

---

**Krylov Integration Factor Method for High Spatial Dimension Convection-Diffusion Problems on Sparse Grids****Dong Lu****Publication Date**

13-06-2017

**License**

This work is made available under a All Rights Reserved license and should only be used in accordance with that license.

**Citation for this work (American Psychological Association 7th edition)**

Lu, D. (2017). *Krylov Integration Factor Method for High Spatial Dimension Convection-Diffusion Problems on Sparse Grids* (Version 1). University of Notre Dame. <https://doi.org/10.7274/mw22v408901>

This work was downloaded from CurateND, the University of Notre Dame's institutional repository.

For more information about this work, to report or an issue, or to preserve and share your original work, please contact the CurateND team for assistance at [curate@nd.edu](mailto:curate@nd.edu).

KRYLOV INTEGRATION FACTOR METHOD FOR HIGH SPATIAL  
DIMENSION CONVECTION-DIFFUSION PROBLEMS ON SPARSE GRIDS

A Dissertation

Submitted to the Graduate School  
of the University of Notre Dame  
in Partial Fulfillment of the Requirements  
for the Degree of

Doctor of Philosophy

by  
Dong Lu

---

Yong-Tao Zhang, Director

Graduate Program in Applied and Computational Mathematics and Statistics

Notre Dame, Indiana

June 2017

© Copyright by

Dong Lu

2017

All Rights Reserved

# KRYLOV INTEGRATION FACTOR METHOD FOR HIGH SPATIAL DIMENSION CONVECTION-DIFFUSION PROBLEMS ON SPARSE GRIDS

Abstract

by

Dong Lu

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 (multi-dimensional PDEs) is how to deal with the matrix exponential for very large matrices. Compact integration factor methods developed in [Nie et al., Journal of Computational Physics, 227 (2008) 5238-5255] 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., Journal of Computational Physics, 258 (2014) 585-600] 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, Journal of Computational Physics, 230 (2011) 4336-4352], 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, Journal of Computational Physics, 253 (2013) 368-

388] 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 the first part of this dissertation, we study the computational power of Krylov IF-WENO methods for solving high spatial dimension convection-diffusion PDE problems (up to four 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.

In the second part of this dissertation, 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.

Dedicated to my family.

# CONTENTS

FIGURES . . . . .	v
TABLES . . . . .	viii
ACKNOWLEDGMENTS . . . . .	xi
CHAPTER 1: INTRODUCTION . . . . .	1
CHAPTER 2: NUMERICAL METHODS . . . . .	6
2.1 IIF-WENO methods . . . . .	6
2.2 Two approaches for high dimensional problems . . . . .	11
2.2.1 Krylov approximation method . . . . .	11
2.2.2 Compact / array-representation method . . . . .	15
2.2.2.1 cIIF/AcIIF for reaction-diffusion equations . . . . .	16
2.2.2.2 AcIIF-WENO schemes for CDR equations . . . . .	20
2.2.2.3 Detailed formulae for AcIIF-WENO schemes. . . . .	27
2.3 Krylov IF method on sparse grids . . . . .	29
2.4 Linear stability analysis of the IIF2 scheme for CDR equations. . . . .	33
CHAPTER 3: NUMERICAL EXAMPLES FOR COMPUTATIONAL COM- PLEXITY STUDY OF KRYLOV INTEGRATION FACTOR WENO METHOD . . . . .	38
3.1 Diffusion problems . . . . .	38
3.1.1 Diffusion problems without cross-derivatives . . . . .	39
3.1.2 Diffusion problems with cross-derivatives . . . . .	52
3.1.3 A system with stiff reactions from mathematical biology . . . . .	67
3.2 Convection-Diffusion problems . . . . .	69
CHAPTER 4: NUMERICAL EXAMPLES FOR KRYLOV INTEGRATION METHODS ON SPARSE GRIDS . . . . .	96
4.1 Isotropic diffusion problems . . . . .	97
4.2 A 3D problem with anisotropic diffusion and constant diffusion coeffi- cients . . . . .	109
4.3 A 3D problem with anisotropic diffusion and variable diffusion coeffi- cients . . . . .	113

4.4	A convection-diffusion problem . . . . .	117
4.5	Three dimensional Fokker-Planck equations . . . . .	119
CHAPTER 5: CONCLUSION . . . . .		128
BIBLIOGRAPHY . . . . .		131



## FIGURES

2.1	Semi-coarsened sparse grids $\{\Omega^{l_1, l_2}\}$ with the finest level $N_L = 3$ . . . .	30
2.2	Linear stability regions of the IIF2 scheme (2.10) 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) $a\Delta t = -10.0$ . . . . .	36
2.3	Linear stability regions of the IIF2 scheme (2.10) 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) $d\Delta t = 20.0$ . . . . .	37
3.1	Numerical solutions of Example 6 using the Krylov IIF2 scheme: concentrations of $[L]$ , $[LR]$ , $[LS]$ , $[S]$ at $T = 100$ seconds for the Dpp-Sog system when receptors are over-expressed. $\Delta t = h_x = h_y = 0.001375$ in the simulation. Parameters are $D_L = D_{LS} = D_S = 85\mu m^2 s^{-1}$ ; $v_L = 1nM s^{-1}$ ; $v_S = 80nM s^{-1}$ ; $k_{on} = 0.4\mu M^{-1} s^{-1}$ ; $k_{off} = 4 \times 10^{-6} s^{-1}$ ; $k_{deg} = 5 \times 10^{-4} s^{-1}$ ; $j_{on} = 95\mu M^{-1} s^{-1}$ ; $j_{off} = 4 \times 10^{-6} s^{-1}$ ; $j_{deg} = 0.54 s^{-1}$ ; $\tau = 1$ ; $R_h = 9\mu M$ ; $R_0 = 3\mu M$ . . . . .	70
3.2	Numerical solutions of Example 6 using the cIIF2 scheme: concentrations of $[L]$ , $[LR]$ , $[LS]$ , $[S]$ at $T = 100$ seconds for the Dpp-Sog system when receptors are over-expressed. $\Delta t = h_x = h_y = 0.001375$ in the simulation. Parameters are $D_L = D_{LS} = D_S = 85\mu m^2 s^{-1}$ ; $v_L = 1nM s^{-1}$ ; $v_S = 80nM s^{-1}$ ; $k_{on} = 0.4\mu M^{-1} s^{-1}$ ; $k_{off} = 4 \times 10^{-6} s^{-1}$ ; $k_{deg} = 5 \times 10^{-4} s^{-1}$ ; $j_{on} = 95\mu M^{-1} s^{-1}$ ; $j_{off} = 4 \times 10^{-6} s^{-1}$ ; $j_{deg} = 0.54 s^{-1}$ ; $\tau = 1$ ; $R_h = 9\mu M$ ; $R_0 = 3\mu M$ . . . . .	71
3.3	Numerical solutions of nonlinear viscous Burgers' equation on a $80 \times 80$ 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. . . . .	79
3.4	Distribution of A and B with $E_A = 12.75, 21.75, 30.75, 39.75$ . Numerical solutions of (4.5) using the KrylovIIF2-WENO scheme. Final time $T = 10$ . $\Delta t = 0.017$ . The numbers of spatial grid points are $N_A = 120$ , $N_B = 120$ , $N_{E_A} = 60$ . . . . .	86
3.5	Distribution of A and B with $E_A = 12.75, 21.75, 30.75, 39.75$ . Numerical solutions of (4.5) using the KrylovIIF2-WENO scheme. Final time $T = 50$ . $\Delta t = 0.017$ . The numbers of spatial grid points are $N_A = 120$ , $N_B = 120$ , $N_{E_A} = 60$ . . . . .	86

3.6	Distribution of A and B with different $E_A$ values, at time $T = 0, 20, 35, 50$ . Numerical solutions of (4.5) using the KrylovIIF2-WENO scheme. $\Delta t = 0.017$ . The numbers of spatial grid points are $N_A = 120$ , $N_B = 120$ , $N_{E_A} = 60$ . . . . .	87
3.7	Distribution of A and B with $E_A = 12.75, 21.75, 30.75, 39.75$ . Numerical solutions of (4.5) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time $T = 10$ . $\Delta t = 0.017$ . The numbers of spatial grid points are $N_A = 120$ , $N_B = 120$ , $N_{E_A} = 60$ . .	87
3.8	Distribution of A and B with $E_A = 12.75, 21.75, 30.75, 39.75$ . Numerical solutions of (4.5) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time $T = 50$ . $\Delta t = 0.017$ . The numbers of spatial grid points are $N_A = 120$ , $N_B = 120$ , $N_{E_A} = 60$ . .	88
3.9	Distribution of A and B with different $E_A$ values, at time $T = 0, 20, 35, 50$ . Numerical solutions of (4.5) using the AcIIF2-WENO scheme with Krylov subspace approximations. $\Delta t = 0.017$ . The numbers of spatial grid points are $N_A = 120$ , $N_B = 120$ , $N_{E_A} = 60$ . . . . .	88
3.10	Distribution of A and B with $E_A = 4.5, 12, 19.5, 27$ and $E_B = 15$ . Numerical solutions of (3.17) using the KrylovIIF2-WENO scheme. Final time $T = 10$ . $\Delta t = 0.1$ . The numbers of spatial grid points are $N_A = 40$ , $N_B = 40$ , $N_{E_A} = 20$ , $N_{E_B} = 20$ . . . . .	93
3.11	Distribution of A and B with $E_A = 4.5, 12, 19.5, 27$ and $E_B = 15$ . Numerical solutions of (3.17) using the KrylovIIF2-WENO scheme. Final time $T = 50$ . $\Delta t = 0.1$ . The numbers of spatial grid points are $N_A = 40$ , $N_B = 40$ , $N_{E_A} = 20$ , $N_{E_B} = 20$ . . . . .	93
3.12	Distribution of A and B with $E_A = 15$ and $E_B = 15$ , at time $T = 0, 10, 30, 50$ . Numerical solutions of (3.17) using the KrylovIIF2-WENO scheme. $\Delta t = 0.1$ . The numbers of spatial grid points are $N_A = 40$ , $N_B = 40$ , $N_{E_A} = 20$ , $N_{E_B} = 20$ . . . . .	94
3.13	Distribution of A and B with $E_A = 4.5, 12, 19.5, 27$ and $E_B = 15$ . Numerical solutions of (3.17) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time $T = 10$ . $\Delta t = 0.1$ . The numbers of spatial grid points are $N_A = 40$ , $N_B = 40$ , $N_{E_A} = 20$ , $N_{E_B} = 20$ . . . . .	94
3.14	Distribution of A and B with $E_A = 4.5, 12, 19.5, 27$ and $E_B = 15$ . Numerical solutions of (3.17) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time $T = 50$ . $\Delta t = 0.1$ . The numbers of spatial grid points are $N_A = 40$ , $N_B = 40$ , $N_{E_A} = 20$ , $N_{E_B} = 20$ . . . . .	95

3.15	Distribution of A and B with $E_A = 15$ and $E_B = 15$ , at time $T = 0, 10, 30, 50$ . Numerical solutions of (3.17) using the AcIIF2-WENO scheme with Krylov subspace approximations. $\Delta t = 0.1$ . The numbers of spatial grid points are $N_A = 40$ , $N_B = 40$ , $N_{E_A} = 20$ , $N_{E_B} = 20$ . . .	95
4.1	Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on single-grid. Final time $T = 10$ . $\Delta t = 0.015$ . Distribution of molecular species A and B with $E_A = 12.66, 21.09, 29.53, 37.97$ . . .	123
4.2	Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on sparse-grid. Final time $T = 10$ . $\Delta t = 0.015$ . Distribution of molecular species A and B with $E_A = 12.66, 21.09, 29.53, 37.97$ . . .	125
4.3	Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on single-grid. Final time $T = 50$ . $\Delta t = 0.015$ . Distribution of molecular species A and B with $E_A = 12.66, 21.09, 29.53, 37.97$ . . .	125
4.4	Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on sparse-grid. Final time $T = 50$ . $\Delta t = 0.015$ . Distribution of molecular species A and B with $E_A = 12.66, 21.09, 29.53, 37.97$ . . .	126
4.5	Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on single-grid. Distribution of molecular species A and B with different $E_A$ values, at time $T = 0, 20, 35, 50$ . $\Delta t = 0.015$ . . . . .	126
4.6	Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on sparse-grid. Distribution of molecular species A and B with different $E_A$ values, at time $T = 0, 20, 35, 50$ . $\Delta t = 0.015$ . . . . .	127

## TABLES

3.1	EXAMPLE 1. 2D CASE, CIIF2 SCHEME . . . . .	43
3.2	EXAMPLE 1. 2D CASE, KRYLOVIIF2 SCHEME . . . . .	44
3.3	EXAMPLE 1. 2D CASE, CIIF3 SCHEME . . . . .	45
3.4	EXAMPLE 1. 2D CASE, KRYLOVIIF3 SCHEME . . . . .	46
3.5	EXAMPLE 1. 3D CASE, CIIF2 SCHEME . . . . .	47
3.6	EXAMPLE 1. 3D CASE, KRYLOVIIF2 SCHEME . . . . .	48
3.7	EXAMPLE 2. CIIF2 SCHEME . . . . .	50
3.8	EXAMPLE 2. KRYLOVIIF2 SCHEME . . . . .	51
3.9	EXAMPLE 3. KRYLOVIIF2 SCHEME . . . . .	54
3.10	EXAMPLE 3. DIRECT ACIIF2 SCHEME . . . . .	55
3.11	EXAMPLE 3. ACIIF2 SCHEME WITH KRYLOV SUBSPACE AP- PROXIMATIONS . . . . .	56
3.12	EXAMPLE 4. KRYLOVIIF2 SCHEME . . . . .	59
3.13	EXAMPLE 4. DIRECT ACIIF2 SCHEME . . . . .	60
3.14	EXAMPLE 4. ACIIF2 SCHEME WITH KRYLOV SUBSPACE AP- PROXIMATIONS . . . . .	61
3.15	EXAMPLE 5. KRYLOVIIF2 SCHEME . . . . .	64
3.16	EXAMPLE 5. DIRECT ACIIF2 SCHEME . . . . .	65
3.17	EXAMPLE 5. ACIIF2 SCHEME WITH KRYLOV SUBSPACE AP- PROXIMATIONS . . . . .	66
3.18	EXAMPLE 7. KRYLOVIIF2-WENO SCHEME . . . . .	73
3.19	EXAMPLE 7. ACIIF2-WENO SCHEME WITH KRYLOV SUBSPACE APPROXIMATIONS . . . . .	74
3.20	EXAMPLE 8. KRYLOVIIF2-WENO SCHEME . . . . .	76
3.21	EXAMPLE 8. ACIIF2-WENO SCHEME WITH KRYLOV SUBSPACE APPROXIMATIONS . . . . .	77
3.22	CPU TIME COMPARISONS FOR SOLVING THE NONLINEAR VISCOUS BURGERS' EQUATION . . . . .	80

3.23	NUMERICAL ERRORS AND ACCURACY ORDERS FOR THE KRYLOVIIF2 SCHEME AND THE ACIIF2 SCHEME WITH KRYLOV SUBSPACE APPROXIMATIONS FOR THE 3D FOKKER-PLANCK EQUATION (4.5)	84
3.24	CPU TIME FOR KRYLOVIIF2 SCHEME AND THE ACIIF2 SCHEME WITH KRYLOV SUBSPACE APPROXIMATIONS FOR THE 3D FOKKER-PLANCK EQUATION (4.5)	85
3.25	NUMERICAL ERRORS AND ACCURACY ORDERS FOR THE KRYLOVIIF2 SCHEME AND THE ACIIF2 SCHEME WITH KRYLOV SUBSPACE APPROXIMATIONS FOR THE 4D FOKKER-PLANCK EQUATION (3.17)	91
3.26	CPU TIME FOR KRYLOVIIF2 SCHEME AND THE ACIIF2 SCHEME WITH KRYLOV SUBSPACE APPROXIMATIONS FOR THE 4D FOKKER-PLANCK EQUATION (3.17)	92
4.1	EXAMPLE 1, 2D CASE, KRYLOVIIF2 SCHEME, COMPARISON OF NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON SINGLE-GRID AND SPARSE-GRID.	103
4.2	EXAMPLE 1, 2D CASE, KRYLOVIIF3 SCHEME, COMPARISON OF NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON SINGLE-GRID AND SPARSE-GRID.	104
4.3	EXAMPLE 1, 3D CASE, KRYLOVIIF2 SCHEME, COMPARISON OF NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON SINGLE-GRID AND SPARSE-GRID.	105
4.4	EXAMPLE 1, 2D CASE, KRYLOVIIF2 SCHEME, COMPARISON OF THE NUMBER OF MULTIPLICATION AND DIVISION OPERATIONS IN ONE TIME STEP.	107
4.5	EXAMPLE 1, 2D CASE, CRANK-NICOLSON SCHEME WITH A MULTIGRID SOLVER (THE TWO-GRID CORRECTION SCHEME) FOR THE LINEAR SYSTEMS.	107
4.6	EXAMPLE 1, 2D CASE, KRYLOVIIF2 SCHEME, NON-UNIFORM GRIDS. COMPARISON OF NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON SINGLE-GRID AND SPARSE-GRID.	108
4.7	EXAMPLE 2. KRYLOVIIF2 SCHEME, COMPARISON OF NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON SINGLE-GRID AND SPARSE-GRID.	111
4.8	EXAMPLE 3. KRYLOVIIF2 SCHEME, COMPARISON OF NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON SINGLE-GRID AND SPARSE-GRID.	115
4.9	EXAMPLE 4. KRYLOVIIF2 SCHEME, COMPARISON OF NUMERICAL ERRORS AND CPU TIMES	118

4.10	NUMERICAL ERRORS AND ACCURACY ORDERS FOR THE KRYLOVIIF2 SCHEME TO SOLVE THE 3D FOKKER-PLANCK EQUATION ON SINGLE AND SPARSE GRIDS. . . . .	124
4.11	CPU TIME FOR THE KRYLOVIIF2 SCHEME TO SOLVE THE 3D FOKKER-PLANCK EQUATION ON SINGLE AND SPARSE GRIDS . . . . .	124

## ACKNOWLEDGMENTS

First I would like to thank my advisor Dr. Yong-Tao Zhang, for the continuous support of my graduate study and research, for his patience, encouragement, enthusiasm, and immense knowledge.

I would like to thank Dr. Bei Hu and Dr. Martina Bukac for being my dissertation readers. I appreciate their helpful comments and suggestions. I would also like to thank other faculty at Notre Dame for their lecture and help.

I would like to thank my group members including Yuan Liu, Tian Jiang, Liang Wu and Michael Machen for their suggestions and encouragement to my research. I especially would like to thank every member of the Department of Applied and Computational Mathematics and Statistics for building such a great environment which respects and supports each individual's development. I would also like to thank my other friends at Notre Dame for making my graduate study an enjoyable journey.

I would like to thank my parents. They were always supporting me and encouraging me with their best wishes.

Finally, I would like to thank my husband Jianxu Chen and my daughter Lily Chen. They are the source of my joy and inspiration.

## CHAPTER 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 [13, 14, 45, 47]; spectral deferred correction (SDC) methods [4, 11, 21, 32, 39]; high order implicit-explicit (IMEX) multistep / RK methods [1, 9, 27, 52, 57]; hybrid methods of SDC and high order RK schemes [8]; 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, 10, 26, 28, 38]. Here time direction numerical errors are numerical errors for solving the semi-discretized ODE system resulting from spatial discretizations of the PDE. In [42], 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 [42] from implicit exponential time differencing (ETD) methods in [3].

Nonlinear convection-diffusion-reaction (CDR) systems of equations [22] 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

$$\vec{u}_t + \sum_{i=1}^d \vec{f}_i(\vec{u})_{x_i} = \nabla \cdot (\mathbf{D}(\vec{u}) \nabla \vec{u}) + \vec{r}(\vec{u}), \quad (1.1)$$

where  $\vec{u}$  is the unknown vector function,  $\vec{f}_i, i = 1, \dots, d$  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 (1.1), 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 [37]. 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 [17, 47]. The first WENO scheme was constructed in [34] for a third order finite volume version. In [23], 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. [20, 35, 55, 56].

Recently, we developed IIF-WENO methods for solving nonlinear CDR systems in [24]. 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 (1.1) defined on high dimensional spatial domains, the major 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 [33, 43]. 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 [53] 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 [7], 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 the first part of this dissertation, 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.

In the second part of this dissertation, 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 underlying 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 [6]. Sparse-grid techniques were introduced by Zenger [54] 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. [15], 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 [15, 30, 31].

The rest of this dissertation is organized as following. In chapter 2, we first review the IIF-WENO methods for solving CDR equations developed in [24]. Then we present two different approach to deal with the high dimensional problems, i.e., the direct Krylov approach and the AcIIF approach. In order to compare it with the Krylov approach, we combine the AcIIF method with WENO method for solving CDR equations. After that, Krylov IIF methods on sparse grids are developed. In chapter 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. In chapter 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 chapter 5.

## CHAPTER 2

### NUMERICAL METHODS

In section 2.1, we briefly review the IIF-WENO methods for solving CDR equations developed in [24]. In section 2.2, we present two approaches for dealing with high dimensional problems: Krylov approximation method and compact/array-representation method. For the AcIIF method designed in [53], we combine it with WENO method and derive the corresponding schemes for solving CDR equations. In section 2.3, we develop the krylov IF methods on sparse grids. In section 2.4, linear stability analysis of the IIF2 scheme(2.10) for CDR equations is given.

#### 2.1 IIF-WENO methods

The method of lines (MOL) approach is applied to the equation (1.1). 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 [46] 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

$$f(u)_x|_{x=x_i} \approx \frac{1}{\Delta x}(\hat{f}_{i+1/2} - \hat{f}_{i-1/2}), \quad (2.1)$$

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

$$\hat{f}_{i+1/2} = w_0 \left[ \frac{1}{2}f(u_i) + \frac{1}{2}f(u_{i+1}) \right] + w_1 \left[ -\frac{1}{2}f(u_{i-1}) + \frac{3}{2}f(u_i) \right], \quad (2.2)$$

where

$$w_r = \frac{\alpha_r}{\alpha_1 + \alpha_2}, \quad \alpha_r = \frac{d_r}{(\epsilon + \beta_r)^2}, \quad r = 0, 1. \quad (2.3)$$

$d_0 = 2/3, d_1 = 1/3$  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”.  $\epsilon$  is a small positive number chosen to avoid the denominator becoming 0. We take  $\epsilon = 10^{-3}$  in this dissertation.

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"

$$f^+(u) = \frac{1}{2}(f(u) + \alpha u), \quad f^-(u) = \frac{1}{2}(f(u) - \alpha u), \quad (2.4)$$

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 [23, 46] for more details. For diffusion terms, central differences are used. After spatial discretizations, a semi-discretized ODE system

$$\frac{d\vec{U}}{dt} = \vec{F}_d(\vec{U}) + \vec{F}_a(\vec{U}) + \vec{R}(\vec{U}) \quad (2.5)$$

is obtained. Here  $\vec{U} = (u_i)_{1 \leq i \leq N}$ ,  $\vec{F}_d(\vec{U}) = (\hat{F}_{di}(\vec{U}))_{1 \leq i \leq N}$ ,  $\vec{F}_a(\vec{U}) = (\hat{F}_{ai}(\vec{U}))_{1 \leq i \leq N}$ ,  $\vec{R} = (r(u_i))_{1 \leq i \leq N}$ .  $N$  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 [24], 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 dissertation, 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

(2.5) are constructed by exactly integrating the linear part of the system. Directly multiply (2.5) 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

$$\begin{aligned}\vec{U}(t_{n+1}) &= e^{C\Delta t_n} \vec{U}(t_n) + e^{C\Delta t_n} \int_0^{\Delta t_n} e^{-C\tau} \vec{F}_a(\vec{U}(t_n + \tau)) d\tau \\ &\quad + e^{C\Delta t_n} \int_0^{\Delta t_n} e^{-C\tau} \vec{R}(\vec{U}(t_n + \tau)) d\tau.\end{aligned}\tag{2.6}$$

Two of the nonlinear terms in (2.6) 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 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  $t_{n+1}, t_n, \dots, t_{n+2-r}$ . 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  $t_n, t_{n-1}, \dots, t_{n+1-r}$ . The  $r$ -th order IIF scheme for CDR equations is obtained as

$$\begin{aligned}\vec{U}_{n+1} &= e^{C\Delta t_n} \vec{U}_n + \Delta t_n \{ \alpha_{n+1} \vec{R}(\vec{U}_{n+1}) + \sum_{i=2-r}^0 \alpha_{n+i} e^{C(\Delta t_n - \tau_i)} \vec{R}(\vec{U}_{n+i}) \\ &\quad + \sum_{i=1-r}^0 \beta_{n+i} e^{C(\Delta t_n - \tau_i)} \vec{F}_a(\vec{U}_{n+i}) \},\end{aligned}\tag{2.7}$$

where the coefficients

$$\alpha_{n+i} = \frac{1}{\Delta t_n} \int_0^{\Delta t_n} \prod_{j=2-r, j \neq i}^1 \frac{\tau - \tau_j}{\tau_i - \tau_j} d\tau, \quad i = 1, 0, -1, \dots, 2-r;\tag{2.8}$$



$$\beta_{n+i} = \frac{1}{\Delta t_n} \int_0^{\Delta t_n} \prod_{j=1-r, j \neq i}^0 \frac{\tau - \tau_j}{\tau_i - \tau_j} d\tau, \quad i = 0, -1, -2, \dots, 1-r. \quad (2.9)$$

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

$$\begin{aligned} \vec{U}_{n+1} = & e^{C\Delta t_n} \vec{U}_n + \Delta t_n \left\{ \alpha_{n+1} \vec{R}(\vec{U}_{n+1}) + \alpha_n e^{C\Delta t_n} \vec{R}(\vec{U}_n) \right. \\ & \left. + \beta_{n-1} e^{C(\Delta t_n + \Delta t_{n-1})} \vec{F}_a(\vec{U}_{n-1}) + \beta_n e^{C\Delta t_n} \vec{F}_a(\vec{U}_n) \right\}, \end{aligned} \quad (2.10)$$

where

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

And the third order scheme (IIF3) is

$$\begin{aligned} \vec{U}_{n+1} = & e^{C\Delta t_n} \vec{U}_n + \Delta t_n \left\{ \alpha_{n+1} \vec{R}(\vec{U}_{n+1}) + \alpha_n e^{C\Delta t_n} \vec{R}(\vec{U}_n) + \alpha_{n-1} e^{C(\Delta t_n + \Delta t_{n-1})} \vec{R}(\vec{U}_{n-1}) \right. \\ & \left. + \beta_{n-2} e^{C(\Delta t_n + \Delta t_{n-1} + \Delta t_{n-2})} \vec{F}_a(\vec{U}_{n-2}) + \beta_{n-1} e^{C(\Delta t_n + \Delta t_{n-1})} \vec{F}_a(\vec{U}_{n-1}) + \beta_n e^{C\Delta t_n} \vec{F}_a(\vec{U}_n) \right\}, \end{aligned} \quad (2.11)$$

where

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

**Remark:** Theoretical analysis including stability and error analysis of the IIF schemes for convection-diffusion-reaction equations is given in [24, 25]. Due to the nonlinearity of WENO schemes [46] 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 (2.7). 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 [7]. It has been applied for solving CDR equations in [24].

### 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 (2.7). 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. [12, 41]), 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

$$K_M = \text{span}\{v, Cv, C^2v, \dots, C^{M-1}v\}. \quad (2.12)$$

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 dissertation, we take  $M = 25$  for different  $N$ , and accurate results are obtained in the numerical experiments. An orthonormal basis  $V_M = [v_1, v_2, v_3, \dots, v_M]$  of the Krylov subspace  $K_M$  is generated by the well-known Arnoldi algorithm [51]:

1. Compute the initial vector:  $v_1 = v/\|v\|_2$ .
2. Perform iterations: Do  $j = 1, 2, \dots, M$ :
  - 1) Compute the vector  $w = Cv_j$ .
  - 2) Do  $i = 1, 2, \dots, j$ :
    - (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  $K_M = \text{span}\{v_1, v_2, \dots, v_j\}$ , 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

$$H_M = V_M^T C V_M. \quad (2.13)$$

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 (2.13) we obtain the approximation

$$e^{C\Delta t} v \simeq \beta V_M e^{H_M \Delta t} e_1. \quad (2.14)$$

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 [12, 18, 41]. Then the Krylov approximations are directly applied in schemes (2.7), (2.10) or (2.11) to obtain Krylov IIF schemes for CDR equations [24]. The  $r$ -th order Krylov IIF scheme for CDR equations has the following form

$$\begin{aligned} \vec{U}_{n+1} = & \Delta t_n \alpha_{n+1} \vec{R}(\vec{U}_{n+1}) + \gamma_{0,n} V_{M,0,n} e^{H_{M,0,n} \Delta t_n} e_1 \\ & + \Delta t_n \left( \beta_{n+1-r} \gamma_{1-r,n} V_{M,1-r,n} e^{H_{M,1-r,n} (\Delta t_n - \tau_{1-r})} e_1 + \sum_{i=2-r}^{-1} \gamma_{i,n} V_{M,i,n} e^{H_{M,i,n} (\Delta t_n - \tau_i)} e_1 \right), \end{aligned} \quad (2.15)$$

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  $i = 2 - r, 3 - r, \dots, -1$ . Notice that  $V_{M,0,n}$ ,  $V_{M,1-r,n}$  and  $V_{M,i,n}$ ,  $i = 2 - r, 3 - r, \dots, -1$  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

$$\begin{aligned} \vec{U}_{n+1} = & \frac{1}{2} \Delta t_n \vec{R}(\vec{U}_{n+1}) + \gamma_{0,n} V_{M,0,n} e^{H_{M,0,n} \Delta t_n} e_1 \\ & - \frac{(\Delta t_n)^2}{2 \Delta t_{n-1}} (\gamma_{-1,n} V_{M,-1,n} e^{H_{M,-1,n}(\Delta t_n + \Delta t_{n-1})} e_1), \end{aligned} \quad (2.16)$$

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

$$\begin{aligned} \vec{U}_{n+1} = & \frac{2\Delta t_n + 3\Delta t_{n-1}}{6(\Delta t_n + \Delta t_{n-1})} \Delta t_n \vec{R}(\vec{U}_{n+1}) + \gamma_{0,n} V_{M,0,n} e^{H_{M,0,n} \Delta t_n} e_1 \\ & + \Delta t_n \left( \frac{2(\Delta t_n)^2 + 3\Delta t_n \Delta t_{n-1}}{6\Delta t_{n-2}(\Delta t_{n-1} + \Delta t_{n-2})} \gamma_{-2,n} V_{M,-2,n} e^{H_{M,-2,n}(\Delta t_n + \Delta t_{n-1} + \Delta t_{n-2})} e_1 \right. \\ & \left. + \gamma_{-1,n} V_{M,-1,n} e^{H_{M,-1,n}(\Delta t_n + \Delta t_{n-1})} e_1 \right), \end{aligned} \quad (2.17)$$

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 equation (2.11) for values of  $\alpha_n, \beta_n, \alpha_{n-1}, \beta_{n-1}$ .

As that pointed out in [24], 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 [12, 19], an error estimation of the Krylov subspace approximation (2.14) is

$$\|e^{C\Delta t}v - \beta V_M e^{H_M \Delta t} e_1\|_2 \leq 10\beta e^{-M^2/(5\rho\Delta t)}, \quad (2.18)$$

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 (2.18) 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 [43] and [53]. Then we discuss how to apply the cIIF / AcIIF method in the IIF-WENO schemes for solving high dimensional CDR equations.

### 2.2.2.1 cIIF/AcIIF for reaction-diffusion equations

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

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + R(u), \quad (x, y) \in \Omega = \{a < x < b, c < y < d\}, \quad (2.19)$$

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

$$\frac{du_{i,j}}{dt} = D\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2}\right) + R(u_{i,j}) \quad (2.20)$$

is obtained. The idea of cIIF method [43] is that instead of representing numerical values  $u_{i,j}$  in a large vector, numerical values are organized and stored in a matrix (see (2.22)). The semi-discretized ODE system is written in a compact form

$$\frac{d\mathbf{U}}{dt} = \mathbf{A}\mathbf{U} + \mathbf{U}\mathbf{B} + R(\mathbf{U}), \quad (2.21)$$

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

$$\mathbf{U}_{N_x \times (N_y+1)} = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,N_y} & u_{1,N_y+1} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,N_y} & u_{2,N_y+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{N_x,1} & u_{N_x,2} & \cdots & u_{N_x,N_y} & u_{N_x,N_y+1} \end{pmatrix}, \quad (2.22)$$

$$\mathbf{A}_{N_x \times N_x} = \frac{D}{h_x^2} \begin{pmatrix} -\frac{2}{3} & \frac{2}{3} & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & \frac{2}{3} & -\frac{2}{3} \end{pmatrix}, \quad (2.23)$$

$$\mathbf{B}_{(N_y+1) \times (N_y+1)} = \frac{D}{h_y^2} \begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 1 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & 1 & -2 & 1 \\ 1 & 0 & \cdots & 0 & 1 & -2 \end{pmatrix}. \quad (2.24)$$

Then following the similar procedure for deriving IIF methods [42], we multiply (2.21) by the integration factors  $e^{-\mathbf{A}t}$  from the left and  $e^{-\mathbf{B}t}$  from the right, and integrate over one time step from  $t_n$  to  $t_{n+1} \equiv t_n + \Delta t$  to obtain

$$\mathbf{U}_{n+1} = e^{\mathbf{A}\Delta t} \mathbf{U}_n e^{\mathbf{B}\Delta t} + e^{\mathbf{A}\Delta t} \left( \int_0^{\Delta t} e^{-\mathbf{A}\tau} R(\mathbf{U}(t_n + \tau)) e^{-\mathbf{B}\tau} d\tau \right) e^{\mathbf{B}\Delta t}. \quad (2.25)$$

We approximate the integrand in (2.25) by an  $(r-1)$ th order lagrange interpolation polynomial with interpolation points at  $t_{n+1}, t_n, \dots, t_{n+2-r}$ , and obtain the  $r$ th order cIIF scheme for two-dimensional reaction-diffusion equations

$$\mathbf{U}_{n+1} = e^{\mathbf{A}\Delta t} \mathbf{U}_n e^{\mathbf{B}\Delta t} + \Delta t \left( \alpha_1 R(\mathbf{U}_{n+1}) + \sum_{i=0}^{r-2} \alpha_{-i} e^{(i+1)\mathbf{A}\Delta t} R(\mathbf{U}_{n-i}) e^{(i+1)\mathbf{B}\Delta t} \right), \quad (2.26)$$

where

$$\alpha_{-i} = \frac{1}{\Delta t} \int_0^{\Delta t} \prod_{\substack{k=-1 \\ k \neq i}}^{r-2} \frac{\tau + k\Delta t}{(k-i)\Delta t} d\tau, \quad -1 \leq i \leq r-2. \quad (2.27)$$



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

$$\mathbf{U}_{n+1} = e^{\mathbf{A}\Delta t} \left( \mathbf{U}_n + \frac{\Delta t}{2} R(\mathbf{U}_n) \right) e^{\mathbf{B}\Delta t} + \frac{\Delta t}{2} R(\mathbf{U}_{n+1}). \quad (2.28)$$

Note that the matrices  $A$  and  $B$  have sizes of a one-dimensional problem. Hence in cIIF schemes (2.26), (2.28) 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 [43].

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 [53]. We review the AcIIF method [53] 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  $U = (U_{k_1, k_2}), k_1 = 1, \dots, N_x; k_2 = 1, \dots, N_y + 1$  for the two-dimensional problem (2.20)-(2.24). If we fix the second index  $k_2$ , the two-dimensional array  $U$  defines a vector

$$U(:, k_2) = (U_{1, k_2}, U_{2, k_2}, \dots, U_{N_x, k_2})^T. \quad (2.29)$$

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 [53], so we can write

$$U = \bigotimes_{1 \leq k_2 \leq N_y + 1} U(:, k_2). \quad (2.30)$$

The finite difference operators are linear operators in (2.20) since the diffusion terms

here are linear. Define finite difference operators  $\mathcal{L}_x$  and  $\mathcal{L}_y$  as

$$(\mathcal{L}_x U)_{k_1, k_2} = D \left( \frac{U_{k_1+1, k_2} - 2U_{k_1, k_2} + U_{k_1-1, k_2}}{h_x^2} \right), \quad (2.31)$$

and

$$(\mathcal{L}_y U)_{k_1, k_2} = D \left( \frac{U_{k_1, k_2+1} - 2U_{k_1, k_2} + U_{k_1, k_2-1}}{h_y^2} \right), \quad (2.32)$$

then the semi-discretized scheme (2.20) with the array  $U$  can be written as

$$\frac{dU}{dt} = (\mathcal{L}_x + \mathcal{L}_y)U + R(U). \quad (2.33)$$

Apply IIF schemes, e.g., the second order IIF scheme (IIF2) [42] in (2.33) to obtain

$$U_{n+1} = e^{(\mathcal{L}_x + \mathcal{L}_y)\Delta t} \left( U_n + \frac{\Delta t}{2} R(U_n) \right) + \frac{\Delta t}{2} R(U_{n+1}). \quad (2.34)$$

To implement the scheme (2.34) using array-representation technique, we first represent

$$\mathcal{L}_x U = \bigotimes_{1 \leq k_2 \leq N_y+1} \mathbf{A} U(:, k_2), \quad (2.35)$$

where  $\mathbf{A}$  is given in (2.23). So the exponential of  $\mathcal{L}_x$  can have the array-representation

$$e^{\mathcal{L}_x \Delta t} U = \bigotimes_{1 \leq k_2 \leq N_y+1} e^{\mathbf{A} \Delta t} U(:, k_2). \quad (2.36)$$

Similarly,

$$e^{\mathcal{L}_y \Delta t} U = \bigotimes_{1 \leq k_1 \leq N_x} e^{\mathbf{B} \Delta t} U(k_1, :), \quad (2.37)$$

where  $\mathbf{B}$  is given in (2.24). Since  $\mathcal{L}_x$  and  $\mathcal{L}_y$  commute with each other for this constant diffusion coefficient equation case,  $e^{(\mathcal{L}_x + \mathcal{L}_y)\Delta t} = e^{\mathcal{L}_x \Delta t} e^{\mathcal{L}_y \Delta t}$ . The array-representation form of the IIF2 scheme [42], i.e., the AcIIF2 scheme for the 2D reaction-diffusion

equation (2.19), is

$$U_{n+1} - \frac{\Delta t}{2} R(U_{n+1}) = \bigotimes_{1 \leq k_2 \leq N_y + 1} e^{\mathbf{A} \Delta t} \left( \bigotimes_{1 \leq k_1 \leq N_x} e^{\mathbf{B} \Delta t} V(k_1, :) \right) (:, k_2), \quad (2.38)$$

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

$$\begin{aligned} & U_{n+1} - \frac{\Delta t}{2} R(U_{n+1}) \\ &= \bigotimes_{\substack{1 \leq k_2 \leq N_y \\ 1 \leq k_3 \leq N_z}} e^{\mathbf{A}_{11} \Delta t} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_x \\ 1 \leq k_3 \leq N_z}} e^{\mathbf{A}_{22} \Delta t} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_x \\ 1 \leq k_2 \leq N_y}} e^{\mathbf{A}_{33} \Delta t} V(k_1, k_2, :) \right) (k_1, :, k_3) \right) (:, k_2, k_3), \end{aligned} \quad (2.39)$$

where  $V = U_n + \frac{\Delta t}{2} R(U_n)$ ,  $U$  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.  $\mathbf{A}_{11}, \mathbf{A}_{22}, \mathbf{A}_{33}$  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 (2.38) is equivalent to the cIIF2 scheme (2.28). As that pointed out in [53], 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 [53].

#### 2.2.2.2 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 chapter 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 (1.1),  $d = 3$ , 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

$$\mathbf{D} = \begin{pmatrix} a_1 + a_2 & b_1 & b_2 \\ b_1 & a_3 + c_1 & b_3 \\ b_2 & b_3 & c_2 + c_3 \end{pmatrix}, \quad (2.40)$$

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 (2.10) to the equation and re-grouping the exponential terms, we obtain

$$\begin{aligned} \vec{U}_{n+1} &= e^{C\Delta t_n} \left( \vec{U}_n + \Delta t_n \alpha_n \vec{R}(\vec{U}_n) + \Delta t_n \beta_n \vec{F}_a(\vec{U}_n) \right) \\ &\quad + e^{C(\Delta t_n + \Delta t_{n-1})} \left( \Delta t_n \beta_{n-1} \vec{F}_a(\vec{U}_{n-1}) \right) + \Delta t_n \alpha_{n+1} \vec{R}(\vec{U}_{n+1}) \\ &= \Theta_1 + \Theta_2 + \Delta t_n \alpha_{n+1} \vec{R}(\vec{U}_{n+1}), \end{aligned} \quad (2.41)$$

where

$$\Theta_1 = e^{C\Delta t_n} \vec{V}_1, \quad \vec{V}_1 \triangleq \vec{U}_n + \Delta t_n \alpha_n \vec{R}(\vec{U}_n) + \Delta t_n \beta_n \vec{F}_a(\vec{U}_n), \quad (2.42)$$

$$\Theta_2 = e^{C(\Delta t_n + \Delta t_{n-1})} \vec{V}_2, \quad \vec{V}_2 \triangleq \Delta t_n \beta_{n-1} \vec{F}_a(\vec{U}_{n-1}). \quad (2.43)$$

$\alpha_n, \alpha_{n+1}, \beta_{n-1}, \beta_n$  are given in (2.10). 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

$$\begin{aligned} (\mathcal{L}_{12}U)_{k_1, k_2, k_3} = & \frac{a_1}{h_1^2} (U_{k_1+1, k_2, k_3} - 2U_{k_1, k_2, k_3} + U_{k_1-1, k_2, k_3}) \\ & + \frac{2b_1}{4h_1 h_2} (U_{k_1+1, k_2+1, k_3} + U_{k_1-1, k_2-1, k_3} - U_{k_1+1, k_2-1, k_3} - U_{k_1-1, k_2+1, k_3}) \\ & + \frac{c_1}{h_2^2} (U_{k_1, k_2+1, k_3} - 2U_{k_1, k_2, k_3} + U_{k_1, k_2-1, k_3}). \end{aligned} \quad (2.44)$$

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

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

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

$$\begin{aligned} (\mathcal{A}_{12}M)_{i,j} = & \frac{2b_1}{4h_1 h_2} (M_{i+1, j+1} + M_{i-1, j-1} - M_{i-1, j+1} - M_{i+1, j-1}) \\ & + \frac{a_1}{h_1^2} (M_{i+1, j} - 2M_{i, j} + M_{i-1, j}) + \frac{c_1}{h_2^2} (M_{i, j+1} - 2M_{i, j} + M_{i, j-1}). \end{aligned} \quad (2.45)$$

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

$$\mathcal{L}_{12}U = \bigotimes_{1 \leq k_3 \leq N_3} \mathcal{A}_{12}U(:, :, k_3),$$

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

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 (2.41) - (2.43), 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 (2.42) and (2.43) 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 (2.50) in section 2.2.2.3.

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 [49] is needed to obtain a second order accuracy. By the Strang symmetric operator splitting, we have

$$e^{C\Delta t_n} = e^{(\mathcal{L}_{12}+\mathcal{L}_{13}+\mathcal{L}_{23})\Delta t_n} = e^{\frac{\Delta t_n}{2}\mathcal{L}_{12}}e^{\frac{\Delta t_n}{2}\mathcal{L}_{13}}e^{\Delta t_n\mathcal{L}_{23}}e^{\frac{\Delta t_n}{2}\mathcal{L}_{13}}e^{\frac{\Delta t_n}{2}\mathcal{L}_{12}} + O(\Delta t_n^3). \quad (2.46)$$

Then array representations are applied in (2.42) and (2.43) for decomposition of large

matrix exponentials of  $C$ . See the equations in (2.51) and (2.52) in section 2.2.2.3 for detailed implementation formulas.

Similarly, for a four dimensional CDR equation (1.1),  $d = 4$ , with cross-derivatives for the linear diffusion terms and periodic boundary conditions, the diffusion matrix  $\mathbf{D}$  is

$$\mathbf{D} = \begin{pmatrix} a_1 + a_2 + a_3 & b_1 & b_2 & b_3 \\ b_1 & a_4 + a_5 + c_1 & b_4 & b_5 \\ b_2 & b_4 & a_6 + c_2 + c_4 & b_6 \\ b_3 & b_5 & b_6 & c_3 + c_5 + c_6 \end{pmatrix}, \quad (2.47)$$

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 (2.10) and obtain the same form schemes (2.41)-(2.43), 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

$$\begin{aligned} (\mathcal{L}_{12}U)_{k_1, k_2, k_3, k_4} &= \frac{a_1}{h_1^2} (U_{k_1+1, k_2, k_3, k_4} - 2U_{k_1, k_2, k_3, k_4} + U_{k_1-1, k_2, k_3, k_4}) \\ &\quad + \frac{2b_1}{4h_1 h_2} (U_{k_1+1, k_2+1, k_3, k_4} + U_{k_1-1, k_2-1, k_3, k_4} - U_{k_1+1, k_2-1, k_3, k_4} \\ &\quad - U_{k_1-1, k_2+1, k_3, k_4}) + \frac{c_1}{h_2^2} (U_{k_1, k_2+1, k_3, k_4} - 2U_{k_1, k_2, k_3, k_4} + U_{k_1, k_2-1, k_3, k_4}). \end{aligned} \quad (2.48)$$

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

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

The same linear mapping  $\mathcal{A}_{12}$  is defined as (2.45) for three dimensional cases. The array representation of  $\mathcal{L}_{12}$  and its exponential are

$$\mathcal{L}_{12}U = \bigotimes_{\substack{1 \leq k_3 \leq N_3 \\ 1 \leq k_4 \leq N_4}} \mathcal{A}_{12}U(:, :, k_3, k_4),$$

$$e^{\mathcal{L}_{12}\Delta t}U = \bigotimes_{\substack{1 \leq k_3 \leq N_3 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{12}\Delta t}U(:, :, k_3, k_4).$$

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 (2.41) - (2.43), 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 (2.41) - (2.43) to decompose large matrix exponentials for  $C$  to much smaller ones. For detailed formulas in implementation of the method, see the equations (2.53) and (2.54) in section 2.2.2.3.

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

$$e^{C\Delta t_n} = e^{(\mathcal{L}_{12} + \mathcal{L}_{13} + \mathcal{L}_{14} + \mathcal{L}_{23} + \mathcal{L}_{24} + \mathcal{L}_{34})\Delta t_n} = e^{\frac{\Delta t_n}{2}\mathcal{L}_{34}} e^{\frac{\Delta t_n}{2}\mathcal{L}_{24}} e^{\frac{\Delta t_n}{2}\mathcal{L}_{23}} e^{\frac{\Delta t_n}{2}\mathcal{L}_{14}} e^{\frac{\Delta t_n}{2}\mathcal{L}_{13}} e^{\Delta t_n\mathcal{L}_{12}} \\ e^{\frac{\Delta t_n}{2}\mathcal{L}_{13}} e^{\frac{\Delta t_n}{2}\mathcal{L}_{14}} e^{\frac{\Delta t_n}{2}\mathcal{L}_{23}} e^{\frac{\Delta t_n}{2}\mathcal{L}_{24}} e^{\frac{\Delta t_n}{2}\mathcal{L}_{34}} + O(\Delta t_n^3). \quad (2.49)$$

Then application of array representation in (2.42) and (2.43) leads to decomposition of large matrix exponentials of  $C$  into much smaller ones. See the equations (2.55) - (2.58) in section 2.2.2.3 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 section 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 [43]. As that shown in the numerical experiments of the chapter 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.

### 2.2.2.3 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

$$\begin{aligned}\Theta_1 &= \bigotimes_{1 \leq k_1 \leq N_1} e^{\mathcal{A}_{23} \Delta t_n} \left( \bigotimes_{1 \leq k_2 \leq N_2} e^{\mathcal{A}_{13} \Delta t_n} \left( \bigotimes_{1 \leq k_3 \leq N_3} e^{\mathcal{A}_{12} \Delta t_n} V_1(:, :, k_3) \right) (:, k_2, :) \right) (k_1, :, :), \\ \Theta_2 &= \bigotimes_{1 \leq k_1 \leq N_1} e^{\mathcal{A}_{23} (\Delta t_n + \Delta t_{n-1})} \left( \bigotimes_{1 \leq k_2 \leq N_2} e^{\mathcal{A}_{13} (\Delta t_n + \Delta t_{n-1})} \right. \\ &\quad \left. \left( \bigotimes_{1 \leq k_3 \leq N_3} e^{\mathcal{A}_{12} (\Delta t_n + \Delta t_{n-1})} V_2(:, :, k_3) \right) (:, k_2, :) \right) (k_1, :, :).\end{aligned}\tag{2.50}$$

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

$$\begin{aligned}\Theta_1 &= \bigotimes_{1 \leq k_3 \leq N_3} e^{\mathcal{A}_{12}^{k_3} \frac{\Delta t_n}{2}} \left( \bigotimes_{1 \leq k_2 \leq N_2} e^{\mathcal{A}_{13}^{k_2} \frac{\Delta t_n}{2}} V_1^*(:, k_2, :) \right) (:, :, k_3), \\ V_1^* &= \bigotimes_{1 \leq k_1 \leq N_1} e^{\mathcal{A}_{23}^{k_1} \Delta t_n} \left( \bigotimes_{1 \leq k_2 \leq N_2} e^{\mathcal{A}_{13}^{k_2} \frac{\Delta t_n}{2}} \left( \bigotimes_{1 \leq k_3 \leq N_3} e^{\mathcal{A}_{12}^{k_3} \frac{\Delta t_n}{2}} V_1(:, :, k_3) \right) (:, k_2, :) \right) (k_1, :, :);\end{aligned}\tag{2.51}$$

and

$$\begin{aligned}\Theta_2 &= \bigotimes_{1 \leq k_3 \leq N_3} e^{\mathcal{A}_{12}^{k_3} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \left( \bigotimes_{1 \leq k_2 \leq N_2} e^{\mathcal{A}_{13}^{k_2} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} V_2^*(:, k_2, :) \right) (:, :, k_3), \\ V_2^* &= \bigotimes_{1 \leq k_1 \leq N_1} e^{\mathcal{A}_{23}^{k_1} (\Delta t_n + \Delta t_{n-1})} \left( \bigotimes_{1 \leq k_2 \leq N_2} e^{\mathcal{A}_{13}^{k_2} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \right. \\ &\quad \left. \left( \bigotimes_{1 \leq k_3 \leq N_3} e^{\mathcal{A}_{12}^{k_3} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} V_2(:, :, k_3) \right) (:, k_2, :) \right) (k_1, :, :).\end{aligned}\tag{2.52}$$

(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

$$\begin{aligned} \Theta_1 = & \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_2 \leq N_2}} e^{\mathcal{A}_{34} \Delta t_n} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{24} \Delta t_n} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{23} \Delta t_n} \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{14} \Delta t_n} \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{13} \Delta t_n} \left( \right. \right. \right. \right. \\ & \left. \bigotimes_{\substack{1 \leq k_3 \leq N_3 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{12} \Delta t_n} V_1(:, :, k_3, k_4) \right) (:, k_2, :, k_4) \right) (:, k_2, k_3, :) \right) (k_1, :, :, k_4) \right) (k_1, :, k_3, :) \right) (k_1, k_2, :, :), \end{aligned} \quad (2.53)$$

$$\begin{aligned} \Theta_2 = & \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_2 \leq N_2}} e^{\mathcal{A}_{34}(\Delta t_n + \Delta t_{n-1})} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{24}(\Delta t_n + \Delta t_{n-1})} \right. \\ & \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{23}(\Delta t_n + \Delta t_{n-1})} \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{14}(\Delta t_n + \Delta t_{n-1})} \left( \right. \right. \right. \\ & \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{13}(\Delta t_n + \Delta t_{n-1})} \left( \bigotimes_{\substack{1 \leq k_3 \leq N_3 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{12}(\Delta t_n + \Delta t_{n-1})} V_2(:, :, k_3, k_4) \right) (:, k_2, :, k_4) \right) \\ & \left. (:, k_2, k_3, :) \right) (k_1, :, :, k_4) \right) (k_1, :, k_3, :) \right) (k_1, k_2, :, :). \end{aligned} \quad (2.54)$$

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

$$\begin{aligned} \Theta_1 = & \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_2 \leq N_2}} e^{\mathcal{A}_{34}^{k_1, k_2} \frac{\Delta t_n}{2}} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{24}^{k_1, k_3} \frac{\Delta t_n}{2}} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{23}^{k_1, k_4} \frac{\Delta t_n}{2}} \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{14}^{k_2, k_3} \frac{\Delta t_n}{2}} \left( \right. \right. \right. \right. \\ & \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{13}^{k_2, k_4} \frac{\Delta t_n}{2}} V_1^*(:, k_2, :, k_4) \right) (:, k_2, k_3, :) \right) (k_1, :, :, k_4) \right) (k_1, :, k_3, :) \right) (k_1, k_2, :, :), \end{aligned} \quad (2.55)$$

$$\begin{aligned} V_1^* = & \bigotimes_{\substack{1 \leq k_3 \leq N_3 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{12}^{k_3, k_4} \Delta t_n} \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{13}^{k_2, k_4} \frac{\Delta t_n}{2}} \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{14}^{k_2, k_3} \frac{\Delta t_n}{2}} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{23}^{k_1, k_4} \frac{\Delta t_n}{2}} \right. \right. \right. \\ & \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{24}^{k_1, k_3} \frac{\Delta t_n}{2}} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_2 \leq N_2}} e^{\mathcal{A}_{34}^{k_1, k_2} \frac{\Delta t_n}{2}} V_1(k_1, k_2, :, :) \right) (k_1, :, k_3, :) \right) (k_1, :, :, k_4) \right) \\ & \left. (:, k_2, k_3, :) \right) (:, k_2, :, k_4) \right) (:, :, k_3, k_4). \end{aligned} \quad (2.56)$$

And

$$\begin{aligned}
\Theta_2 = & \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_2 \leq N_2}} e^{\mathcal{A}_{34}^{k_1, k_2} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{24}^{k_1, k_3} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{23}^{k_1, k_4} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \right. \right. \\
& \left. \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{14}^{k_2, k_3} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{13}^{k_2, k_4} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} V_2^*(:, k_2, :, k_4) \right) \right. \right. \\
& \left. \left. (:, k_2, k_3, :) \right) (k_1, :, :, k_4) \right) (k_1, :, k_3, :) \right) (k_1, k_2, :, :),
\end{aligned} \tag{2.57}$$

$$\begin{aligned}
V_2^* = & \bigotimes_{\substack{1 \leq k_3 \leq N_3 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{12}^{k_3, k_4} (\Delta t_n + \Delta t_{n-1})} \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{13}^{k_2, k_4} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \left( \bigotimes_{\substack{1 \leq k_2 \leq N_2 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{14}^{k_2, k_3} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \right. \right. \\
& \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_4 \leq N_4}} e^{\mathcal{A}_{23}^{k_1, k_4} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_3 \leq N_3}} e^{\mathcal{A}_{24}^{k_1, k_3} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \left( \bigotimes_{\substack{1 \leq k_1 \leq N_1 \\ 1 \leq k_2 \leq N_2}} e^{\mathcal{A}_{34}^{k_1, k_2} \frac{(\Delta t_n + \Delta t_{n-1})}{2}} \right. \right. \\
& \left. \left. V_2(k_1, k_2, :, :) \right) (k_1, :, k_3, :) \right) (k_1, :, :, k_4) \right) (:, k_2, k_3, :) \right) (:, k_2, :, k_4) \right) (:, :, k_3, k_4).
\end{aligned} \tag{2.58}$$

### 2.3 Krylov IF method on sparse grids

To achieve further efficiency in solving the CDR equations (1.1) 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 [15, 30, 31].

Here we use two dimensional (2D) case as the example to illustrate the idea.

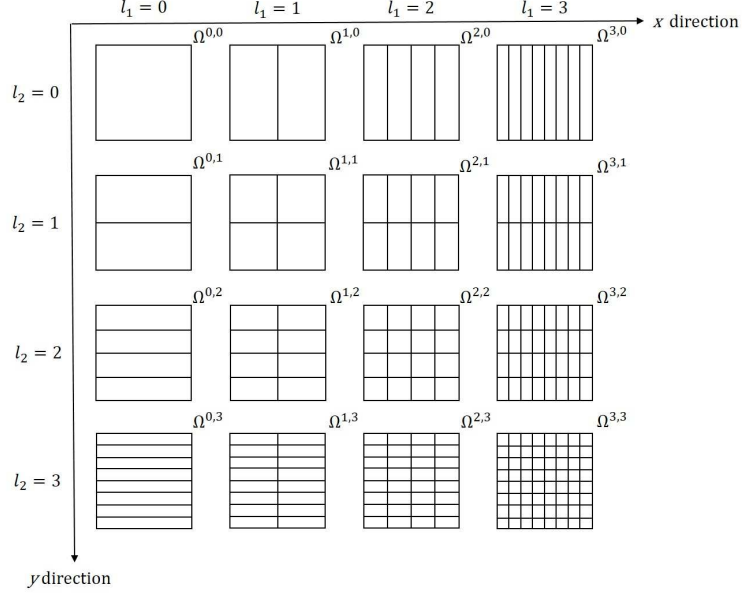


Figure 2.1. Semi-coarsened sparse grids  $\{\Omega^{l_1, l_2}\}$  with the finest level  $N_L = 3$ .

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  $l_1 = 0, 1, \dots, N_L$ ,  $l_2 = 0, 1, \dots, N_L$ , see figure 2.1. 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  $\Omega^{N_L, N_L}$  with mesh size  $h = 2^{-N_L}H$  for both  $x$  and  $y$  directions.

To solve equation (1.1), we will use the second order Krylov IIF (KrylovIIF2) method (2.16) or the third order Krylov IIF (KrylovIIF3) scheme (2.17) 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 (1.1) is solved on the following  $(2N_L + 1)$  sparse grids  $\{\Omega^{l_1, l_2}\}_I$ :

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

And  $I$  denotes the index set

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

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  $\Omega^{N_L, N_L}$ . The key point here is that the PDE is never solved directly on  $\Omega^{N_L, N_L}$  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 [15, 30, 31]) on each sparse grid solution  $U^{l_1, l_2}$  to obtain  $(2N_L + 1)$  solutions on the finest grid  $\Omega^{N_L, N_L}$ . And finally, these solutions are combined to form the final solution  $\hat{U}^{N_L, N_L}$  on  $\Omega^{N_L, N_L}$ .

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  $\Omega^{N_L, N_L}$ . And a prolongation operator is basically an interpolation operator. For example,  $U^{l_1, l_2}$  is numerical solution on  $\Omega^{l_1, l_2}$ , then  $P^{N_L, N_L} U^{l_1, l_2}$  gives numerical values on the most refined mesh  $\Omega^{N_L, N_L}$ . For the 2D case, first in one direction (e.g. the  $x$  direction), we construct  $(N_r 2^{l_1-1})$  quadratic interpolation polynomials  $P_i^2(x)$ ,  $i = 1, \dots, N_r 2^{l_1-1}$ , 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 [15, 30, 31]). Then we evaluate  $P_i^2(x)$  on the grid points of  $\Omega^{N_L, l_2}$ , 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  $P_j^2(y)$ ,  $j = 1, \dots, N_r 2^{l_2-1}$ , and evaluate them on the grid points of  $\Omega^{N_L, N_L}$ . At last we get  $P^{N_L, N_L} U^{l_1, l_2}$ , defined on the finest grid  $\Omega^{N_L, N_L}$ . 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  $\Omega^{l_1, l_2}$ , solve the equation (1.1) by KrylovIIF 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  $\Omega^{l_1, l_2}$ , apply prolongation operator  $P^{N_L, N_L}$  on  $U^{l_1, l_2}$ . We get  $P^{N_L, N_L} U^{l_1, l_2}$ , defined on the most refined mesh  $\Omega^{N_L, N_L}$ .
  - do the combination to get the final solution

$$\hat{U}^{N_L, N_L} = \sum_{l_1+l_2=N_L} P^{N_L, N_L} U^{l_1, l_2} - \sum_{l_1+l_2=N_L-1} P^{N_L, N_L} U^{l_1, l_2}. \quad (2.59)$$

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 [15]. Specifically the 3D formula is

$$\begin{aligned} \hat{U}^{N_L, N_L, N_L} = & \sum_{l_1+l_2+l_3=N_L} P^{N_L, N_L, N_L} U^{l_1, l_2, l_3} - 2 \sum_{l_1+l_2+l_3=N_L-1} P^{N_L, N_L, N_L} U^{l_1, l_2, l_3} \\ & + \sum_{l_1+l_2+l_3=N_L-2} P^{N_L, N_L, N_L} U^{l_1, l_2, l_3}. \end{aligned} \quad (2.60)$$

## 2.4 Linear stability analysis of the IIF2 scheme for CDR equations.

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

$$u_t = au - du + ru, \quad \text{with } r \in \mathcal{C}, \text{ and } a, d \in \mathcal{R}, d > 0. \quad (2.61)$$

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 [42], 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 (2.10) as an example. More details and analysis results can be found in [24].

Applying the IIF2 scheme (2.10) to the equation (2.61) with a uniform time step size  $\Delta t$ , then substituting  $u_n = e^{in\theta}$  into the resulting equation, we obtain

$$(1 - \frac{\lambda}{2})e^{2i\theta} = e^{-d\Delta t}(1 + \frac{\lambda}{2} + \frac{3}{2}a\Delta t)e^{i\theta} - \frac{a}{2}\Delta te^{-2d\Delta t}, \quad (2.62)$$

where  $\lambda = r\Delta t$  has a real part  $\lambda_r$  and imaginary part  $\lambda_i$ . Solve the equation (2.62) for  $\lambda_r$  and  $\lambda_i$  to have

$$\begin{cases} \lambda_r = \frac{B_1C_2 - B_2C_1}{A_1B_2 - A_2B_1}; \\ \lambda_i = \frac{A_1C_2 - A_2C_1}{A_2B_1 - A_1B_2}, \end{cases} \quad (2.63)$$



where

$$\begin{cases} A_1 = e^{-d\Delta t} \frac{1}{2} \cos \theta + \frac{1}{2} \cos 2\theta, \\ B_1 = -e^{-d\Delta t} \frac{1}{2} \sin \theta - \frac{1}{2} \sin 2\theta, \\ C_1 = -\frac{a}{2} \Delta t e^{-2d\Delta t} + e^{-d\Delta t} (1 + \frac{3}{2} a \Delta t) \cos \theta - \cos 2\theta, \\ A_2 = e^{-d\Delta t} \frac{1}{2} \sin \theta + \frac{1}{2} \sin 2\theta, \\ B_2 = e^{-d\Delta t} \frac{1}{2} \cos \theta + \frac{1}{2} \cos 2\theta, \\ C_2 = e^{-d\Delta t} (1 + \frac{3}{2} a \Delta t) \sin \theta - \sin 2\theta. \end{cases} \quad (2.64)$$

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 Figure 2.2. 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  $\theta$  from 0 to  $2\pi$  in (2.63) and (2.64). 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 (2.10) for some special values of  $d\Delta t$ ,  $a\Delta t$  and  $\lambda$ , 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 Figure 2.2. Then stable and unstable regions are determined and shown in Figure 2.2. From Figure 2.2, we can see that the whole regions outside of the stability boundary curves are stable regions, which shows that the IIF2 scheme (2.10) 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 Figure 2.3.  $d\Delta t = 1.0$ ,  $d\Delta t = 2.0$ ,  $d\Delta t = 10.0$  and  $d\Delta t = 20.0$  are chosen as examples. Again, analysis of the growth factor of the scheme (2.10) for some special values of  $d\Delta t$ ,  $a\Delta t$  and  $\lambda$ , 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 Figure 2.3. Stable regions for the cases shown in Figure 2.3 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  $a$ .

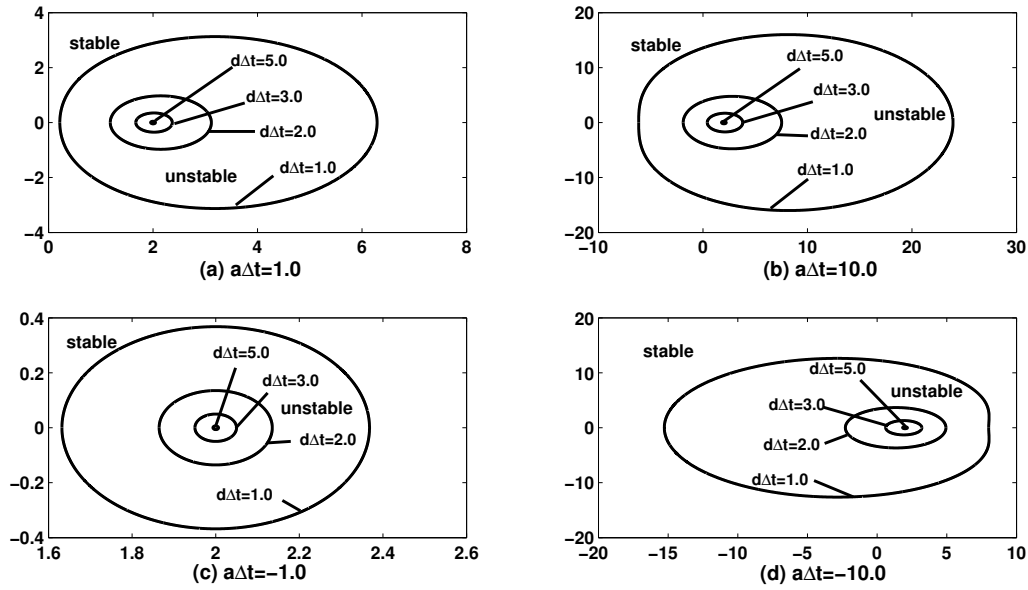


Figure 2.2. Linear stability regions of the IIF2 scheme (2.10) for different values of  $d\Delta t$  under a fixed value of  $a\Delta t$ . (a)  $a\Delta t = 1.0$ ; (b)  $a\Delta t = 10.0$ ; (c)  $a\Delta t = -1.0$ ; (d)  $a\Delta t = -10.0$ .

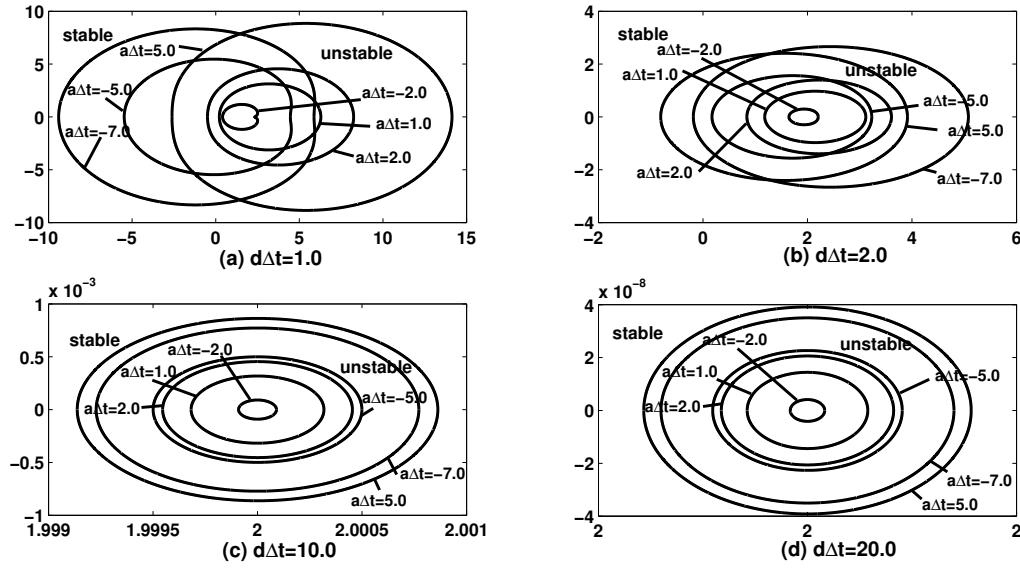


Figure 2.3. Linear stability regions of the IIF2 scheme (2.10) for different values of  $a\Delta t$  under a fixed value of  $d\Delta t$ . (a)  $d\Delta t = 1.0$ ; (b)  $d\Delta t = 2.0$ ; (c)  $d\Delta t = 10.0$ ; (d)  $d\Delta t = 20.0$ .

# CHAPTER 3

## NUMERICAL EXAMPLES FOR COMPUTATIONAL COMPLEXITY STUDY OF KRYLOV INTEGRATION FACTOR WENO METHOD

In this chapter, 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 dissertation can provide certain guidance in choosing the suitable numerical methods. All of the numerical simulations in this chapter 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 (2.28) or (2.38) (they are equivalent), and the second order Krylov IIF scheme (2.16) 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  $L^\infty$  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 dissertation, 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 Table 3.1 and Table 3.2 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 3.2 verify its linear computational complexity. On the other hand, although the cIIF2 scheme's CPU time ratios shown in Table 3.1 are not linear, the cIIF2 scheme is more efficient than the KrylovIIF2 scheme on  $40 \times 40$ ,  $80 \times 80$  and  $160 \times 160$  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  $640 \times 640$  and  $1280 \times 1280$ , the KrylovIIF2 scheme is more efficient than the cIIF2. On  $320 \times 320$  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  $\Delta t$  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 (2.26) with  $r = 3$ ) and the third order KrylovIIF scheme KrylovIIF3 (2.17) are used to compute the same two-dimensional problem until the final time  $T = 1$ . The comparison results are presented in Table 3.3 and Table 3.4. 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  $40 \times 40$ ,  $80 \times 80$ ,  $160 \times 160$  and  $320 \times 320$ . On very refined meshes  $640 \times 640$  and  $1280 \times 1280$ , KrylovIIF3 is more efficient.

Then we test the three dimensional case defined on the domain  $\Omega = \{0 \leq x \leq \pi, 0 \leq y \leq \pi, 0 \leq z \leq \pi\}$ , 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  $M = 25$ , 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 Table 3.5 and Table 3.6 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 3.6 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 to compute a few  $N \times N$  matrix exponentials before the time evolution and at the last time step.

TABLE 3.1

EXAMPLE 1. 2D CASE, CIIF2 SCHEME

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

TABLE 3.2

EXAMPLE 1. 2D CASE, KRYLOVIF2 SCHEME

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

TABLE 3.3

EXAMPLE 1. 2D CASE, CIIF3 SCHEME

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

TABLE 3.4

EXAMPLE 1. 2D CASE, KRYLOVIF3 SCHEME

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

TABLE 3.5

EXAMPLE 1. 3D CASE, CIIF2 SCHEME

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

TABLE 3.6

EXAMPLE 1. 3D CASE, KRYLOVIF2 SCHEME

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

**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  $L^\infty$  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 [42]. The results are reported in Table 3.7 and Table 3.8. 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  $640 \times 640$  and  $1280 \times 1280$ . The ratios of corresponding CPU times on an  $N \times N$  mesh to that on a  $\frac{N}{2} \times \frac{N}{2}$  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  $40 \times 40$ ,  $80 \times 80$ ,  $160 \times 160$  and  $320 \times 320$ . On more refined meshes  $640 \times 640$  and  $1280 \times 1280$ , the KrylovIIF2 scheme is more efficient than the cIIF2 scheme.



TABLE 3.7

EXAMPLE 2. CIIF2 SCHEME

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

TABLE 3.8

EXAMPLE 2. KRYLOVIF2 SCHEME

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

### 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

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

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 [53] for testing the AcIIF2 scheme. We compute the problem until the final time  $T = 1$  by the KrylovIIF2 scheme (2.16) with the convection term  $F_a = 0$ , and the AcIIF2 scheme (2.41), (2.50) 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 (2.50). 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 section 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  $L^\infty$  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 Table 3.9, Table 3.10, and Table 3.11. 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 3.10, 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 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 3.10 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 3.11. 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 3.9, 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 Table 3.9 and Table 3.11.

TABLE 3.9

EXAMPLE 3. KRYLOVIF2 SCHEME

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

TABLE 3.10

EXAMPLE 3. DIRECT ACIIF2 SCHEME

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

TABLE 3.11

EXAMPLE 3. ACIIF2 SCHEME WITH KRYLOV SUBSPACE  
APPROXIMATIONS

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

**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

$$\begin{aligned}
u_t = & (0.1u_{x_1x_1} - 0.15u_{x_1x_2} + 0.1u_{x_2x_2}) + (0.1u_{x_1x_1} + 0.2u_{x_1x_3} + 0.2u_{x_3x_3}) + \\
& (0.1u_{x_1x_1} + 0.2u_{x_1x_4} + 0.2u_{x_4x_4}) + (0.1u_{x_2x_2} + 0.2u_{x_2x_3} + 0.2u_{x_3x_3}) + \\
& (0.1u_{x_2x_2} + 0.2u_{x_2x_4} + 0.2u_{x_4x_4}) + (0.2u_{x_3x_3} + 0.15u_{x_3x_4} + 0.1u_{x_4x_4}) + 2u,
\end{aligned} \tag{3.1}$$

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

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

We compute the problem until the final time  $T = 1$  by the KrylovIIF2 scheme (2.16) with the convection term  $F_a = 0$ , and the AcIIF2 scheme (2.41), (2.53), (2.54) 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 Table 3.12, Table 3.13 and Table 3.14. 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 3.13. 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 3.14. Again, the KrylovIIF2 scheme is the most efficient one among all three approaches here on all meshes as shown in Table 3.12. 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 3.12

EXAMPLE 4. KRYLOVIF2 SCHEME

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

TABLE 3.13

EXAMPLE 4. DIRECT ACIIF2 SCHEME

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

TABLE 3.14

EXAMPLE 4. ACIIF2 SCHEME WITH KRYLOV SUBSPACE  
APPROXIMATIONS

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

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

$$\begin{aligned}
u_t = & 0.5u_{xx} - 0.5\sin(x+y)u_{xy} + 0.5u_{yy} \\
& + 0.5u_{xx} - \frac{1}{3}\cos y u_{xz} + \frac{1}{3}u_{zz} \\
& + 0.5(1 + \cos x)u_{yy} - 0.5(1 + \cos x)u_{yz} + \frac{1}{3}(1 + \cos x)u_{zz} + f(x, y, z, u),
\end{aligned} \tag{3.2}$$

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

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

This problem was used in [53] for testing the AcIIF2 scheme. We compute the problem until the final time  $T = 1$ . The KrylovIIF2 scheme (2.16) with the convection term  $F_a = 0$ , and the AcIIF2 scheme (2.41), (2.51), (2.52) 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 Table 3.15, Table 3.16 and Table 3.17. 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 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 (2.51) and (2.52) 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 3.16 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 3.17. And computations can be performed on much more refined meshes (Table 3.17) 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 3.15. 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 3.15.

TABLE 3.15

EXAMPLE 5. KRYLOVIF2 SCHEME

$N \times N \times N$	$L^\infty$ error	Order	CPU(s)	R1	CPU1(s)	R2	CPU2(s)	R3
$10 \times 10 \times 10$	$2.15 \times 10^{-1}$		0.17		0.17		0.03	
$20 \times 20 \times 20$	$5.29 \times 10^{-2}$	2.02	2.20	13.06	2.19	13.10	0.22	6.56
$40 \times 40 \times 40$	$1.34 \times 10^{-2}$	1.99	35.05	15.94	35.00	15.98	1.75	7.89
$80 \times 80 \times 80$	$3.34 \times 10^{-3}$	2.00	551.57	15.73	551.17	15.75	14.13	8.07
$160 \times 160 \times 160$	$8.34 \times 10^{-4}$	2.00	8,992.13	16.30	8,989.12	16.31	115.99	8.21
$320 \times 320 \times 320$	$2.09 \times 10^{-4}$	2.00	153,195.14	17.04	153,171.55	17.04	958.75	8.27

TABLE 3.16

EXAMPLE 5. DIRECT ACIIF2 SCHEME

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



TABLE 3.17

EXAMPLE 5. ACIIF2 SCHEME WITH KRYLOV SUBSPACE  
APPROXIMATIONS

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

### 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 [40]. 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 [40], a two-dimensional reaction diffusion model was developed [29] 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 [43]. 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 [29]:

$$\begin{aligned}
\frac{\partial [L]}{\partial t} &= D_L \left( \frac{\partial^2 [L]}{\partial x^2} + \frac{\partial^2 [L]}{\partial y^2} \right) - k_{on}[L] (R(x, y) - [LR]) + k_{off}[LR] \\
&\quad - j_{on}[L][S] + (j_{off} + \tau j_{deg})[LS] + V_L(x, y) \\
\frac{\partial [LR]}{\partial t} &= k_{on}[L] (R(x, y) - [LR]) - (k_{off} + k_{deg})[LR] \\
\frac{\partial [LS]}{\partial t} &= D_{LS} \left( \frac{\partial^2 [LS]}{\partial x^2} + \frac{\partial^2 [LS]}{\partial y^2} \right) + j_{on}[L][S] - (j_{off} + j_{deg})[LS] \\
\frac{\partial [S]}{\partial t} &= D_S \left( \frac{\partial^2 [S]}{\partial x^2} + \frac{\partial^2 [S]}{\partial y^2} \right) - j_{on}[L][S] + j_{off}[LS] + V_S(x, y)
\end{aligned} \tag{3.3}$$

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

$$R(x, y) = \begin{cases} R_h, & x \leq X_h, \\ R_0, & x > X_h. \end{cases} \quad (3.4)$$

$$V_L(X, Y) = \begin{cases} v_L, & y < \frac{1}{2}Y_{\max}, \\ 0, & y \geq \frac{1}{2}Y_{\max}. \end{cases} \quad (3.5)$$

$$V_S(X, Y) = \begin{cases} 0, & y < \frac{1}{2}Y_{\max}, \\ v_S, & y \geq \frac{1}{2}Y_{\max}. \end{cases} \quad (3.6)$$

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;  $\tau$  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 [40]. We study the cell receptor over-expression experiments in [40] by setting  $R_h = 9\mu M$  in the region  $0 < x \leq X_h = 0.02\text{cm}$ , 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 Figure 3.1 and Figure 3.2. Similar results are obtained for these two methods. Simulations by both methods confirm that the over-expression of receptor induces a local boost of Dppreceptor activities near the boundary of two different concentration regions of receptors, similar to the

experimental observations in [40]. However 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

$$\begin{aligned}
u_t + \left(\frac{1}{2}u^2\right)_{x_1} + \left(\frac{1}{2}u^2\right)_{x_2} + \left(\frac{1}{2}u^2\right)_{x_3} + \left(\frac{1}{2}u^2\right)_{x_4} = \\
(0.1u_{x_1x_1} - 0.15u_{x_1x_2} + 0.1u_{x_2x_2}) + (0.1u_{x_1x_1} + 0.2u_{x_1x_3} + 0.2u_{x_3x_3}) + \\
(0.1u_{x_1x_1} + 0.2u_{x_1x_4} + 0.2u_{x_4x_4}) + (0.1u_{x_2x_2} + 0.2u_{x_2x_3} + 0.2u_{x_3x_3}) + \\
(0.1u_{x_2x_2} + 0.2u_{x_2x_4} + 0.2u_{x_4x_4}) + (0.2u_{x_3x_3} + 0.15u_{x_3x_4} + 0.1u_{x_4x_4}) + S(x_1, x_2, x_3, x_4, t),
\end{aligned} \tag{3.7}$$

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

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

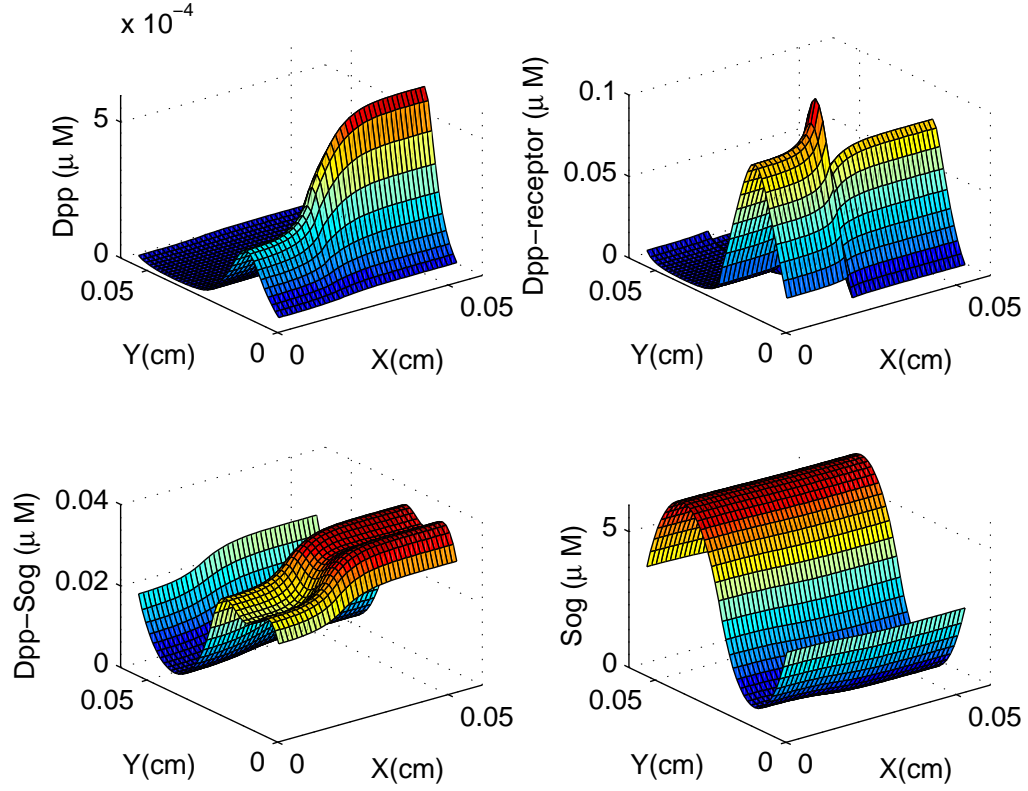


Figure 3.1. Numerical solutions of Example 6 using the Krylov IIF2 scheme: concentrations of  $[L]$ ,  $[LR]$ ,  $[LS]$ ,  $[S]$  at  $T = 100$  seconds for the

Dpp-Sog system when receptors are over-expressed.

$\Delta t = h_x = h_y = 0.001375$  in the simulation. Parameters are  $D_L = D_{LS} = D_S = 85 \mu m^2 s^{-1}$ ;  $v_L = 1 nM s^{-1}$ ;  $v_S = 80 nM s^{-1}$ ;  $k_{on} = 0.4 \mu M^{-1} s^{-1}$ ;  $k_{off} = 4 \times 10^{-6} s^{-1}$ ;  $k_{deg} = 5 \times 10^{-4} s^{-1}$ ;  $j_{on} = 95 \mu M^{-1} s^{-1}$ ;  $j_{off} = 4 \times 10^{-6} s^{-1}$ ;  $j_{deg} = 0.54 s^{-1}$ ;  $\tau = 1$ ;  $R_h = 9 \mu M$ ;  $R_0 = 3 \mu M$ .

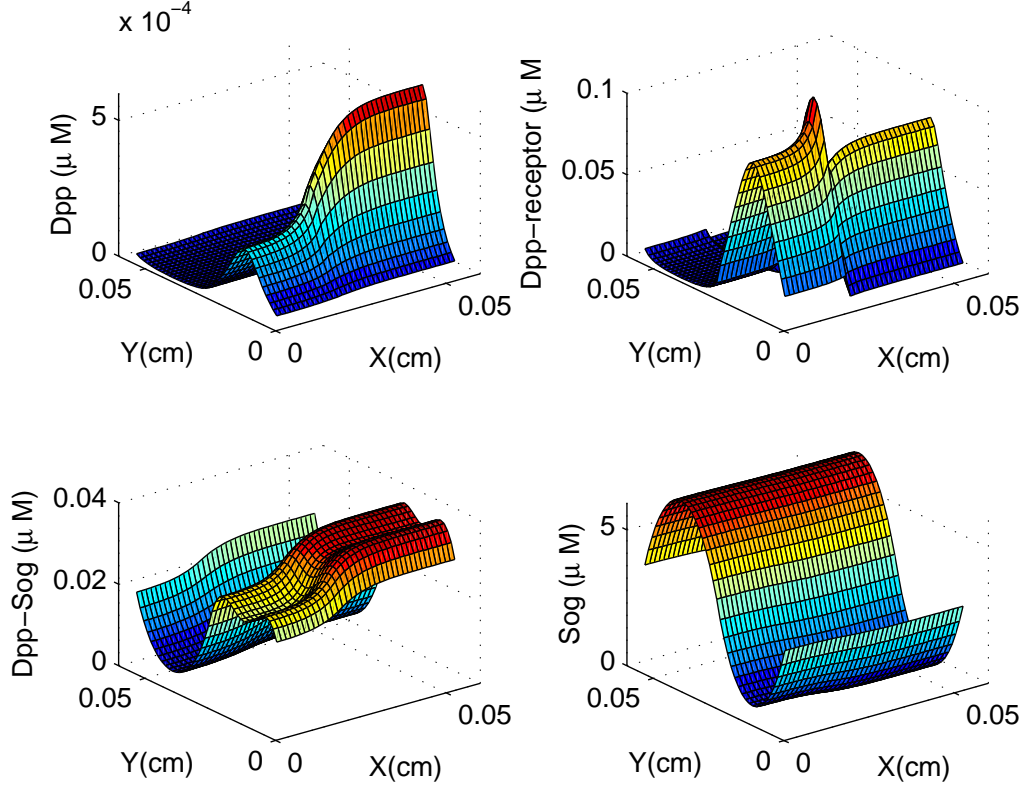


Figure 3.2. Numerical solutions of Example 6 using the cIIF2 scheme: concentrations of  $[L]$ ,  $[LR]$ ,  $[LS]$ ,  $[S]$  at  $T = 100$  seconds for the Dpp-Sog system when receptors are over-expressed.  $\Delta t = h_x = h_y = 0.001375$  in the simulation. Parameters are  $D_L = D_{LS} = D_S = 85\mu m^2 s^{-1}$ ;  $v_L = 1nM s^{-1}$ ;  $v_S = 80nM s^{-1}$ ;  $k_{on} = 0.4\mu M^{-1} s^{-1}$ ;  $k_{off} = 4 \times 10^{-6} s^{-1}$ ;  $k_{deg} = 5 \times 10^{-4} s^{-1}$ ;  $j_{on} = 95\mu M^{-1} s^{-1}$ ;  $j_{off} = 4 \times 10^{-6} s^{-1}$ ;  $j_{deg} = 0.54 s^{-1}$ ;  $\tau = 1$ ;  $R_h = 9\mu M$ ;  $R_0 = 3\mu M$ .

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 (2.16) and the AcIIF2-WENO scheme (2.41), (2.53), (2.54) with Krylov subspace approximations to matrix exponentials in (2.53) and (2.54) 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 Table 3.18 and Table 3.19. 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.

TABLE 3.18

EXAMPLE 7. KRYLOVIF2-WENO SCHEME

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



TABLE 3.19

EXAMPLE 7. ACIIF2-WENO SCHEME WITH KRYLOV SUBSPACE  
APPROXIMATIONS

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

**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

$$\begin{aligned}
& u_t + \left(\frac{1}{2}u^2\right)_x + \left(\frac{1}{2}u^2\right)_y + \left(\frac{1}{2}u^2\right)_z = \\
& 0.5u_{xx} - 0.5\sin(x+y)u_{xy} + 0.5u_{yy} + 0.5u_{xx} - \frac{1}{3}\cos(y)u_{xz} + \frac{1}{3}u_{zz} \\
& + 0.5(1+\cos x)u_{yy} - 0.5(1+\cos x)u_{yz} + \frac{1}{3}(1+\cos x)u_{zz} + S(x, y, z, t),
\end{aligned} \tag{3.8}$$

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

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

We compute the problem until the final time  $T = 1$ . The KrylovIIF2-WENO scheme (2.16), and the AcIIF2-WENO scheme (2.41), (2.51), (2.52) with Krylov subspace approximations to matrix exponentials in (2.51) and (2.52) 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 Table 3.20 and Table 3.21. 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.

TABLE 3.20

EXAMPLE 8. KRYLOVIIF2-WENO SCHEME

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

TABLE 3.21

EXAMPLE 8. ACIIF2-WENO SCHEME WITH KRYLOV SUBSPACE  
APPROXIMATIONS

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

**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 Burgers' equation

$$\begin{cases} u_t + (\frac{u^2}{2})_x + (\frac{u^2}{2})_y = 0.01\Delta u, & -2 \leq x \leq 2, \quad -2 \leq y \leq 2, \\ u(x, y, 0) = 0.3 + 0.7 \sin(\frac{\pi}{2}(x + y)), \end{cases} \quad (3.9)$$

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 Figure 3.3. 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 3.22. 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.

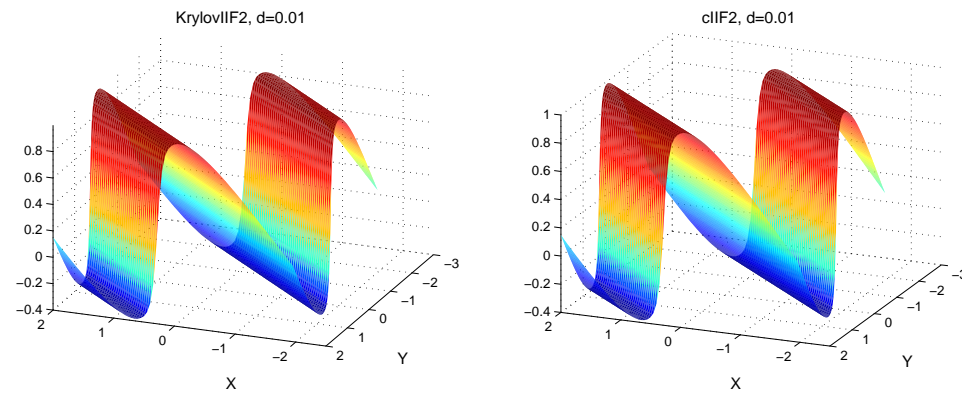


Figure 3.3. Numerical solutions of nonlinear viscous Burgers' equation on a  $80 \times 80$  mesh by the Krylov IIF2-WENO scheme and the cIIF2-WENO scheme. Time  $T = 5/\pi^2$ . Left picture: result of Krylov IIF2-WENO; right picture: result of cIIF2-WENO.

TABLE 3.22

CPU TIME COMPARISONS FOR SOLVING THE NONLINEAR  
VISCOUS BURGERS' EQUATION

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

**Example 10 (Fokker-Planck equations).** The Fokker-Planck equation (FPE) [44] 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 [53], AcIIF schemes with second order 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 [44]. The general form of FPEs is

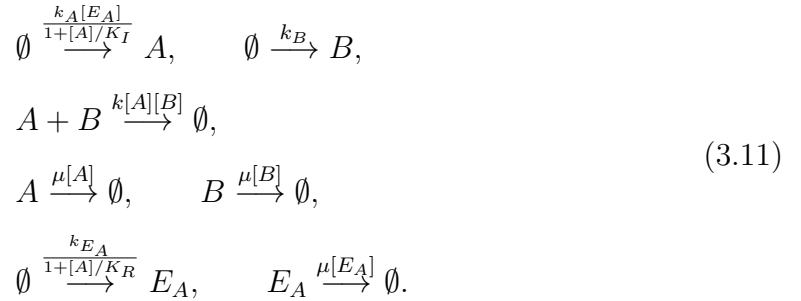
$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = - \sum_{r=1}^R \left\{ \sum_{i=1}^N n_{ri} \frac{\partial}{\partial x_i} \left( q_r(\mathbf{x}, t) - \frac{1}{2} \sum_{j=1}^N n_{rj} \frac{\partial q_r(\mathbf{x}, t)}{\partial x_j} \right) \right\}, \quad (3.10)$$

where  $p(\mathbf{x}, t)$  is the probability density of the system at the state  $\mathbf{x} = (x_1, x_2, \dots, x_N)$  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.  $q_r(\mathbf{x}, t)$  is defined by  $q_r(\mathbf{x}, t) = w_r(\mathbf{x})p(\mathbf{x}, t)$ , where  $w_r(\mathbf{x}, t)$  is the reaction propensity function for  $r$ -th reaction at state  $\mathbf{x}$  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 dissertation 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 (2.16) and the AcIIF2-WENO scheme (2.41), (2.51), (2.52) with Krylov subspace approximations for a three dimensional Fokker-Planck equation [48] 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):



In this system, the total number of reactions  $R$  is 7, and the total number of chemical species  $N$  is 3. The vectors  $\mathbf{n}_r = (n_{r1}, n_{r2}, n_{r3})$  are  $\mathbf{n}_1 = (1, 0, 0)$ ,  $\mathbf{n}_2 = (0, 1, 0)$ ,  $\mathbf{n}_3 = (-1, -1, 0)$ ,  $\mathbf{n}_4 = (-1, 0, 0)$ ,  $\mathbf{n}_5 = (0, -1, 0)$ ,  $\mathbf{n}_6 = (0, 0, 1)$ ,  $\mathbf{n}_7 = (0, 0, -1)$ . We denote the system state  $\mathbf{x}$  by  $\mathbf{x} = (x_1, x_2, x_3)$  which is  $([A], [B], [E_A])$



in this case. Then the propensity functions  $w_r(\mathbf{x})$  are

$$\begin{aligned} w_1 &= \frac{k_A x_3}{1 + x_1/K_I}, & w_2 &= k_B, & w_3 &= k x_1 x_2, \\ w_4 &= \mu x_1, & w_5 &= \mu x_2, & w_6 &= \frac{k_{E_A}}{1 + x_1/K_R}, & w_7 &= \mu x_3, \end{aligned} \quad (3.12)$$

where  $k_A = 0.3s^{-1}$ ,  $k_B = 2s^{-1}$ ,  $K_I = 30$ ,  $k = 0.001s^{-1}$ ,  $\mu = 0.004s^{-1}$ ,  $K_R = 30$  and  $k_{E_A} = 1s^{-1}$  [48]. Then the FPE can be written as

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -(L_1 + L_2 + L_3 + L_4 + L_5 + L_6 + L_7), \quad (3.13)$$

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

$$\begin{aligned} L_1 &= \frac{\partial q_1(\mathbf{x}, t)}{\partial x_1} - \frac{1}{2} \frac{\partial^2 q_1(\mathbf{x}, t)}{\partial x_1^2}, \\ L_2 &= \frac{\partial q_2(\mathbf{x}, t)}{\partial x_2} - \frac{1}{2} \frac{\partial^2 q_2(\mathbf{x}, t)}{\partial x_2^2}, \\ L_3 &= -\frac{\partial q_3(\mathbf{x}, t)}{\partial x_1} - \frac{\partial q_3(\mathbf{x}, t)}{\partial x_2} - \frac{1}{2} \left( \frac{\partial^2 q_3(\mathbf{x}, t)}{\partial x_1^2} + \frac{\partial^2 q_3(\mathbf{x}, t)}{\partial x_2^2} + 2 \frac{\partial^2 q_3(\mathbf{x}, t)}{\partial x_1 \partial x_2} \right), \\ L_4 &= -\frac{\partial q_4(\mathbf{x}, t)}{\partial x_1} - \frac{1}{2} \frac{\partial^2 q_4(\mathbf{x}, t)}{\partial x_1^2}, \\ L_5 &= -\frac{\partial q_5(\mathbf{x}, t)}{\partial x_2} - \frac{1}{2} \frac{\partial^2 q_5(\mathbf{x}, t)}{\partial x_2^2}, \\ L_6 &= \frac{\partial q_6(\mathbf{x}, t)}{\partial x_3} - \frac{1}{2} \frac{\partial^2 q_6(\mathbf{x}, t)}{\partial x_3^2}, \\ L_7 &= -\frac{\partial q_7(\mathbf{x}, t)}{\partial x_3} - \frac{1}{2} \frac{\partial^2 q_7(\mathbf{x}, t)}{\partial x_3^2}. \end{aligned} \quad (3.14)$$

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 convec-

tion 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  $\Delta t$  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 3.23, 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  $\Delta t$  are obtained by calculating the difference of numerical values for  $\Delta t$  and  $\Delta t/2$ . 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  $\Delta t = 0.017$ , in Table 3.24. The CPU times in Table 3.24 show that the KrylovIIF2-WENO scheme is more efficient than the AcIIF2-WENO scheme with Krylov subspace approximations, for this example. In Figures 3.4, 3.5, and 3.6, 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 Figures 3.7, 3.8, and 3.9. We see that both methods generate similar numerical solutions.

TABLE 3.23

NUMERICAL ERRORS AND ACCURACY ORDERS FOR THE  
 KRYLOVIIF2 SCHEME AND THE ACIIF2 SCHEME WITH KRYLOV  
 SUBSPACE APPROXIMATIONS FOR THE 3D FOKKER-PLANCK  
 EQUATION (4.5)

KrylovIIF2-WENO			
time step size	$L^\infty$ error	Order	
$\Delta t$	$1.56 \times 10^{-8}$		
$\Delta t/2$	$3.90 \times 10^{-9}$	2.00	
$\Delta t/4$	$1.00 \times 10^{-9}$	1.96	
AcIIF2-WENO with Krylov subspace approx.			
time step	$L^\infty$ error	Order	
$\Delta t$	$1.56 \times 10^{-8}$		
$\Delta t/2$	$3.90 \times 10^{-9}$	2.00	
$\Delta t/4$	$1.00 \times 10^{-9}$	1.96	

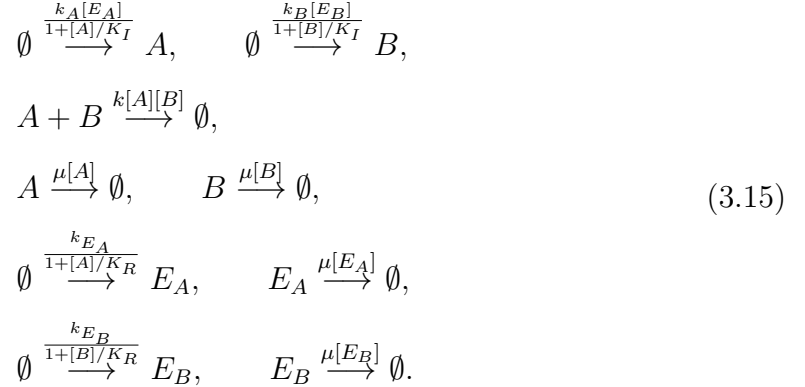
TABLE 3.24

CPU TIME FOR KRYLOVIF2 SCHEME AND THE ACIIF2 SCHEME  
WITH KRYLOV SUBSPACE APPROXIMATIONS FOR THE 3D  
FOKKER-PLANCK EQUATION (4.5)

	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

**(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):



In this system, the total number of reactions  $R$  is 9, and the total number of chemical species  $N$  is 4. The vectors  $\mathbf{n}_r = (n_{r1}, n_{r2}, n_{r3}, n_{r4})$  are  $\mathbf{n}_1 = (1, 0, 0, 0)$ ,  $\mathbf{n}_2 = (0, 1, 0, 0)$ ,  $\mathbf{n}_3 = (-1, -1, 0, 0)$ ,  $\mathbf{n}_4 = (-1, 0, 0, 0)$ ,  $\mathbf{n}_5 = (0, -1, 0, 0)$ ,  $\mathbf{n}_6 = (0, 0, 1, 0)$ ,  $\mathbf{n}_7 = (0, 0, -1, 0)$ ,  $\mathbf{n}_8 = (0, 0, 0, 1)$ ,  $\mathbf{n}_9 = (0, 0, 0, -1)$ . We denote the

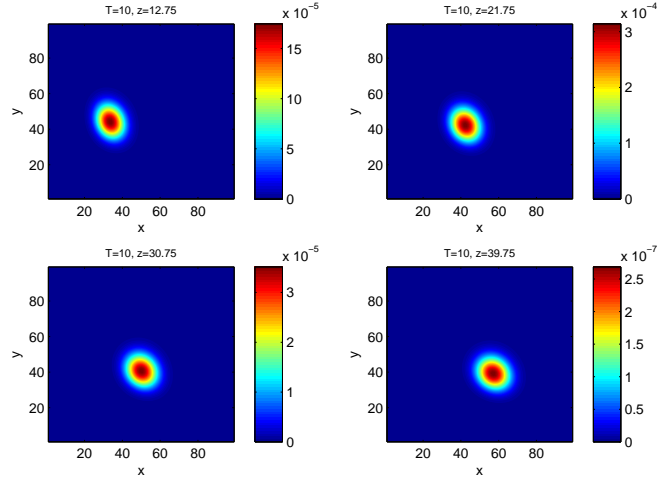


Figure 3.4. Distribution of A and B with  $E_A = 12.75, 21.75, 30.75, 39.75$ . Numerical solutions of (4.5) using the KrylovIIF2-WENO scheme. Final time  $T = 10$ .  $\Delta t = 0.017$ . The numbers of spatial grid points are  $N_A = 120, N_B = 120, N_{E_A} = 60$ .

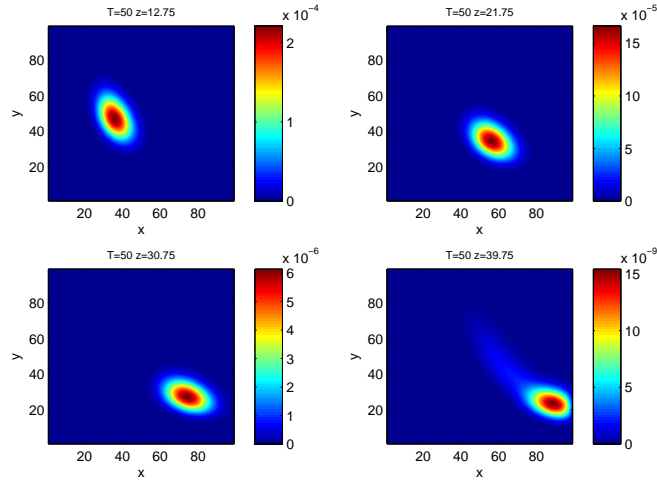


Figure 3.5. Distribution of A and B with  $E_A = 12.75, 21.75, 30.75, 39.75$ . Numerical solutions of (4.5) using the KrylovIIF2-WENO scheme. Final time  $T = 50$ .  $\Delta t = 0.017$ . The numbers of spatial grid points are  $N_A = 120, N_B = 120, N_{E_A} = 60$ .

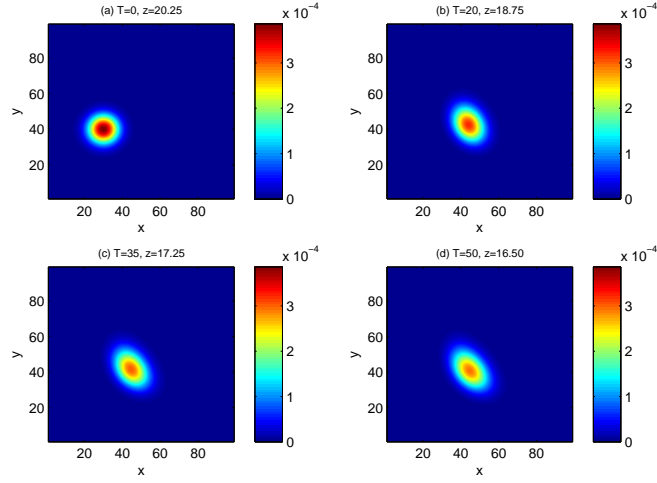


Figure 3.6. Distribution of A and B with different  $E_A$  values, at time  $T = 0, 20, 35, 50$ . Numerical solutions of (4.5) using the KrylovIIF2-WENO scheme.  $\Delta t = 0.017$ . The numbers of spatial grid points are  $N_A = 120$ ,  $N_B = 120$ ,  $N_{E_A} = 60$ .

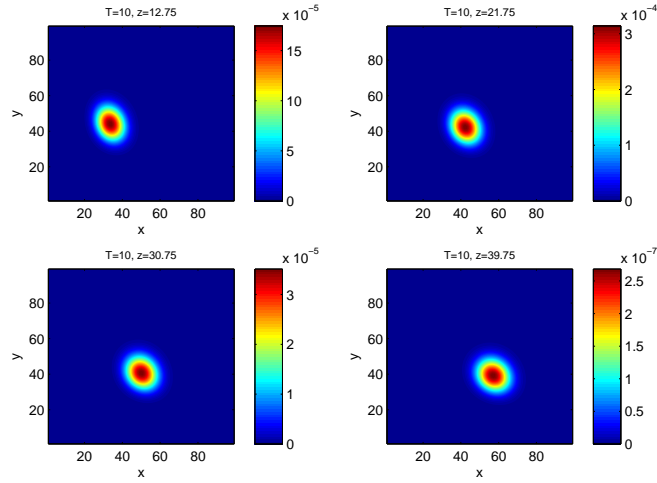


Figure 3.7. Distribution of A and B with  $E_A = 12.75, 21.75, 30.75, 39.75$ . Numerical solutions of (4.5) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time  $T = 10$ .  $\Delta t = 0.017$ . The numbers of spatial grid points are  $N_A = 120$ ,  $N_B = 120$ ,  $N_{E_A} = 60$ .

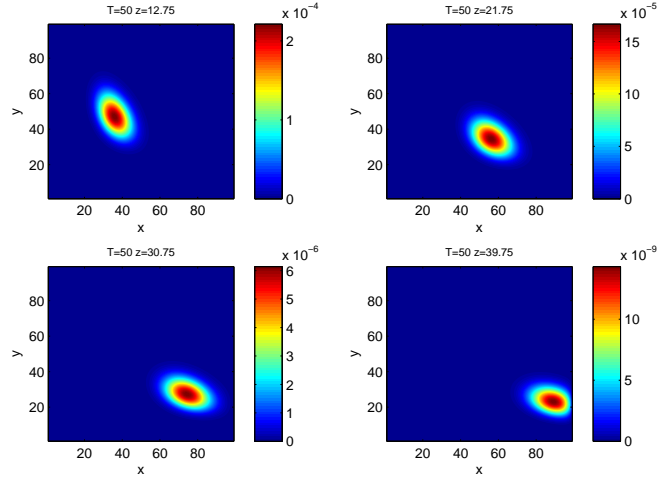


Figure 3.8. Distribution of A and B with  $E_A = 12.75, 21.75, 30.75, 39.75$ . Numerical solutions of (4.5) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time  $T = 50$ .  $\Delta t = 0.017$ . The numbers of spatial grid points are  $N_A = 120$ ,  $N_B = 120$ ,  $N_{E_A} = 60$ .

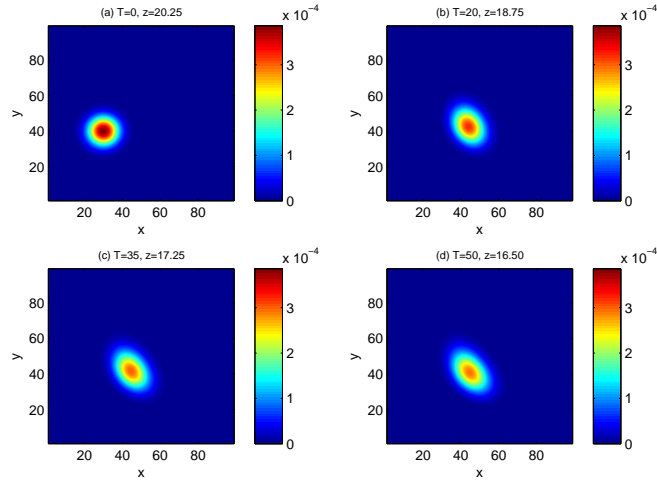


Figure 3.9. Distribution of A and B with different  $E_A$  values, at time  $T = 0, 20, 35, 50$ . Numerical solutions of (4.5) using the AcIIF2-WENO scheme with Krylov subspace approximations.  $\Delta t = 0.017$ . The numbers of spatial grid points are  $N_A = 120$ ,  $N_B = 120$ ,  $N_{E_A} = 60$ .

system state  $\mathbf{x}$  by  $\mathbf{x} = (x_1, x_2, x_3, x_4)$  which is  $([A], [B], [E_A], [E_B])$  in this case. Then the propensity functions  $w_r(\mathbf{x})$  are

$$\begin{aligned} w_1 &= \frac{k_A x_3}{1 + x_1/K_I}, & w_2 &= \frac{k_B x_4}{1 + x_2/K_I}, & w_3 &= k x_1 x_2, & w_4 &= \mu x_1, \\ w_5 &= \mu x_2, & w_6 &= \frac{k_{E_A}}{1 + x_1/K_R}, & w_7 &= \mu x_3, & w_8 &= \frac{k_{E_B}}{1 + x_2/K_R}, & w_9 &= \mu x_4, \end{aligned} \quad (3.16)$$

where  $k_A = 0.3s^{-1}$ ,  $k_B = 0.3s^{-1}$ ,  $K_I = 60$ ,  $k = 0.001s^{-1}$ ,  $\mu = 0.002s^{-1}$ ,  $K_R = 30$ ,  $k_{E_A} = 0.02s^{-1}$  and  $k_{E_B} = 0.02s^{-1}$ [48]. Then the FPE can be written as

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -(L_1 + L_2 + L_3 + L_4 + L_5 + L_6 + L_7 + L_8 + L_9), \quad (3.17)$$

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

$$\begin{aligned} L_1 &= \frac{\partial q_1(\mathbf{x}, t)}{\partial x_1} - \frac{1}{2} \frac{\partial^2 q_1(\mathbf{x}, t)}{\partial x_1^2}, \\ L_2 &= \frac{\partial q_2(\mathbf{x}, t)}{\partial x_2} - \frac{1}{2} \frac{\partial^2 q_2(\mathbf{x}, t)}{\partial x_2^2}, \\ L_3 &= -\frac{\partial q_3(\mathbf{x}, t)}{\partial x_1} - \frac{\partial q_3(\mathbf{x}, t)}{\partial x_2} - \frac{1}{2} \left( \frac{\partial^2 q_3(\mathbf{x}, t)}{\partial x_1^2} + \frac{\partial^2 q_3(\mathbf{x}, t)}{\partial x_2^2} + 2 \frac{\partial^2 q_3(\mathbf{x}, t)}{\partial x_1 \partial x_2} \right), \\ L_4 &= -\frac{\partial q_4(\mathbf{x}, t)}{\partial x_1} - \frac{1}{2} \frac{\partial^2 q_4(\mathbf{x}, t)}{\partial x_1^2}, \\ L_5 &= -\frac{\partial q_5(\mathbf{x}, t)}{\partial x_2} - \frac{1}{2} \frac{\partial^2 q_5(\mathbf{x}, t)}{\partial x_2^2}, \\ L_6 &= \frac{\partial q_6(\mathbf{x}, t)}{\partial x_3} - \frac{1}{2} \frac{\partial^2 q_6(\mathbf{x}, t)}{\partial x_3^2}, \\ L_7 &= -\frac{\partial q_7(\mathbf{x}, t)}{\partial x_3} - \frac{1}{2} \frac{\partial^2 q_7(\mathbf{x}, t)}{\partial x_3^2}, \\ L_8 &= \frac{\partial q_8(\mathbf{x}, t)}{\partial x_4} - \frac{1}{2} \frac{\partial^2 q_8(\mathbf{x}, t)}{\partial x_4^2}, \\ L_9 &= -\frac{\partial q_9(\mathbf{x}, t)}{\partial x_4} - \frac{1}{2} \frac{\partial^2 q_9(\mathbf{x}, t)}{\partial x_4^2}. \end{aligned} \quad (3.18)$$

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 for the diffusion terms. We compare the computational efficiency of the second order Krylov IIF scheme (2.16) and the second order AcIIF scheme (2.41), (2.55)-(2.58) with Krylov subspace approximations. For simulation results shown in the figures here, the time step size  $\Delta t$  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 3.25, 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  $\Delta t = 0.1$ , in Table 3.26. We obtain the same conclusion as that for the three dimensional problem. The CPU times in Table 3.26 show that the KrylovIIF2-WENO scheme is more efficient than the AcIIF2-WENO scheme with Krylov subspace approximations, for this four dimensional example. In Figures 3.10, 3.11, and 3.12, 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 Figures 3.13, 3.14, and 3.15. We see that both methods generate similar numerical solutions.

TABLE 3.25

NUMERICAL ERRORS AND ACCURACY ORDERS FOR THE  
 KRYLOVIIF2 SCHEME AND THE ACIIF2 SCHEME WITH KRYLOV  
 SUBSPACE APPROXIMATIONS FOR THE 4D FOKKER-PLANCK  
 EQUATION (3.17)

KrylovIIF2-WENO			
time step size	$L^\infty$ error	Order	
$\Delta t$	$1.03 \times 10^{-8}$		
$\Delta t/2$	$2.58 \times 10^{-9}$	2.00	
$\Delta t/4$	$6.47 \times 10^{-10}$	2.00	
AcIIF2-WENO with Krylov subspace approx.			
time step	$L^\infty$ error	Order	
$\Delta t$	$1.03 \times 10^{-8}$		
$\Delta t/2$	$2.58 \times 10^{-9}$	2.00	
$\Delta t/4$	$6.47 \times 10^{-10}$	2.00	

TABLE 3.26

CPU TIME FOR KRYLOVIIF2 SCHEME AND THE ACIIF2 SCHEME  
WITH KRYLOV SUBSPACE APPROXIMATIONS FOR THE 4D  
FOKKER-PLANCK EQUATION (3.17)

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

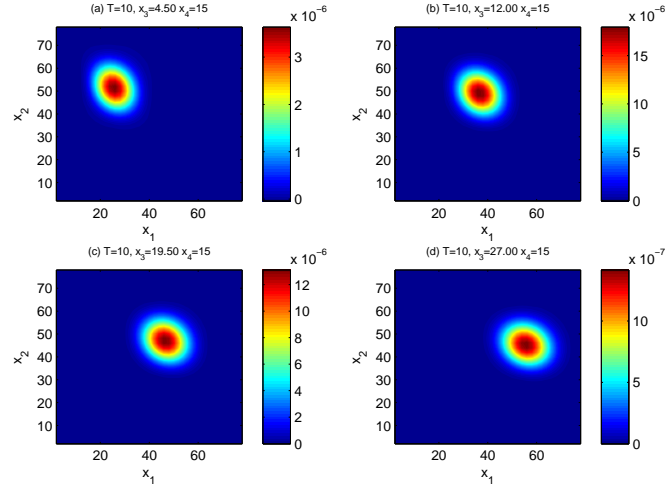


Figure 3.10. Distribution of A and B with  $E_A = 4.5, 12, 19.5, 27$  and  $E_B = 15$ . Numerical solutions of (3.17) using the KrylovIIF2-WENO scheme. Final time  $T = 10$ .  $\Delta t = 0.1$ . The numbers of spatial grid points are  $N_A = 40$ ,  $N_B = 40$ ,  $N_{E_A} = 20$ ,  $N_{E_B} = 20$ .

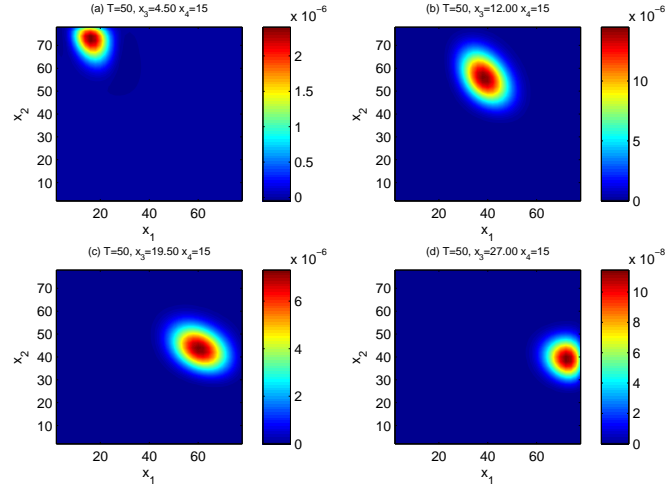


Figure 3.11. Distribution of A and B with  $E_A = 4.5, 12, 19.5, 27$  and  $E_B = 15$ . Numerical solutions of (3.17) using the KrylovIIF2-WENO scheme. Final time  $T = 50$ .  $\Delta t = 0.1$ . The numbers of spatial grid points are  $N_A = 40$ ,  $N_B = 40$ ,  $N_{E_A} = 20$ ,  $N_{E_B} = 20$ .

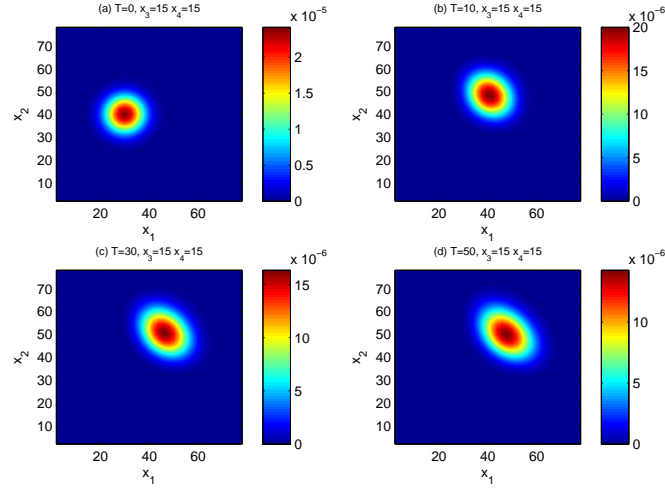


Figure 3.12. Distribution of A and B with  $E_A = 15$  and  $E_B = 15$ , at time  $T = 0, 10, 30, 50$ . Numerical solutions of (3.17) using the KrylovIIF2-WENO scheme.  $\Delta t = 0.1$ . The numbers of spatial grid points are  $N_A = 40$ ,  $N_B = 40$ ,  $N_{E_A} = 20$ ,  $N_{E_B} = 20$ .

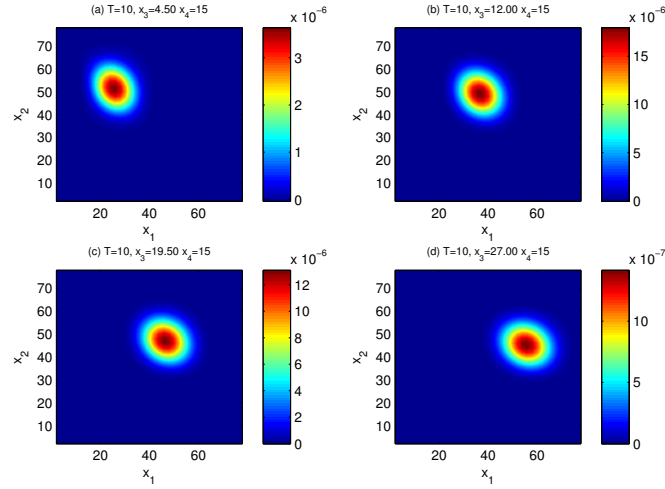


Figure 3.13. Distribution of A and B with  $E_A = 4.5, 12, 19.5, 27$  and  $E_B = 15$ . Numerical solutions of (3.17) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time  $T = 10$ .  $\Delta t = 0.1$ . The numbers of spatial grid points are  $N_A = 40$ ,  $N_B = 40$ ,  $N_{E_A} = 20$ ,  $N_{E_B} = 20$ .

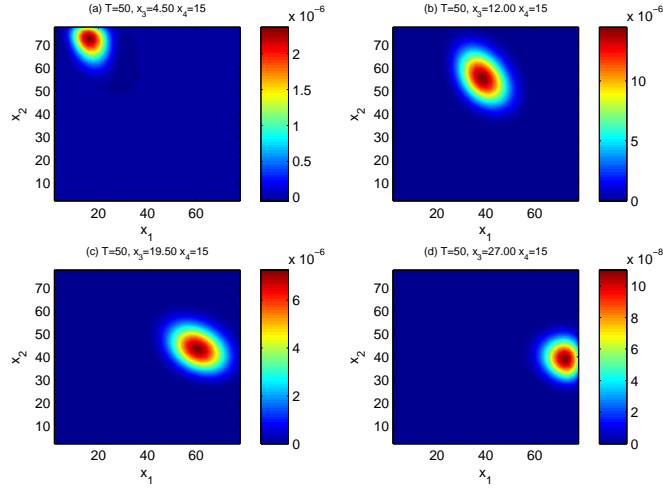


Figure 3.14. Distribution of A and B with  $E_A = 4.5, 12, 19.5, 27$  and  $E_B = 15$ . Numerical solutions of (3.17) using the AcIIF2-WENO scheme with Krylov subspace approximations. Final time  $T = 50$ .  $\Delta t = 0.1$ . The numbers of spatial grid points are  $N_A = 40$ ,  $N_B = 40$ ,  $N_{E_A} = 20$ ,  $N_{E_B} = 20$ .

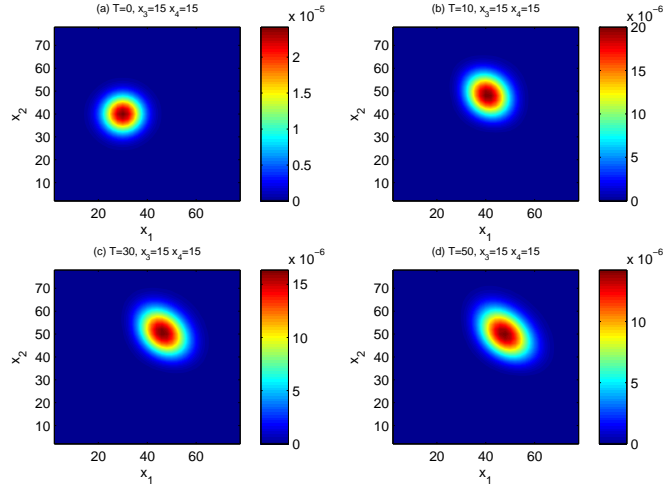


Figure 3.15. Distribution of A and B with  $E_A = 15$  and  $E_B = 15$ , at time  $T = 0, 10, 30, 50$ . Numerical solutions of (3.17) using the AcIIF2-WENO scheme with Krylov subspace approximations.  $\Delta t = 0.1$ . The numbers of spatial grid points are  $N_A = 40$ ,  $N_B = 40$ ,  $N_{E_A} = 20$ ,  $N_{E_B} = 20$ .

## CHAPTER 4

### NUMERICAL EXAMPLES FOR KRYLOV INTEGRATION METHODS ON SPARSE GRIDS

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  $\Omega^{l_1, l_2}$ , i.e.  $\Delta t = c \times \min(h_{l_1}, h_{l_2})$ .  $c$  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 Example 2, Example 3, Example 4 and Example 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  $\Omega^{N_L, N_L}$ , namely, it is proportional to  $h$  with  $\Delta t = c \times h$ . 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 dissertation 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  $\Omega^{N_L, N_L}$ . All of the numerical simulations in this chapter are performed on a 2.3 GHz, 16GB RAM Linux workstation.

#### 4.1 Isotropic diffusion problems

**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 (KrylovIIF2) (2.16) on both single grids and sparse grids, and compare their computational efficiency. We present the  $L^\infty$  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  $10 \times 10$  root grid and  $N_L = 3$  has the finest mesh  $80 \times 80$ . If the root grid is refined once to be  $20 \times 20$ , with  $N_L = 3$  unchanged we can obtain the finest mesh  $160 \times 160$ . 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  $10 \times 10$  root grid, the finest mesh which is  $80 \times 80$  for the  $N_L = 3$  case is refined to be  $160 \times 160$  for the  $N_L = 4$  case. The numerical errors, accuracy orders, and CPU times are listed in Table 4.1 for computations by the KrylovIIF2 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 4.1, 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 4.1, 92% CPU time can be saved for the computation on a  $640 \times 640$  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  $\Delta t = O(h)$  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 4.1 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 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 (KrylovIIF3) (2.17) 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 4.2. 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 4.2. Especially for the the second mesh refinement method (i.e., refine level), 82% CPU time can be saved for the computation on a  $640 \times 640$  mesh. In terms of computational complexity, the KrylovIIF3 scheme shows a linear computational complexity on single-grid as that for the second order scheme. The computational complexity of the KrylovIIF3 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  $\Omega = \{0 \leq x \leq \pi, 0 \leq y \leq \pi, 0 \leq z \leq \pi\}$ , 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 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 4.3 for the KrylovIIF2 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 KrylovIIF2 scheme's CPU time ratios shown in Table 4.3 verify its linear computational complexity. We also observe that the KrylovIIF2 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 4.3, 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 4.3. 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 (25, 244.30 seconds) is about 1/10 of that on a single-grid (225, 543.28 seconds). 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 seconds) is about 1/100 of that on a single-grid (225, 543.28 seconds), 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 KrylovIIF2 scheme here as an example to further analyze the computational complexity on single-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 4.4, 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 single-grid and sparse-grid studied in this dissertation with a fully implicit scheme with an advanced linear system solver such as a multigrid method. As an example, we apply the Crank-Nicolson scheme [16] in discretizing the 2D case here. A multigrid solver (the Two-Grid correction scheme) [5] 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 [5]. The results including numerical errors, accuracy orders and CPU times are reported in Table 4.5. 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 KrylovIIF2 scheme on single-grid and sparse-grid (Table 4.1). 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  $80 \times 80$  mesh) than the KrylovIIF2 scheme. However, on more refined meshes the KrylovIIF2 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 dissertation. The approach can be straightforwardly implemented on non-uniform rectangular meshes. Here we test the KrylovIIF2 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 4.6 for computations by the KrylovIIF2 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.

TABLE 4.1

EXAMPLE 1, 2D CASE, KRYLOVIF2 SCHEME, COMPARISON OF  
NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON  
SINGLE-GRID AND SPARSE-GRID.

Single-grid								
	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio	
	$80 \times 80$	$1.86 \times 10^{-4}$		$8.74 \times 10^{-5}$		3.56		
	$160 \times 160$	$4.66 \times 10^{-5}$	2.00	$2.25 \times 10^{-5}$	1.96	27.34	7.68	
	$320 \times 320$	$1.16 \times 10^{-5}$	2.00	$5.71 \times 10^{-6}$	1.98	219.15	8.02	
	$640 \times 640$	$2.91 \times 10^{-6}$	2.00	$1.44 \times 10^{-6}$	1.99	1,828.21	8.34	
Sparse-grid, refine root grids								
$N_r$	$N_L$	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80$	$1.83 \times 10^{-4}$		$9.15 \times 10^{-5}$		2.50	
20	3	$160 \times 160$	$4.57 \times 10^{-5}$	2.00	$2.29 \times 10^{-5}$	2.00	14.74	5.91
40	3	$320 \times 320$	$1.14 \times 10^{-5}$	2.00	$5.71 \times 10^{-6}$	2.00	104.47	7.09
80	3	$640 \times 640$	$2.86 \times 10^{-6}$	2.00	$1.43 \times 10^{-6}$	2.00	817.09	7.82
Sparse-grid, refine level								
$N_r$	$N_L$	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80$	$1.83 \times 10^{-4}$		$9.15 \times 10^{-5}$		2.50	
10	4	$160 \times 160$	$4.57 \times 10^{-5}$	2.00	$2.29 \times 10^{-5}$	2.00	9.33	3.74
10	5	$320 \times 320$	$1.08 \times 10^{-5}$	2.09	$5.38 \times 10^{-6}$	2.09	36.03	3.86
10	6	$640 \times 640$	$2.68 \times 10^{-6}$	2.00	$1.34 \times 10^{-6}$	2.00	142.53	3.96

TABLE 4.2

EXAMPLE 1, 2D CASE, KRYLOVIF3 SCHEME, COMPARISON OF  
NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON  
SINGLE-GRID AND SPARSE-GRID.

Single-grid								
		$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
		$80 \times 80$	$8.82 \times 10^{-7}$		$4.41 \times 10^{-7}$		7.45	
		$160 \times 160$	$5.63 \times 10^{-8}$	4.00	$2.82 \times 10^{-8}$	3.97	62.08	8.33
		$320 \times 320$	$3.56 \times 10^{-9}$	4.00	$1.78 \times 10^{-9}$	3.98	504.81	8.13
		$640 \times 640$	$2.33 \times 10^{-10}$	3.94	$1.17 \times 10^{-10}$	3.93	3,743.59	7.42
Sparse-grid, refine root grids								
$N_r$	$N_L$	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80$	$8.82 \times 10^{-7}$		$4.41 \times 10^{-7}$		7.85	
20	3	$160 \times 160$	$5.63 \times 10^{-8}$	3.97	$2.82 \times 10^{-8}$	3.97	48.09	6.13
40	3	$320 \times 320$	$3.56 \times 10^{-9}$	3.98	$1.78 \times 10^{-9}$	3.98	356.76	7.42
80	3	$640 \times 640$	$2.26 \times 10^{-10}$	3.98	$1.13 \times 10^{-10}$	3.98	2,850.46	7.99
Sparse-grid, refine level								
$N_r$	$N_L$	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80$	$8.82 \times 10^{-7}$		$4.41 \times 10^{-7}$		7.85	
10	4	$160 \times 160$	$5.63 \times 10^{-8}$	3.97	$2.82 \times 10^{-8}$	3.97	34.26	4.36
10	5	$320 \times 320$	$3.56 \times 10^{-9}$	3.98	$1.78 \times 10^{-9}$	3.98	152.83	4.46
10	6	$640 \times 640$	$2.26 \times 10^{-10}$	3.98	$1.13 \times 10^{-10}$	3.98	688.69	4.51





TABLE 4.3

*Continued*

$N_r$	$N_L$	$N_h \times N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80 \times 80$	$5.40 \times 10^{-5}$		$2.39 \times 10^{-5}$		89.35	
10	4	$160 \times 160 \times 160$	$1.54 \times 10^{-5}$	1.81	$6.68 \times 10^{-6}$	1.84	453.00	5.07
10	5	$320 \times 320 \times 320$	$4.21 \times 10^{-6}$	1.87	$1.78 \times 10^{-6}$	1.91	2,321.95	5.13
10	6	$640 \times 640 \times 640$	$1.09 \times 10^{-6}$	1.95	$4.53 \times 10^{-7}$	1.97	12,730.90	5.48

TABLE 4.4

EXAMPLE 1, 2D CASE, KRYLOVIF2 SCHEME, COMPARISON OF  
THE NUMBER OF MULTIPLICATION AND DIVISION OPERATIONS  
IN ONE TIME STEP.

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

TABLE 4.5

EXAMPLE 1, 2D CASE, CRANK-NICOLSON SCHEME WITH A  
MULTIGRID SOLVER (THE TWO-GRID CORRECTION SCHEME)  
FOR THE LINEAR SYSTEMS.

$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)
$80 \times 80$	$1.86 \times 10^{-4}$		$9.29 \times 10^{-5}$		1.81
$160 \times 160$	$4.65 \times 10^{-5}$	2.00	$2.32 \times 10^{-5}$	2.00	24.76
$320 \times 320$	$1.16 \times 10^{-5}$	2.00	$5.81 \times 10^{-6}$	2.00	352.57
$640 \times 640$	$2.90 \times 10^{-6}$	2.00	$1.45 \times 10^{-6}$	2.00	5,125.76

TABLE 4.6

EXAMPLE 1, 2D CASE, KRYLOVIF2 SCHEME, NON-UNIFORM  
GRIDS. COMPARISON OF NUMERICAL ERRORS AND CPU TIMES  
FOR COMPUTATIONS ON SINGLE-GRID AND SPARSE-GRID.

Single-grid								
	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio	
	$80 \times 80$	$1.17 \times 10^{-6}$		$3.44 \times 10^{-7}$		16.86		
	$160 \times 160$	$8.21 \times 10^{-8}$	3.83	$2.58 \times 10^{-8}$	3.73	190.58	11.30	
	$320 \times 320$	$6.75 \times 10^{-9}$	3.60	$2.96 \times 10^{-9}$	3.12	1,045.65	5.49	
	$640 \times 640$	$1.10 \times 10^{-9}$	2.61	$5.66 \times 10^{-10}$	2.39	8,495.37	8.12	
Sparse-grid, refine root grids								
$N_r$	$N_L$	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80$	$1.17 \times 10^{-6}$		$3.44 \times 10^{-7}$		9.72	
20	3	$160 \times 160$	$8.21 \times 10^{-8}$	3.83	$2.58 \times 10^{-8}$	3.73	65.78	6.77
40	3	$320 \times 320$	$6.75 \times 10^{-9}$	3.60	$2.96 \times 10^{-9}$	3.12	491.27	7.47
80	3	$640 \times 640$	$1.12 \times 10^{-9}$	2.60	$5.72 \times 10^{-10}$	2.37	3,852.75	7.84
Sparse-grid, refine level								
$N_r$	$N_L$	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80$	$1.17 \times 10^{-6}$		$3.44 \times 10^{-7}$		9.72	
10	4	$160 \times 160$	$8.21 \times 10^{-8}$	3.83	$2.58 \times 10^{-8}$	3.73	45.06	4.64
10	5	$320 \times 320$	$6.75 \times 10^{-9}$	3.60	$2.96 \times 10^{-9}$	3.12	206.02	4.57
10	6	$640 \times 640$	$1.10 \times 10^{-9}$	2.61	$5.66 \times 10^{-10}$	2.39	936.73	4.55

## 4.2 A 3D problem with anisotropic diffusion and constant diffusion coefficients

**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

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

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 [36], we show that for high dimensional problems with anisotropic diffusion terms, Krylov IIF schemes are more efficient than compact IIF methods [43]. It is interesting to test Krylov IIF scheme on sparse-grid for such problems with anisotropic diffusion terms. We compute the problem till final time  $T = 1$  by the KrylovIIF2 scheme (2.16) on both single-grid and sparse-grid. The  $L^\infty$  errors,  $L^2$  errors, the corresponding numerical accuracy orders, and CPU times on successively refined meshes are reported in Table 4.7.

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 4.7 that by performing 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 seconds) is about 22% of that on a single-grid (132,359.95 seconds), 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 seconds) is only 3.3% of that on a single-grid (132,359.95 seconds), and 96.7% CPU time is saved. We can also observe that if the mesh refinement is done by refining root grids, the KrylovIIF2 scheme on sparse grids has the linear computational complexity as that for the KrylovIIF2 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.

TABLE 4.7

EXAMPLE 2. KRYLOVIF2 SCHEME, COMPARISON OF  
NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON  
SINGLE-GRID AND SPARSE-GRID.

Single-grid								
		$N_h \times N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
		$80 \times 80 \times 80$	$6.97 \times 10^{-4}$		$4.93 \times 10^{-4}$		538.81	
		$160 \times 160 \times 160$	$1.74 \times 10^{-4}$	2.00	$1.23 \times 10^{-4}$	2.00	8,413.74	15.62
		$320 \times 320 \times 320$	$4.36 \times 10^{-5}$	2.00	$3.08 \times 10^{-5}$	2.00	132,359.95	15.73
Sparse-grid, refine root grids								
$N_r$	$N_L$	$N_h \times N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80 \times 80$	$7.49 \times 10^{-4}$		$5.13 \times 10^{-4}$		118.58	
20	3	$160 \times 160 \times 160$	$1.76 \times 10^{-4}$	2.09	$1.24 \times 10^{-4}$	2.05	1,817.95	15.33
40	3	$320 \times 320 \times 320$	$4.36 \times 10^{-5}$	2.10	$3.09 \times 10^{-5}$	2.01	29,573.60	16.27
80	3	$640 \times 640 \times 640$	$1.09 \times 10^{-5}$	2.00	$7.71 \times 10^{-6}$	2.00	465,538.00	15.74

TABLE 4.7

*Continued*

Sparse-grid, refine level								
$N_r$	$N_L$	$N_h \times N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80 \times 80$	$7.49 \times 10^{-4}$		$5.13 \times 10^{-4}$		118.58	
10	4	$160 \times 160 \times 160$	$1.87 \times 10^{-4}$	2.00	$1.30 \times 10^{-4}$	1.98	728.21	6.14
10	5	$320 \times 320 \times 320$	$4.73 \times 10^{-5}$	1.98	$3.28 \times 10^{-5}$	1.98	4,371.18	6.00
10	6	$640 \times 640 \times 640$	$1.20 \times 10^{-5}$	1.98	$8.30 \times 10^{-6}$	1.98	25,736.20	5.89

### 4.3 A 3D problem with anisotropic diffusion and variable diffusion coefficients

**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

$$\begin{aligned}
u_t = & 0.5u_{xx} - 0.5\sin(x+y)u_{xy} + 0.5u_{yy} \\
& + 0.5u_{xx} - \frac{1}{3}\cos y u_{xz} + \frac{1}{3}u_{zz} \\
& + 0.5(1+\cos x)u_{yy} - 0.5(1+\cos x)u_{yz} + \frac{1}{3}(1+\cos x)u_{zz} + f(x, y, z, u),
\end{aligned} \tag{4.1}$$

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

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

As the last example, in [36] we show that for this problem with anisotropic diffusion terms, Krylov IIF schemes are more efficient than compact IIF methods [43]. 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 KrylovIIF2 scheme (2.16) 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 4.8. 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 seconds) is about 36% of that on a single-grid (153,195.14 seconds), 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 seconds) is only 5.3% of that on a single-grid (153,195.14 seconds), and 94.7% CPU time is saved. We can also observe that if the mesh refinement is done by refining root grids, the KrylovIIF2 scheme on sparse grids has the linear computational complexity as that for the KrylovIIF2 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.

TABLE 4.8  
EXAMPLE 3. KRYLOVIF2 SCHEME, COMPARISON OF  
NUMERICAL ERRORS AND CPU TIMES FOR COMPUTATIONS ON  
SINGLE-GRID AND SPARSE-GRID.

Single-grid								
		$N_h \times N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
		$80 \times 80 \times 80$	$3.34 \times 10^{-3}$		$1.09 \times 10^{-3}$		551.57	
		$160 \times 160 \times 160$	$8.34 \times 10^{-4}$	2.00	$2.71 \times 10^{-4}$	2.01	8,992.13	16.30
		$320 \times 320 \times 320$	$2.09 \times 10^{-4}$	2.00	$6.79 \times 10^{-5}$	2.00	153,195.14	17.04
Sparse-grid, refine root grids								
$N_r$	$N_L$	$N_h \times N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80 \times 80$	$3.19 \times 10^{-3}$		$1.10 \times 10^{-3}$		229.37	
20	3	$160 \times 160 \times 160$	$8.13 \times 10^{-4}$	1.97	$2.70 \times 10^{-4}$	2.03	3,618.00	15.77
40	3	$320 \times 320 \times 320$	$2.07 \times 10^{-4}$	1.97	$6.77 \times 10^{-5}$	1.99	55,060.30	15.22
80	3	$640 \times 640 \times 640$	$5.21 \times 10^{-5}$	1.99	$1.70 \times 10^{-5}$	2.00	865,203.00	15.71

TABLE 4.8

*Continued*

Sparse-grid, refine level

$N_r$	$N_L$	$N_h \times N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80 \times 80$	$3.19 \times 10^{-3}$		$1.10 \times 10^{-3}$		229.37	
10	4	$160 \times 160 \times 160$	$7.85 \times 10^{-4}$	2.02	$2.82 \times 10^{-4}$	1.97	1,414.63	6.17
10	5	$320 \times 320 \times 320$	$1.94 \times 10^{-4}$	2.02	$7.30 \times 10^{-5}$	1.95	8,139.66	5.75
10	6	$640 \times 640 \times 640$	$4.83 \times 10^{-5}$	2.01	$1.90 \times 10^{-5}$	1.94	46,392.90	5.70

#### 4.4 A convection-diffusion problem

**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

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

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

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

The KrylovIIF2 scheme (2.16) 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 [24]. 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 4.9. 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. Specifically, for example for the  $640 \times 640$  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 4.9.

Again we can also observe that if the mesh refinement is done by refining root grids, KrylovIIF2 scheme on sparse grids has the linear computation complexity as that for the KrylovIIF2 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.

TABLE 4.9

EXAMPLE 4. KRYLOVIIF2 SCHEME, COMPARISON OF  
NUMERICAL ERRORS AND CPU TIMES

Single-grid							
$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio	
$80 \times 80$	$1.67 \times 10^{-4}$		$6.91 \times 10^{-5}$		13.77		
$160 \times 160$	$2.23 \times 10^{-5}$	2.91	$1.27 \times 10^{-5}$	2.44	104.16	7.56	
$320 \times 320$	$9.54 \times 10^{-6}$	1.22	$4.61 \times 10^{-6}$	1.47	851.93	8.18	
$640 \times 640$	$2.80 \times 10^{-6}$	1.77	$1.30 \times 10^{-6}$	1.82	6,958.20	8.17	

TABLE 4.9

*Continued*

Sparse-grid, refine root grids, $N_L = 3$								
$N_r$	$N_L$	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80$	$3.52 \times 10^{-3}$		$8.24 \times 10^{-4}$		13.24	
20	3	$160 \times 160$	$3.32 \times 10^{-5}$	6.72	$1.36 \times 10^{-5}$	5.92	81.58	6.16
40	3	$320 \times 320$	$9.44 \times 10^{-6}$	1.82	$4.60 \times 10^{-6}$	1.56	601.76	7.38
80	3	$640 \times 640$	$2.80 \times 10^{-6}$	1.76	$1.30 \times 10^{-6}$	1.82	4,712.98	7.83
Sparse-grid, refine root grids, $N_L = 4$								
$N_r$	$N_L$	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	4	$160 \times 160$	$4.97 \times 10^{-4}$		$1.07 \times 10^{-4}$		58.22	
20	4	$320 \times 320$	$8.70 \times 10^{-6}$	5.84	$4.48 \times 10^{-6}$	4.58	395.54	6.79
40	4	$640 \times 640$	$2.78 \times 10^{-6}$	1.65	$1.30 \times 10^{-6}$	1.78	3,001.72	7.59
80	4	$1280 \times 1280$	$7.46 \times 10^{-7}$	1.90	$3.43 \times 10^{-7}$	1.93	24,183.80	8.06
Sparse-grid, refine level								
$N_r$	$N_L$	$N_h \times N_h$	$L^\infty$ error	Order	$L^2$ error	Order	CPU(s)	Ratio
10	3	$80 \times 80$	$3.52 \times 10^{-3}$		$8.24 \times 10^{-4}$		13.24	
10	4	$160 \times 160$	$4.97 \times 10^{-4}$	2.82	$1.07 \times 10^{-4}$	2.94	58.22	4.40
10	5	$320 \times 320$	$3.88 \times 10^{-5}$	3.68	$8.63 \times 10^{-6}$	3.63	260.20	4.47
10	6	$640 \times 640$	$5.69 \times 10^{-6}$	2.77	$1.59 \times 10^{-6}$	2.44	1,170.34	4.50

## 4.5 Three dimensional Fokker-Planck equations

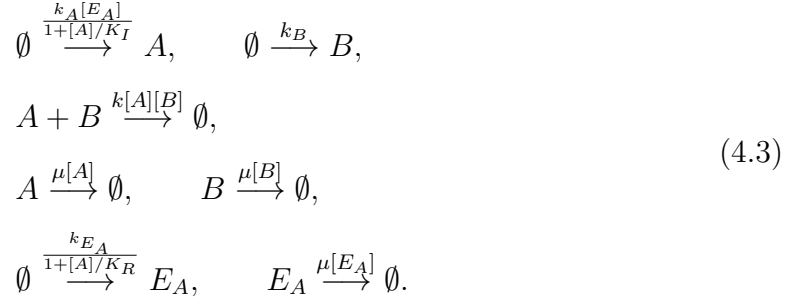
**Example 5 (Three dimensional Fokker-Planck equations).**

The Fokker-Planck equation (FPE) [44] 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 [53], Array-representation integration factor scheme was applied in solving FPEs which describe the time evolution of the probability density function of stochastic systems [44]. The general form of FPEs is

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = - \sum_{r=1}^R \left\{ \sum_{i=1}^N n_{ri} \frac{\partial}{\partial x_i} \left( q_r(\mathbf{x}, t) - \frac{1}{2} \sum_{j=1}^N n_{rj} \frac{\partial q_r(\mathbf{x}, t)}{\partial x_j} \right) \right\}, \quad (4.2)$$

where  $p(\mathbf{x}, t)$  is the probability density of the system at the state  $\mathbf{x} = (x_1, x_2, \dots, x_N)$  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.  $q_r(\mathbf{x}, t)$  is defined by  $q_r(\mathbf{x}, t) = w_r(\mathbf{x})p(\mathbf{x}, t)$ , where  $w_r(\mathbf{x}, t)$  is the reaction propensity function for  $r$ -th reaction at state  $\mathbf{x}$  and time  $t$ . Here we apply the second order Krylov IIF scheme KrylovIIF2 (2.16) on both single-grid and sparse-grid in solving a three dimensional Fokker-Planck equation [48] which involves two metabolites  $A$  and  $B$  and one enzyme  $E_A$  and show computational efficiency of the scheme on sparse-grid. Since KrylovIIF2 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 KrylovIIF2 scheme is used to continue the time evolution. The reactions are described as following (here  $\emptyset$  means that there is no reactant or

product in the reaction):



In this system, the total number of reactions  $R$  is 7, and the total number of chemical species  $N$  is 3. The vectors  $\mathbf{n}_r = (n_{r1}, n_{r2}, n_{r3})$  are  $\mathbf{n}_1 = (1, 0, 0)$ ,  $\mathbf{n}_2 = (0, 1, 0)$ ,  $\mathbf{n}_3 = (-1, -1, 0)$ ,  $\mathbf{n}_4 = (-1, 0, 0)$ ,  $\mathbf{n}_5 = (0, -1, 0)$ ,  $\mathbf{n}_6 = (0, 0, 1)$ ,  $\mathbf{n}_7 = (0, 0, -1)$ . We denote the system state  $\mathbf{x}$  by  $\mathbf{x} = (x_1, x_2, x_3)$  which is  $([A], [B], [E_A])$  in this case. Then the propensity functions  $w_r(\mathbf{x})$  are

$$\begin{aligned}
w_1 &= \frac{k_A x_3}{1 + x_1/K_I}, & w_2 &= k_B, & w_3 &= k x_1 x_2, \\
w_4 &= \mu x_1, & w_5 &= \mu x_2, & w_6 &= \frac{k_{E_A}}{1 + x_1/K_R}, & w_7 &= \mu x_3,
\end{aligned} \tag{4.4}$$

where  $k_A = 0.3s^{-1}$ ,  $k_B = 2s^{-1}$ ,  $K_I = 30$ ,  $k = 0.001s^{-1}$ ,  $\mu = 0.004s^{-1}$ ,  $K_R = 30$  and  $k_{E_A} = 1s^{-1}$  [48]. Then the FPE can be written as

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -(L_1 + L_2 + L_3 + L_4 + L_5 + L_6 + L_7), \tag{4.5}$$



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

$$\begin{aligned}
L_1 &= \frac{\partial q_1(\mathbf{x}, t)}{\partial x_1} - \frac{1}{2} \frac{\partial^2 q_1(\mathbf{x}, t)}{\partial x_1^2}, \\
L_2 &= \frac{\partial q_2(\mathbf{x}, t)}{\partial x_2} - \frac{1}{2} \frac{\partial^2 q_2(\mathbf{x}, t)}{\partial x_2^2}, \\
L_3 &= -\frac{\partial q_3(\mathbf{x}, t)}{\partial x_1} - \frac{\partial q_3(\mathbf{x}, t)}{\partial x_2} - \frac{1}{2} \left( \frac{\partial^2 q_3(\mathbf{x}, t)}{\partial x_1^2} + \frac{\partial^2 q_3(\mathbf{x}, t)}{\partial x_2^2} + 2 \frac{\partial^2 q_3(\mathbf{x}, t)}{\partial x_1 \partial x_2} \right), \\
L_4 &= -\frac{\partial q_4(\mathbf{x}, t)}{\partial x_1} - \frac{1}{2} \frac{\partial^2 q_4(\mathbf{x}, t)}{\partial x_1^2}, \\
L_5 &= -\frac{\partial q_5(\mathbf{x}, t)}{\partial x_2} - \frac{1}{2} \frac{\partial^2 q_5(\mathbf{x}, t)}{\partial x_2^2}, \\
L_6 &= \frac{\partial q_6(\mathbf{x}, t)}{\partial x_3} - \frac{1}{2} \frac{\partial^2 q_6(\mathbf{x}, t)}{\partial x_3^2}, \\
L_7 &= -\frac{\partial q_7(\mathbf{x}, t)}{\partial x_3} - \frac{1}{2} \frac{\partial^2 q_7(\mathbf{x}, t)}{\partial x_3^2}.
\end{aligned} \tag{4.6}$$

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  $\Delta t$  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 4.10, 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  $\Delta t$  are obtained by

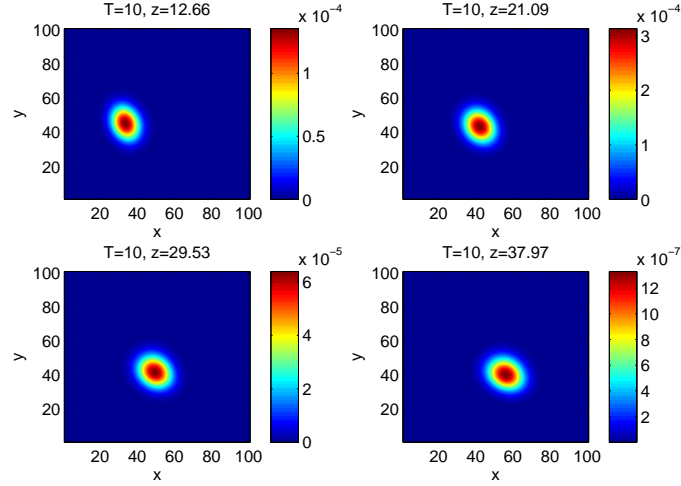


Figure 4.1. Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on single-grid. Final time  $T = 10$ .  $\Delta t = 0.015$ . Distribution of molecular species A and B with  $E_A = 12.66, 21.09, 29.53, 37.97$ .

calculating the difference of numerical values for  $\Delta t$  and  $\Delta t/2$ . 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  $T = 10$  with  $\Delta t = 0.015$ , in Table 4.11. The CPU times in Table 4.11 show that a significant amount of CPU time (82% CPU time) is saved by using the sparse-grid combination technique. In Figures 4.2, 4.4, and 4.6, we show contour plots of the numerical solutions by the KrylovIIF2 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 Figures 4.1, 4.3, and 4.5. We see that both approaches generate similar numerical solutions.

TABLE 4.10

NUMERICAL ERRORS AND ACCURACY ORDERS FOR THE  
KRYLOVIF2 SCHEME TO SOLVE THE 3D FOKKER-PLANCK  
EQUATION ON SINGLE AND SPARSE GRIDS.

On single-grid		
time step	$L^\infty$ error	accuracy
$\Delta t$	$1.20 \times 10^{-11}$	
$\Delta t/2$	$3.04 \times 10^{-12}$	1.99
$\Delta t/4$	$7.61 \times 10^{-13}$	2.00
On sparse-grid		
time step	$L^\infty$ error	accuracy
$\Delta t$	$1.32 \times 10^{-11}$	
$\Delta t/2$	$3.40 \times 10^{-12}$	1.96
$\Delta t/4$	$8.41 \times 10^{-13}$	2.01

TABLE 4.11

CPU TIME FOR THE KRYLOVIF2 SCHEME TO SOLVE THE 3D  
FOKKER-PLANCK EQUATION ON SINGLE AND SPARSE GRIDS

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

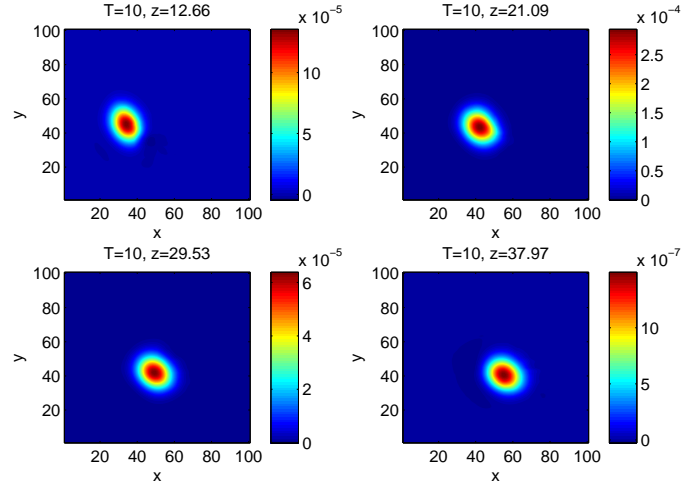


Figure 4.2. Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on sparse-grid. Final time  $T = 10$ .  $\Delta t = 0.015$ .  
Distribution of molecular species A and B with  
 $E_A = 12.66, 21.09, 29.53, 37.97$ .

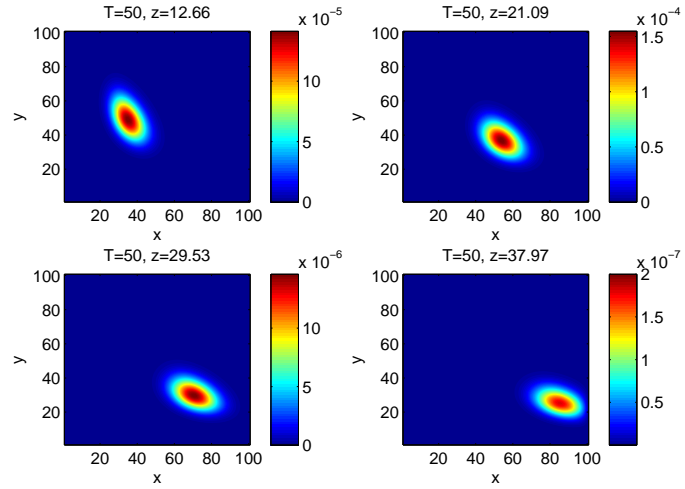


Figure 4.3. Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on single-grid. Final time  $T = 50$ .  $\Delta t = 0.015$ .  
Distribution of molecular species A and B with  
 $E_A = 12.66, 21.09, 29.53, 37.97$ .

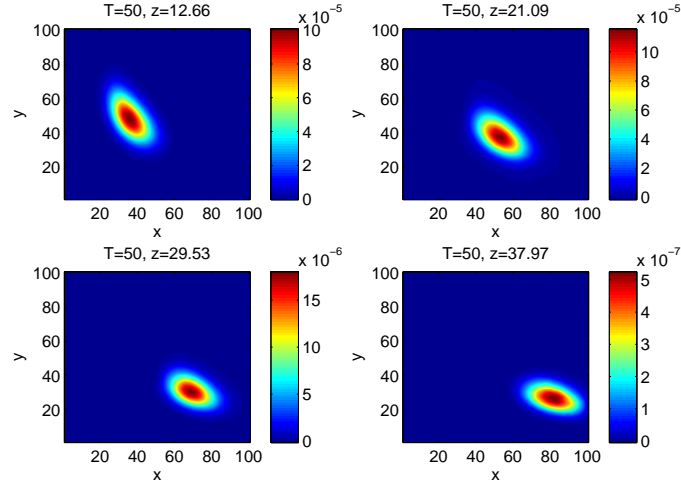


Figure 4.4. Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on sparse-grid. Final time  $T = 50$ .  $\Delta t = 0.015$ . Distribution of molecular species A and B with  $E_A = 12.66, 21.09, 29.53, 37.97$ .

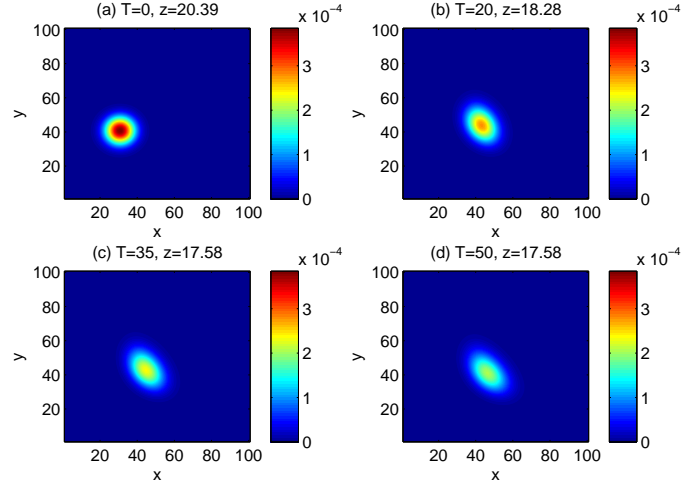


Figure 4.5. Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on single-grid. Distribution of molecular species A and B with different  $E_A$  values, at time  $T = 0, 20, 35, 50$ .  $\Delta t = 0.015$ .

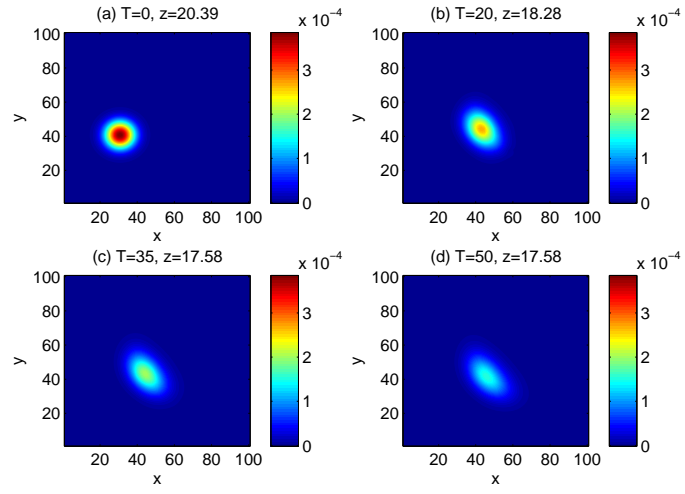


Figure 4.6. Numerical solutions of the 3D Fokker-Planck equation using the KrylovIIF2 scheme on sparse-grid. Distribution of molecular species A and B with different  $E_A$  values, at time  $T = 0, 20, 35, 50$ .  $\Delta t = 0.015$ .

## CHAPTER 5

### CONCLUSION

In the first part of this dissertation, 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 dissertation, 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 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' exponen-

tials 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 [7, 24, 25],  $M$  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 dissertation, 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 possible dependence of the parameter  $M$  on different types of PDEs (different differential operators) and problems, which is one of our future work.

In the second part of this dissertation, we design the Krylov IIF scheme on sparse grids for solving high spatial dimension problems. 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 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.



Another recent interesting work on the IIF method is to apply it in solving stochastic reaction-diffusion equations in [50]. 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 [50], 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 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 dissertation 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 dissertation. This is one of our future research.

## BIBLIOGRAPHY

1. U. M. Ascher, S. J. Ruuth, and B. Wetton. *Implicit-explicit methods for time-dependent PDE's*. University of British Columbia, Department of Computer Science, 1993.
2. H. L. Ashe and M. Levine. Local inhibition and long-range enhancement of dpp signal transduction by sog. *Nature*, 398(6726):427–431, 1999.
3. G. Beylkin, J. M. Keiser, and L. Vozovoi. A new class of time discretization schemes for the solution of nonlinear pdes. *Journal of Computational Physics*, 147(2):362–387, 1998.
4. A. Bourlioux, A. T. Layton, and M. L. Minion. High-order multi-implicit spectral deferred correction methods for problems of reactive flow. *Journal of Computational Physics*, 189(2):651–675, 2003.
5. W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial*. SIAM, 2000.
6. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.
7. S. Chen and Y.-T. Zhang. Krylov implicit integration factor methods for spatial discretization on high dimensional unstructured meshes: application to discontinuous galerkin methods. *Journal of Computational Physics*, 230(11):4336–4352, 2011.
8. A. Christlieb, B. Ong, and J.-M. Qiu. Integral deferred correction methods constructed with high order runge-kutta integrators. *Mathematics of Computation*, 79(270):761–783, 2010.
9. K. Christopher A and C. Mark H. Additive runge-kutta schemes for convection-diffusion-reaction equations. 2001.
10. S. M. Cox and P. C. Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.
11. A. Dutt, L. Greengard, and V. Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT Numerical Mathematics*, 40(2):241–266, 2000.

12. E. Gallopoulos and Y. Saad. Efficient solution of parabolic equations by krylov approximation methods. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1236–1264, 1992.
13. S. Gottlieb and C.-W. Shu. Total variation diminishing runge-kutta schemes. *Mathematics of computation of the American Mathematical Society*, 67(221):73–85, 1998.
14. S. Gottlieb, C.-W. Shu, and E. Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM review*, 43(1):89–112, 2001.
15. M. Griebel, M. Schneider, and C. Zenger. *A combination technique for the solution of sparse grid problems*. Citeseer, 1990.
16. B. Gustafsson, H.-O. Kreiss, and J. Oliger. *Time dependent problems and difference methods*, volume 24. John Wiley & Sons, 1995.
17. A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, iii. *Journal of computational physics*, 71(2):231–303, 1987.
18. N. J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Journal on Matrix Analysis and Applications*, 26(4):1179–1193, 2005.
19. M. Hochbruck and C. Lubich. On krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 34(5):1911–1925, 1997.
20. C. Hu and C.-W. Shu. Weighted essentially non-oscillatory schemes on triangular meshes. *Journal of Computational Physics*, 150(1):97–127, 1999.
21. J. Huang, J. Jia, and M. Minion. Arbitrary order krylov deferred correction methods for differential algebraic equations. *Journal of Computational Physics*, 221(2):739–760, 2007.
22. W. Hundsdorfer and J. G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, volume 33. Springer Science & Business Media, 2013.
23. G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted eno schemes. *Journal of computational physics*, 126(1):202–228, 1996.
24. T. Jiang and Y.-T. Zhang. Krylov implicit integration factor weno methods for semilinear and fully nonlinear advection–diffusion–reaction equations. *Journal of Computational Physics*, 253:368–388, 2013.
25. T. Jiang and Y.-T. Zhang. Krylov single-step implicit integration factor weno methods for advection–diffusion–reaction equations. *Journal of Computational Physics*, 311:22–44, 2016.

26. L. Ju, J. Zhang, L. Zhu, and Q. Du. Fast explicit integration factor methods for semilinear parabolic equations. *Journal of Scientific Computing*, 62(2):431–455, 2015.
27. A. Kanevsky, M. H. Carpenter, D. Gottlieb, and J. S. Hesthaven. Application of implicit–explicit high order runge–kutta methods to discontinuous-galerkin schemes. *Journal of Computational Physics*, 225(2):1753–1781, 2007.
28. A.-K. Kassam and L. N. Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.
29. A. Lander, Q. Nie, F. Wan, and Y.-T. Zhang. Localized ectopic expression of dpp receptors in a drosophila embryo. *Studies in Applied Mathematics*, 123(2):175–214, 2009.
30. B. Lastdrager, B. Koren, and J. Verwer. The sparse-grid combination technique applied to time-dependent advection problems. In *Multigrid Methods VI*, pages 143–149. Springer, 2000.
31. B. Lastdrager, B. Koren, and J. Verwer. Solution of time-dependent advection-diffusion problems with the sparse-grid combination technique and a rosenbrock solver. *Computational Methods in Applied Mathematics Comput. Methods Appl. Math.*, 1(1):86–98, 2001.
32. A. T. Layton and M. L. Minion. Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics. *Journal of Computational Physics*, 194(2):697–715, 2004.
33. X. Liu and Q. Nie. Compact integration factor methods for complex domains and adaptive mesh refinement. *Journal of computational physics*, 229(16):5692–5706, 2010.
34. X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of computational physics*, 115(1):200–212, 1994.
35. Y. Liu and Y.-T. Zhang. A robust reconstruction for unstructured weno schemes. *Journal of Scientific Computing*, 54(2-3):603–621, 2013.
36. D. Lu and Y.-T. Zhang. Krylov integration factor method on sparse grids for high spatial dimension convection–diffusion equations. *Journal of Scientific Computing*, 69(2):736–763, 2016.
37. J. Lu, J. Fang, S. Tan, C.-W. Shu, and M. Zhang. Inverse lax–wendroff procedure for numerical boundary conditions of convection–diffusion equations. *Journal of Computational Physics*, 317:276–300, 2016.
38. Y. Maday, A. T. Patera, and E. M. Rønquist. An operator-integration-factor splitting method for time-dependent problems: application to incompressible fluid flow. *Journal of Scientific Computing*, 5(4):263–292, 1990.

39. M. L. Minion et al. Semi-implicit spectral deferred correction methods for ordinary differential equations. *Communications in Mathematical Sciences*, 1(3): 471–500, 2003.
40. C. M. Mizutani, Q. Nie, F. Y. Wan, Y.-T. Zhang, P. Vilmos, R. Sousa-Neves, E. Bier, J. L. Marsh, and A. D. Lander. Formation of the bmp activity gradient in the drosophila embryo. *Developmental cell*, 8(6):915–924, 2005.
41. C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):3–49, 2003.
42. Q. Nie, Y.-T. Zhang, and R. Zhao. Efficient semi-implicit schemes for stiff systems. *Journal of Computational Physics*, 214(2):521–537, 2006.
43. Q. Nie, F. Y. Wan, Y.-T. Zhang, and X.-F. Liu. Compact integration factor methods in high spatial dimensions. *Journal of Computational Physics*, 227(10): 5238–5255, 2008.
44. H. Risken. The fokker-planck equation. methods of solution and applications, vol. 18 of. *Springer Series in Synergetics*, 1989.
45. C.-W. Shu. Total-variation-diminishing time discretizations. *SIAM Journal on Scientific and Statistical Computing*, 9(6):1073–1084, 1988.
46. C.-W. Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced numerical approximation of nonlinear hyperbolic equations*, pages 325–432. Springer, 1998.
47. C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988.
48. P. Sjöberg, P. Lötstedt, and J. Elf. Fokker–planck approximation of the master equation in molecular biology. *Computing and Visualization in Science*, 12(1): 37–50, 2009.
49. G. Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968.
50. C. Ta, D. Wang, and Q. Nie. An integration factor method for stochastic and stiff reaction–diffusion systems. *Journal of computational physics*, 295:505–522, 2015.
51. L. N. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.
52. J. G. Verwer, B. P. Sommeijer, and W. Hundsdorfer. Rkc time-stepping for advection–diffusion–reaction problems. *Journal of Computational Physics*, 201(1):61–79, 2004.

53. D. Wang, L. Zhang, and Q. Nie. Array-representation integration factor method for high-dimensional systems. *Journal of computational physics*, 258:585–600, 2014.
54. C. Zenger. Sparse grid. In W. Hackbusch, editor, *Notes on Numerical Fluid Mechanics*, volume 31, pages 241–251. Vieweg, Braunschweig, 1991.
55. Y.-T. Zhang and C.-W. Shu. High-order weno schemes for hamilton–jacobi equations on triangular meshes. *SIAM Journal on Scientific Computing*, 24(3):1005–1030, 2003.
56. Y.-T. Zhang and C.-W. Shu. Third order weno scheme on three dimensional tetrahedral meshes. *Communications in Computational Physics*, 5(2-4):836–848, 2009.
57. X. Zhong. Additive semi-implicit runge–kutta methods for computing high-speed nonequilibrium reactive flows. *Journal of Computational Physics*, 128(1):19–31, 1996.