

$$\left(1 - \frac{\lambda}{2}\right) e^{2i\theta} = e^{-d\Delta t} \left(1 + \frac{\lambda}{2} + \frac{3}{2}a\Delta t\right) e^{i\theta} - \frac{a}{2}\Delta t e^{-2d\Delta t}, \tag{24}$$

where $\lambda = r\Delta t$ has a real part λ_r and imaginary part λ_i. Solve the Eq. (24) for λ_r and λ_i to have

$$\begin{cases} \lambda_r = \dfrac{B_1C_2 - B_2C_1}{A_1B_2 - A_2B_1}; \\ \lambda_i = \dfrac{A_1C_2 - A_2C_1}{A_2B_1 - A_1B_2}, \end{cases} \tag{25}$$

where

$$\begin{cases} A_1 = e^{-d\Delta t}\dfrac{1}{2}\cos\theta + \dfrac{1}{2}\cos 2\theta, \\ B_1 = -e^{-d\Delta t}\dfrac{1}{2}\sin\theta - \dfrac{1}{2}\sin 2\theta, \\ C_1 = -\dfrac{a}{2}\Delta t e^{-2d\Delta t} + e^{-d\Delta t}\left(1 + \dfrac{3}{2}a\Delta t\right)\cos\theta - \cos 2\theta, \\ A_2 = e^{-d\Delta t}\dfrac{1}{2}\sin\theta + \dfrac{1}{2}\sin 2\theta, \\ B_2 = e^{-d\Delta t}\dfrac{1}{2}\cos\theta + \dfrac{1}{2}\cos 2\theta, \\ C_2 = e^{-d\Delta t}\left(1 + \dfrac{3}{2}a\Delta t\right)\sin\theta - \sin 2\theta. \end{cases} \tag{26}$$

Stability regions in the complex plane of $r\Delta t$ for different values of $d\Delta t$ under a fixed value of $a\Delta t$ are presented in Fig. 8. As examples we choose four different $a\Delta t$ values: $a\Delta t = 1.0$, $a\Delta t = 10.0$, $a\Delta t = -1.0$ and $a\Delta t = -10.0$. The points on boundaries of stability regions are obtained by varying θ from 0 to 2π in (25) and (26). A stability boundary curve divides the whole complex plane into the stable region and the unstable region for a pair of fixed values of $d\Delta t$ and $a\Delta t$. Based on analyzing the growth factor of the scheme (7) for some special values of $d\Delta t$, $a\Delta t$ and λ, we find that the stable regions always include the point $\lambda = (-20, 0)$ for any values of $d\Delta t$ and $a\Delta t$ used in Fig. 8. Then stable and unstable regions are determined and shown in Fig. 8. From Fig. 8, we can see that the whole regions outside

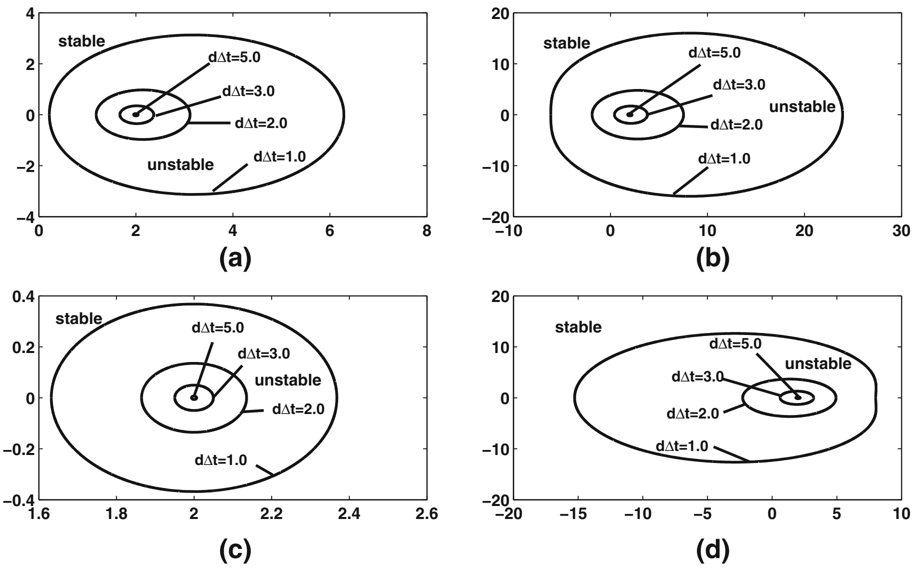

Fig. 8 Linear stability regions of the IIF2 scheme (7) for different values of $d\Delta t$ under a fixed value of $a\Delta t$. **a** $a\Delta t = 1.0$; **b** $a\Delta t = 10.0$; **c** $a\Delta t = -1.0$; **d** $a\Delta t = -10.0$

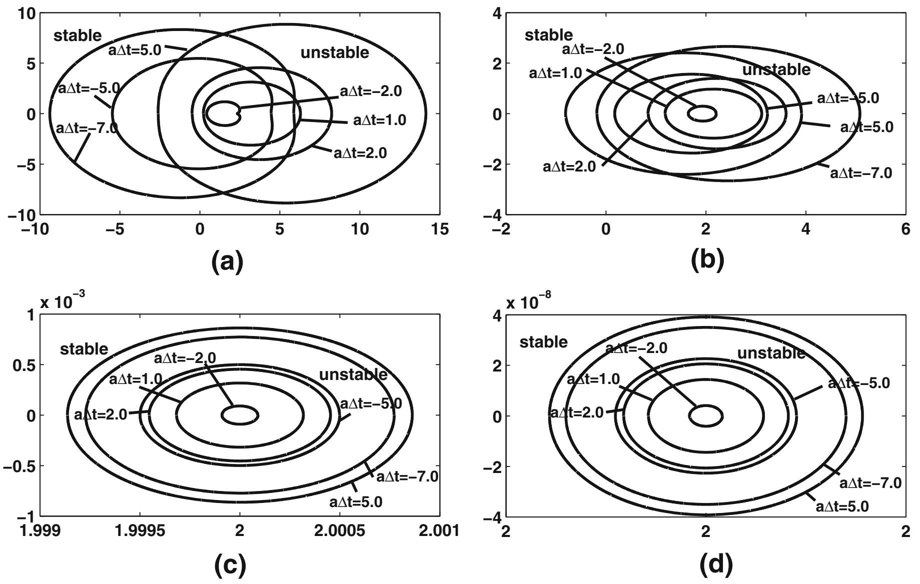

Fig. 9 Linear stability regions of the IIF2 scheme (7) for different values of $a\Delta t$ under a fixed value of $d\Delta t$. **a** $d\Delta t = 1.0$; **b** $d\Delta t = 2.0$; **c** $d\Delta t = 10.0$; **d** $d\Delta t = 20.0$

of the stability boundary curves are stable regions, which shows that the IIF2 scheme (7) has large stability regions. For a fixed $a\Delta t$, the stable region becomes larger with the increase of the value of $d\Delta t$. Next we show stability regions for different values of $a\Delta t$ under a fixed value of $d\Delta t$ in Fig. 9. $d\Delta t = 1.0$, $d\Delta t = 2.0$, $d\Delta t = 10.0$ and $d\Delta t = 20.0$ are chosen as examples. Again, analysis of the growth factor of the scheme (7) for some special values of $d\Delta t$, $a\Delta t$ and λ, we find that the stable regions always include the point $\lambda = (-10, 0)$ for any values of $a\Delta t$ and $d\Delta t$ used in Fig. 9. Stable regions for the cases shown in Fig. 9 are the whole regions outside of the stability boundary curves. For a fixed $d\Delta t$, the stable region becomes smaller with the increase of the value of $|a|\Delta t$ which corresponds to the convection terms. Based on the linear stability analysis, we conclude that the diffusion term tends to stabilize the scheme, while the convection term gives constraints on time step sizes. Due to the implicit property of the scheme, the stability regions are quite large and often include the whole left complex plane, with a relatively large size diffusion parameter d and a mild size convection parameter a.

References

1. 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)
2. Briggs, W.L., Henson, V.E., and McCormick, S.F.: A multigrid tutorial. SIAM, (2000)
3. Bungartz, H.-J., Griebel, M.: Sparse grids. Acta Numer. **13**, 147–269 (2004)
4. 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)
5. Cox, S.M., Matthews, P.C.: Exponential time differencing for stiff systems. J. Comput. Phys. **176**, 430–455 (2002)
6. Fokker, A.D.: Die mittlere energie rotierender elektrischer dipole im strahlungsfeld. Ann. Phys. **348**, 810–820 (1914)

7. Gallopoulos, E., Saad, Y.: Efficient solution of parabolic equations by Krylov approximation methods. SIAM J. Sci. Stat. Comput. **13**(5), 1236–1264 (1992)
8. Griebel, M., Schneider, M., Zenger, C.: A combination technique for the solution of sparse grid problems. In: Beauwens, R., de Groen, P. (eds.) Iterative Methods in Linear Algebra, pp. 263–281. North-Holland, Amsterdam (1992)
9. Gustafsson, B., Kreiss, H.-O., Oliger, J.: Time Dependent Problems and Difference Methods. Wiley, New York (1995)
10. Higham, N.J.: The scaling and squaring method for the matrix exponential revisited. SIAM Rev. **51**(4), 747–764 (2009)
11. Jiang, G.-S., Shu, C.-W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. **126**, 202–228 (1996)
12. Jiang, T., Zhang, Y.-T.: Krylov implicit integration factor WENO methods for semilinear and fully non-linear advection–diffusion-reaction equations. J. Comput. Phys. **253**, 368–388 (2013)
13. 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)
14. Ju, L., Liu, X., Leng, W.: Compact implicit integration factor methods for a family of semilinear fourth-order parabolic equations. Discrete Contin. Dyn. Syst. Ser. B **19**, 1667–1687 (2014)
15. Ju, L., Zhang, J., Zhu, L., Du, Q.: Fast explicit integration factor methods for semilinear parabolic equations. J. Sci. Comput. **62**, 431–455 (2015)
16. Kleefeld, B., Khaliq, A.Q.M., Wade, B.A.: An ETD Crank-Nicolson method for reaction-diffusion systems. Numer. Methods Partial Differ. Equ. **28**, 1309–1335 (2012)
17. Lastdrager, B., Koren, B., Verwer, J.: The sparse-grid combination technique applied to time-dependent advection problems. Appl. Numer. Math. **38**, 377–401 (2001)
18. Lastdrager, B., Koren, B., Verwer, J.: Solution of time-dependent advection-diffusion problems with the sparse-grid combination technique and a rosenbrock solver. Comput. Methods Appl. Math. **1**, 86–99 (2001)
19. Lu, D., Zhang, Y.-T.: Computational complexity study on Krylov integration factor WENO method for high spatial dimension convection–diffusion problems. J. Comput. Appl. Math. submitted, (2015)
20. 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)
21. 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)
22. Nie, Q., Zhang, Y.-T., Zhao, R.: Efficient semi-implicit schemes for stiff systems. J. Comput. Phys. **214**, 521–537 (2006)
23. 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)
24. Planck, M.: Sitzber. Preuss. Akad. Wiss. (1917) p. 324
25. Risken, H.: The Fokker–Planck Equation: Methods of Solution and Applications. Springer, Berlin (1996)
26. Sjoberg, P., Lotstedt, P., Elf, J.: Fokker-planck approximation of the master equation in molecular biology. Comput. Visual Sci. **12**, 37–50 (2009)
27. Trefethen, L.N., Bau, D.: Numerical Linear Algebra, SIAM, (1997)
28. Wang, D., Zhang, L., Nie, Q.: Array-representation integration factor method for high-dimensional systems. J. Comput. Phys. **v258**, 585–600 (2014)
29. Zenger, C.: Sparse grids. In: Hackbusch, W. (ed.) Notes on Numerical Fluid Mechanics, vol. 31, pp. 241–251. Vieweg, Braunschweig (1991)