SPEEDING UP HIGH-ORDER ALGORITHMS IN COMPUTATIONAL FLUID AND KINETIC DYNAMICS: BASED ON CHARACTERISTICS TRACING AND LOW-RANK STRUCTURES

by

Joseph Nakao

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Applied Mathematics.

Summer 2023

SPEEDING UP HIGH-ORDER ALGORITHMS IN COMPUTATIONAL FLUID AND KINETIC DYNAMICS: BASED ON CHARACTERISTICS TRACING AND LOW-RANK STRUCTURES

by

Joseph Nakao

Approved: _____

Mark Gockenbach, Ph.D.
Chair of the Department of Mathematical Sciences

Approved: _____

John A. Pelesko, Ph.D.
Dean of the College of Arts and Sciences

Approved: _____

Louis F. Rossi, Ph.D.
Vice Provost for Graduate and Professional Education and
Dean of the Graduate College

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Jingmei Qiu, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Tobin Driscoll, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Peter Monk, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Michael Shay, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

William Taitano, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

I want to express my sincere gratitude to the many individuals that have supported me these past five years. They have made this such an incredible journey.

First and foremost, I want to thank my advisor, Dr. Jingmei Qiu, for her unparalleled patience, encouragement and mentorship. Through her example she has shown me what it means to be an accomplished scholar, a superb educator and a caring person. Working with her has been an utmost privilege, and this dissertation was only possible because of her enduring support.

Additional thanks goes out to my other committee members, Dr. Tobin Driscoll, Dr. Peter Monk, Dr. Michael Shay and Dr. William Taitano, for taking the time to read my dissertation and provide their invaluable feedback.

I also want to thank my collaborators for their fruitful discussions, gracious support and lasting connections. In particular, I thank my committee member, Dr. William Taitano (Los Alamos National Laboratory), Dr. Alexander Alekseenko (California State University at Northridge), Dr. Robert Martin (Army Research Laboratory) and Dr. Jiajie Chen (University of Pennsylvania). Furthermore, I want to express my gratitude to the Spectra board of directors, especially Dr. Ron Buckmire (Occidental College), for giving me the opportunity to support the LGBTQ+ mathematics community. I also wish to thank the faculty and staff of the Department of Mathematical Sciences for providing a wonderful educational experience, including the late Francisco-Javier Sayas, Deborah See and Pamela Irwin.

Last but certainly not least, a very special thanks goes out to my family and friends. In particular, thank you to my parents, Mary Lou Nakao and Jerry Nakao, for their endless love and support. They have supported me since the very beginning, and words cannot express how lucky I am to have them as my parents.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **x**
LIST OF FIGURES . . . . . **xiii**
ABSTRACT . . . . . **xvii**

Chapter

1 INTRODUCTION **1**

1.1 Kinetic models vs fluid models 1

1.1.1 Starting with a kinetic model 2
1.1.2 Deriving a fluid model from a kinetic model 2
1.1.3 The moment closure problem 4

1.2 Eulerian, Lagrangian and Eulerian-Lagrangian frameworks 6

1.2.1 Eulerian framework 6
1.2.2 Lagrangian framework 8
1.2.3 Eulerian-Lagrangian framework 9

1.3 The proposed schemes 10

1.3.1 An Eulerian-Lagrangian scheme for convection-diffusion equations 10
1.3.2 A low-rank Eulerian scheme for diffusion equations 12
1.3.3 A low-rank Eulerian scheme for the Vlasov-Fokker-Planck equation 15

1.4 Organization of the dissertation 17

2 AN EULERIAN-LAGRANGIAN RUNGE-KUTTA FINITE VOLUME (EL-RK-FV) METHOD FOR SOLVING CONVECTION AND CONVECTION-DIFFUSION EQUATIONS 19

2.1 Review of technical components 20

2.1.1 Spatial reconstructions: WENO and WENO-AO 20

2.1.1.1 WENO5 21
2.1.1.2 WENO-AO(5,3) 24

2.1.2 Time discretizations: Runge-Kutta methods 29

2.1.2.1 Strong stability-preserving Runge-Kutta methods . . 31
2.1.2.2 Implicit-explicit Runge-Kutta methods 33

2.1.3 Operator splitting 36

2.1.3.1 First-order Lie-Trotter splitting 37
2.1.3.2 Second-order Strang splitting 37
2.1.3.3 Higher-order splitting 38

2.1.4 Gauss-Legendre quadrature 39

2.2 The EL-RK-FV method for pure convection problems 41

2.2.1 Scheme formulation 41
2.2.2 Solution remapping onto a traceback grid 43
2.2.3 Reconstruction of point values 46
2.2.4 Time evolution with explicit Runge-Kutta methods 48
2.2.5 Two-dimensional problems 50

2.2.5.1 Going from/to cell averages to/from interval averages 52
2.2.5.2 A demonstration with Strang splitting 53

2.3 The EL-RK-FV method for convection-diffusion equations 55

2.3.1 Computing the uniform cell averages of u_{xx} 56
2.3.2 Time evolution with implicit-explicit Runge-Kutta methods . 56

2.3.3 Mass conservation 61

2.4 Numerical tests 62

2.4.1 Pure convection problems: one-dimensional tests 63
2.4.2 Pure convection problems: two-dimensional tests 65
2.4.3 Convection-diffusion equations: one-dimensional tests 70
2.4.4 Convection-diffusion equations: two-dimensional tests 75

2.5 Conclusions and follow-up work 81

3 IMPLICIT LOW-RANK INTEGRATORS FOR SOLVING DIFFUSION EQUATIONS 85

3.1 Review of technical components 86

3.1.1 Tensor decompositions 86

3.1.1.1 Singular value decomposition (SVD) 87
3.1.1.2 QR factorization 91
3.1.1.3 CP decomposition 93

3.1.2 Low-rank tensor approaches for time-dependent PDEs 96

3.1.2.1 Step-and-truncate methods 96
3.1.2.2 Dynamical low-rank (DLR) methods 99

3.1.3 von Neumann stability analysis 101

3.1.3.1 Backward Euler (bE) 102
3.1.3.2 Crank-Nicolson (CN) 104
3.1.3.3 Backward differentiation formula (BDF2) 105
3.1.3.4 Diagonally implicit Runge-Kutta (DIRK2) 106

3.2 The implicit low-rank scheme 110

3.2.1 A first-order scheme using backward Euler 112
3.2.2 A second-order scheme using DIRK2 117
3.2.3 Computational complexity 122

3.2.3.1 Computational complexity of solving the Sylvester equation 122

3.2.3.2 Computational complexity of the proposed scheme . 124

3.3 Numerical tests 125

3.3.1 CPU runtime 127
3.3.2 Convergence analysis 133
3.3.3 Rank evolution 139

3.4 Conclusions and follow-up work 142

4 A LOW-RANK TENSOR SCHEME WITH STRUCTURE-PRESERVING QUALITIES FOR SOLVING THE 1D2V VLASOV-FOKKER-PLANCK EQUATION 144

4.1 Review of technical components 145

4.1.1 A structure-preserving Chang-Cooper (SPCC) discretization in Cartesian coordinates 146
4.1.2 A matrix exponential-based solver for large linear systems of tensor-product structure 150

4.2 Discretizing the 1D2V Vlasov-Leonard-Bernstein-Fokker-Planck equation 153

4.2.1 The macroscopic system and kinetic fluxes 155
4.2.2 Scheme formulation: discretizing in physical space-time 157
4.2.3 Scheme formulation: discretizing in velocity space 158

4.2.3.1 Computing the macroscopic quantities 160
4.2.3.2 Discretizing the collision operator 163
4.2.3.3 Discretizing the acceleration term 166

4.3 Updating and truncating the discretized equation 167

4.3.1 Solving a linear system of tensor-product structure 167
4.3.2 SVD truncation 169
4.3.3 The first-order scheme 169

4.4 Numerical tests 170

4.4.1 The 0D2V Leonard-Bernstein-Fokker-Planck equation 171

4.4.2 The 1D2V Vlasov-Leonard-Bernstein-Fokker-Planck equation . 173

4.5 Conclusions and follow-up work 180

BIBLIOGRAPHY **181**

Appendix

A AN ILLUSTRATIVE EXAMPLE WITH IMEX(2,2,2) **195**
B THE SECOND-ORDER SCHEME WITH CRANK-NICOLSON **197**
C THE SECOND-ORDER SCHEME WITH BDF2 . . . . . **201**
D NONDIMENSIONALIZING THE 1D2V VLASOV-LEONARD-BERNSTEIN-FOKKER-PLANCK EQUATION IN CYLINDRICAL COORDINATES **205**
E DERIVING THE BALANCE EQUATIONS FOR TOTAL MASS, MOMENTUM AND ENERGY **208**
F STENGER QUADRATURE NODES AND WEIGHTS **213**
G A QUASI-NEWTON SOLVER FOR THE MACROSCOPIC SYSTEM **214**
H PERMISSIONS **217**

LIST OF TABLES

2.1 Nodes and weights for the order-n Gauss-Legendre quadrature over $[-1, 1]$. 40

2.2 Convergence study with spatial mesh refinement for equation (2.4.2) with forward Euler at $T_f = 1$. 64

2.3 Convergence study with spatial mesh refinement for equation (2.4.3) with RK4 at $T_f = 1$. 64

2.4 Convergence study with spatial mesh refinement for equation (2.4.4) with RK4 at $T_f = 1$. 65

2.5 Convergence study with spatial mesh refinement for equation (2.4.5) with forward Euler and $CFL = 8$ at $T_f = 1$. 66

2.6 Convergence study with spatial mesh refinement for equation (2.4.6) with RK4 and $CFL = 0.95$ at $T_f = 0.5$. 67

2.7 Convergence study with spatial mesh refinement for equation (2.4.6) with RK4 and $CFL = 8$ at $T_f = 0.5$. 68

2.8 Convergence study with spatial mesh refinement for equation (2.4.7) with RK4 and $CFL = 0.95$ at $T_f = 1.5$. 69

2.9 Convergence study with spatial mesh refinement for equation (2.4.7) with RK4 and $CFL = 8$ at $T_f = 1.5$. 70

2.10 Convergence study with spatial mesh refinement for equation (2.4.10) with IMEX(2,3,3) at $T_f = 1$. 71

2.11 Convergence study with spatial mesh refinement for equation (2.4.11) with IMEX(2,3,3) at $T_f = 1$. 72

2.12 Convergence study with spatial mesh refinement for equation (2.4.12) with IMEX(2,3,3) at $T_f = 1$. 74

2.13 Convergence study with spatial mesh refinement for equation (2.4.13) with IMEX(4,4,3) at $T_f = 1$. 75

2.14 Convergence study with spatial mesh refinement for equation (2.4.15) with IMEX(2,3,3) and Strang splitting at $T_f = 0.5$. 76

2.15 Convergence study with spatial mesh refinement for equation (2.4.16) with IMEX(4,4,3) and Strang splitting at $T_f = 0.5$. 77

2.16 Convergence study with spatial mesh refinement for equation (2.4.17) with IMEX(2,3,3) and Strang splitting at $T_f = 0.1$. 78

2.17 Convergence study with spatial mesh refinement for equation (2.4.18) with IMEX(2,3,3) and Strang splitting at $T_f = 0.5$. 79

2.18 $n = \pi$, $\overline{\mathbf{v}} = \mathbf{0}$, and $T = 3$. 80

2.19 Convergence study with spatial mesh refinement for equation (2.4.19) with IMEX(2,3,3) and Strang splitting at $T_f = 0.5$. 82

3.1 The dominant computational cost to set up and solve the Sylvester equations applicable to the proposed implicit low-rank integrator. The reference Sylvester equations are equations (3.2.6) and (3.2.8). 123

3.2 CPU runtime of the second-order ADI method. Final time $T_f = 1$ and time-stepping size $\Delta t = 0.05$. 128

3.3 CPU runtime of the first-order scheme with backward Euler. We use the rank-2 initial condition (3.3.2), initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 0.05$. 129

3.4 CPU runtime of the second-order scheme with stiffly-accurate DIRK2. We use the rank-2 initial condition (3.3.2), initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 0.05$. 131

3.5 CPU runtime of the second-order scheme with Crank-Nicolson. We use the rank-2 initial condition (3.3.2), initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 0.05$. 132

3.6 CPU runtime of the second-order scheme with BDF2. We use the rank-2 initial condition (3.3.2), initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 0.05$. 133

3.7 Convergence study with spatial mesh refinement of the second-order ADI method. Final time $T_f = 1$ and time-stepping size $\Delta t = 3\Delta x$. . 134

3.8 Convergence study with spatial mesh refinement of the first-order scheme with backward Euler (with diagonalization). Final time $T_f = 1$, initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 3\Delta x$. 135

3.9 Convergence study with spatial mesh refinement of the second-order scheme with stiffly-accurate DIRK2 (with diagonalization). Final time $T_f = 1$, initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 3\Delta x$. 136

3.10 Convergence study with spatial mesh refinement of the second-order scheme with Crank-Nicolson (with diagonalization). Final time $T_f = 1$, initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 3\Delta x$. 138

3.11 Convergence study with spatial mesh refinement of the second-order scheme with BDF2 (with diagonalization). Final time $T_f = 1$, initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 3\Delta x$. 139

4.1 $R = 1$, $n = \pi$, $\overline{\mathbf{v}} = \mathbf{0}$ and $T = 3$. 172

4.2 The average CPU runtime per time-step simulating the VLBFP equation to relaxation. Spatial mesh $N_x = 80$, velocity mesh $N_{||} \times N_{\perp} = 120 \times 120$, singular value tolerance $\epsilon = 1.0E-05$, Stenger quadrature $K = 15$. 180

D.1 Reference quantities. 207

LIST OF FIGURES

1.1 (left) The Eulerian framework; (right) The Lagrangian framework. . 6

1.2 Tracing the characteristics backwards in time for the Lax-Wendroff method. 8

1.3 (left) Approximating the characteristics in the Eulerian-Lagrangian framework; (right) Following the characteristics exactly in the semi-Lagrangian framework. 9

2.1 (left) Without WENO; (right) With WENO. 23

2.2 The space-time region Ω_j. 42

2.3 The space-time region ${}_1\Omega_j$. 59

2.4 The space-time region ${}_2\Omega_j$. 59

2.5 Error plot corresponding to equation (2.4.3) using RK4 with final time $T_f = 0.5$. 66

2.6 Error plot corresponding to equation (2.4.4) using RK4 with final time $T_f = 0.5$. 66

2.7 Error plot for (2.4.6) with RK4 at $T_f = 0.5$. $N_x = N_y = 200$. 68

2.8 Plot of the numerical solution to (2.4.6) with SSP RK3 and $CFL = 2.2$ at $T_f = 2\pi$. $N_x = N_y = 100$. 68

2.9 Error plot for (2.4.7) with RK4 at $T_f = 1.5$. $N_x = N_y = 200$. 70

2.10 Plot of the numerical solution to (2.4.7) with $g(t) = 1$, SSP RK3 and $CFL = 8$ at $T_f = 5\pi$. $N_x = N_y = 100$. 70

2.11 IMEX(2,3,3), $\epsilon = 1$, Final time $T_f = 0.5$. 72

2.12 IMEX(2,3,3), $\epsilon = 1$, Final time $T_f = 0.5$. 72

2.13 IMEX(2,3,3), $\epsilon = 0.1$, Final time $T_f = 0.5$. 75

2.14 IMEX(4,4,3), Final time $T_f = 0.5$. 75

2.15 IMEX(2,3,3), $\epsilon = 1$, Final time $T_f = 0.5$. 77

2.16 IMEX(4,4,3), $\epsilon = 1$, Final time $T_f = 0.1$. 77

2.17 IMEX(2,3,3), $\epsilon = 1$, Final time $T_f = 0.1$. 79

2.18 IMEX(2,3,3), $\epsilon = 0.1$, Final time $T_f = 0.2$. 79

2.19 IMEX(2,3,3), Final time $T_f = 0.1$. 82

2.20 Figures (a)-(e): Relative macro-parameters for equation (2.4.19) with initial distribution of two Maxwellians defined by Table 2.18. Mesh $N_{v_x} = N_{v_y} = 200$, $CFL = 6$. Figure (f): The initial distribution. . . 83

2.21 Various snapshots of the numerical solution to equation (2.4.19) with initial distribution of two Maxwellians defined by Table 2.18. Mesh $N_{v_x} = N_{v_y} = 200$, $CFL = 6$. Times: 0.15, 0.30, 0.45, 0.60, 0.75, 3. . 84

3.1 Plotting the amplification factor for backward Euler method applied to the heat equation. 103

3.2 Plotting the amplification factor for Crank-Nicolson method applied to the heat equation. 104

3.3 Plotting the absolute value of the amplification factor $A_+(\xi)$ for BDF2 applied to the heat equation. 106

3.4 Plotting the absolute value of the amplification factor $A_-(\xi)$ for BDF2 applied to the heat equation. 107

3.5 Plotting the amplification factor for DIRK2 applied to the heat equation. 109

3.6 Error plot of second-order ADI method with mesh 320×320. The errors are shown for initial conditions (3.3.2)-(3.3.4). 135

3.7 Error plot of first-order scheme using backward Euler with mesh 320×320 and tolerance $\epsilon = 1.0E-10$. The errors are shown for initial conditions (3.3.2)-(3.3.4). 136

3.8 Error plot of second-order scheme using stiffly-accurate DIRK2 with mesh 320×320 and tolerance $\epsilon = 1.0E-10$. The errors are shown for initial conditions (3.3.2)-(3.3.4). 137

3.9 Error plot of second-order scheme using Crank-Nicolson with mesh 320×320 and tolerance $\epsilon = 1.0E-10$. The errors are shown for initial conditions (3.3.2)-(3.3.4). 138

3.10 Error plot of second-order scheme using BDF2 with mesh 320×320 and tolerance $\epsilon = 1.0E-10$. The errors are shown for initial conditions (3.3.2)-(3.3.4). 139

3.11 The rank evolution using backward Euler with mesh 640×640 and tolerance $\epsilon = 1.0E-10$. The rank is shown for all three initial conditions (3.3.2)-(3.3.4). (left) Time-stepping size $\Delta t = 0.05$; (right) Time-stepping size $\Delta t = 3\Delta x$. 140

3.12 The rank evolution using stiffly-accurate DIRK2 with mesh 640×640 and tolerance $\epsilon = 1.0E-10$. The rank is shown for all three initial conditions (3.3.2)-(3.3.4). (left) Time-stepping size $\Delta t = 0.05$; (right) Time-stepping size $\Delta t = 3\Delta x$. 141

3.13 The rank evolution using Crank-Nicolson with mesh 640×640 and tolerance $\epsilon = 1.0E-10$. The rank is shown for all three initial conditions (3.3.2)-(3.3.4). (left) Time-stepping size $\Delta t = 0.05$; (right) Time-stepping size $\Delta t = 3\Delta x$. 141

3.14 The rank evolution using BDF2 with mesh 640×640 and tolerance $\epsilon = 1.0E-10$. The rank is shown for all three initial conditions (3.3.2)-(3.3.4). (left) Time-stepping size $\Delta t = 0.05$; (right) Time-stepping size $\Delta t = 3\Delta x$. 142

4.1 The phase space domain of the 1D2V model. The direction of the parallel velocity component coincides with the spatial direction. . . 153

4.2 Mesh 400×400, time-stepping size $\Delta t = 0.3$, Stenger quadrature $K = 200$, tolerances $\epsilon = 1.0E-05$ and $\epsilon = 1.0E-02$. Figure (a): initial distribution of two Maxwellians defined by Table 4.1. Figure (b): rank evolution. Figure (c): L^1 error, $\|f_\alpha - f_M\|_1$. Figure (d): discrete Kullback relative entropy, $\mathcal{H}_\Delta(f_\alpha, f_M)$. 174

4.3 Time-stepping size $\Delta t = 0.3$, Stenger quadrature $K = 200$, tolerance $\epsilon = 1.0E-05$, meshes 200×200 and 400×400. Figure (a): equilibrium distribution function, f_M. Figure (b): rank evolution. Figure (c): L^1 error, $\|f_\alpha - f_M\|_1$. Figure (d): discrete Kullback relative entropy, $\mathcal{H}_\Delta(f_\alpha, f_M)$. 175

4.4 Figure (a): initial shock (4.4.3) for the 1D2V VLBFP test. The smoothed number density, drift velocity, ion temperature and electron temperature profiles are shown. Spatial mesh $N_x = 80$. Figure (b): average nodal rank. 176

4.5 Time-stepping size $\Delta t^0 = 5 \times 10^{-3}$ and $\Delta t^k = 0.3$, $k = 1, ..., N_t$, Stenger quadrature $K = 15$, tolerance $\epsilon = 1.0E-05$, spatial mesh $N_x = 80$, velocity mesh 120×120. Figure (a): total mass (4.2.8a). Figure (b): total momentum (4.2.8b). Figure (c): total energy (4.2.8c). Figure (d): total electron pressure $p_e(x,t) = (n_e T_e)(x,t)$. . 178

4.6 Various snapshots of the numerical solution to the 1D2V VLBFP equation with initial distribution 4.4.3. Time-stepping size $\Delta t^0 = 5 \times 10^{-3}$ and $\Delta t^k = 0.3$, $k = 1, ..., N_t$, Stenger quadrature $K = 15$, tolerance $\epsilon = 1.0E-05$, spatial mesh $N_x = 80$, velocity mesh 120×120. Times: 25, 50, 75, 100, 125, 150. 179

A.1 The space-time region ${}_1\Omega_j$ for IMEX(2,2,2). 195

C.1 The second-order scheme with BDF2. 201

H.1 Elsevier's copyright permission for authors (screenshot 1). https://www.elsevier.com/about/policies/copyright (link here). . . 218

H.2 Elsevier's copyright permission for authors (screenshot 2). https://www.elsevier.com/about/policies/copyright/permissions (link here). 218

ABSTRACT

Many physical phenomena can be described by nonlinear partial differential equations (PDEs). Yet, analytic solutions are oftentimes unavailable, and lab experiments can be time consuming and expensive; thus motivating the need for numerical solutions. Constructing low-storage, efficient and robust algorithms for solving PDEs comes with several computational challenges. First, classical discretization methods suffer from the *curse of dimensionality*, that is, the computational cost grows exponentially as the number of dimensions increases. Second, shock formations and sharp gradient structures that develop in certain PDEs of interest are challenging to capture due to the *Gibbs phenomenon*, that is, steep oscillations that occur near discontinuities. And third, satisfying physical properties such as conservation, equilibrium preservation and relative entropy dissipation is desired at the discrete level to avoid nonphysical behaviors. The goal of this dissertation is to develop efficient and robust algorithms for solving high-dimensional PDEs in fluid and kinetic applications.

The first contribution of this dissertation is the development of a new Eulerian-Lagrangian Runge-Kutta finite volume (EL-RK-FV) method for solving convection and convection-diffusion problems. Eulerian-Lagrangian and semi-Lagrangian methods have become popular ways to solve hyperbolic conservation laws due to their ability to allow large time-stepping sizes. The proposed scheme is formulated by integrating the PDE on a space-time region partitioned by approximations of the characteristics determined from the Rankine-Hugoniot jump condition; and then rewriting the time-integral form into a time-differential form to allow application of Runge-Kutta methods via the method of lines. The scheme can be viewed as a generalization of the standard Runge-Kutta finite volume (RK-FV) scheme for which the space-time region is partitioned by approximate characteristics with zero velocity. The high-order spatial reconstruction is

achieved using the recently developed weighted essentially non-oscillatory scheme with adaptive order (WENO-AO); and the high-order temporal accuracy is achieved by explicit Runge-Kutta methods for convection equations and implicit-explicit (IMEX) Runge-Kutta methods for convection-diffusion equations. The algorithm extends to higher dimensions via dimensional splitting. Numerical experiments demonstrate the algorithm's robustness, high-order accuracy, and ability to handle extra large time-steps.

The second contribution of this dissertation is the development of an implicit low-rank method for solving diffusion equations. Low-rank tensor methods have become a popular way to efficiently solve and store solutions of high-dimensional time-dependent PDEs. By taking advantage of low-rank structures inherent to some solutions of time-dependent problems, low-rank tensor methods reduce the storage requirements and hence the computational complexities. This helps avoid the curse of dimensionality. However, some PDEs of interest contain stiff operators that require implicit time integrators for reasonable computational efficiency. The proposed scheme is formulated by using traditional implicit time integrators to evolve the solution; decomposing the solution into one-dimensional time-dependent bases connected by time-dependent coefficients; and updating the one-dimensional bases one target direction at a time by freezing the solution in all other non-target directions. Projecting the equation onto the updated bases, the time-dependent coefficients are then updated. The updated solution is truncated using a basis removal procedure based on the singular value decomposition. The backward Euler method is used for the first-order scheme. Second-order schemes are also presented using second-order stiffly-accurate diagonally implicit Runge-Kutta methods, Crank-Nicolson method, and second-order backward differentiation formula. Numerical experiments demonstrate the algorithm's convergence and computational efficiency from enforcing the low-rank structure in solutions.

The third contribution of this dissertation is the developement of a low-rank tensor method for solving the 1D2V Vlasov-Fokker-Planck (VFP) equation. The Vlasov-Fokker-Planck and Fokker-Planck type equations are kinetic models that are used to

describe weakly coupled collisional plasmas. Developing efficient numerical methods for solving such models is of growing interest due to their applications in next-generation designs of field reversed configuration (FRC) thrusters and intertial confinement fusion (ICF) capsules. We consider a hybrid kinetic-ion fluid-electron model in which the ions are kinetically treated using the VFP equation and the electrons are treated using a fluid model. The proposed scheme is formulated by assuming a low-rank tensor structure of the solution in velocity space; discretizing the collision operator with the robust structure-preserving Chang-Cooper (SPCC) method; updating the solution by solving linear systems of tensor product structure; and truncating the solution using a basis removal procedure based on the singular value decomposition. Numerical experiments demonstrate the scheme's structure-preserving qualities, robustness and computational efficiency from enforcing the low-rank structure in solutions.

Chapter 1

INTRODUCTION

The primary focus of the algorithms herein is on efficiency, and the secondary focus is on structure preservation. Our approach to enforcing efficiency is twofold: (*Efficiency in time*) develop high-order accurate methods that allow very large time-stepping sizes, and (*Efficiency in space*) develop efficient methods that take advantage of a solution's low-rank structure. Structure preservation is enforced when appropriate by utilizing well-developed robust schemes. Before overviewing the three algorithms to be presented, we first outline the differences between fluid and kinetic models, and the differences between Eulerian, Lagrangian and Eulerian-Lagrangian frameworks.

1.1 Kinetic models vs fluid models

In this dissertation we are primarily concerned with finding numerical solutions to PDEs common in fluid and kinetic dynamics. That said, there are two descriptions in which a system can be treated: either kinetically or as a fluid [87, 114, 132]. A **kinetic description** comes from a molecular point of view in which particles are described by a *probability distribution function* $f(\mathbf{x}, \mathbf{v}, t) : \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}_+ \to \mathbb{R}_+$. The distribution function represents the probability of finding a certain particle at spatial position $\mathbf{x} \in \mathbb{R}^3$ with velocity $\mathbf{v} \in \mathbb{R}^3$ at time $t \in \mathbb{R}_+$. A **fluid description** comes from a continuum point of view in which most of the particles in a many-particle system are associated most of the time with a particular fluid element to which Newton's laws of motion apply. Depending on the application, one can derive the fluid equations either from a first principles kinetic model (common treatment in most physics textbooks) or by investigating Newton's laws of motion over a fluid element/control volume (common treatment in most engineering textbooks). There are subtle differences between the

two derivations, and sometimes one approach might be more natural than the other. This section will only derive a fluid model from a kinetic model, and we refer the reader to [114] for the other derivation.

1.1.1 Starting with a kinetic model

We start with a kinetic model describing the distribution function f. For the sake of demonstration, we assume a collision-dominated plasma for which both kinetic and fluid treatments might be suitable. A plasma can only be accurately treated as a fluid under high collisionality between particles; even then, a kinetic treatment might still be preferred. A common first principles model is the kinetic equation

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_{\mathbf{v}} f = C(f), \tag{1.1.1}$$

where q is the particle charge, m is the particle mass, $\mathbf{E}$ and $\mathbf{B}$ are respectively the electric and magnetic fields, and $C(f)$ is in general a nonlinear functional of f. When describing a collision-dominated plasma, $C(f)$ is usually taken to be the Boltzmann collision operator [31, 87, 132] or a Fokker-Planck type collision operator [1, 91, 93, 132]; additional theoretical treatments of the Boltzman and Fokker-Planck operators can be found in [9, 50, 53, 54, 78, 134, 170]. Equation (1.1.1) is known as the Vlasov equation when $C(f) \equiv 0$. Describing equation (1.1.1) and the various collision operators is not the purpose of this chapter, so we leave their discussion to the references mentioned.

1.1.2 Deriving a fluid model from a kinetic model

Under the high collisionality assumption, one could take moments of equation (1.1.1) and derive the *moment equations*. By "taking moments" we mean multiplying equation (1.1.1) by powers of velocity $\mathbf{v}$ (or some change of variables $\mathbf{w} = \mathbf{v} - \mathbf{u}$) and integrating over velocity space $\mathbb{R}^3$. The result is a system of equations only dependent on physical space and time.

The zeroth-order moment $\langle \cdot, 1 \rangle_{\mathbf{v}}$ corresponds to mass conservation, the first-order moment $\langle \cdot, \mathbf{v} \rangle_{\mathbf{v}}$ corresponds to momentum conservation, and the second-order

moment $\langle \cdot, \mathbf{vv} \rangle_{\mathbf{v}}$ corresponds to energy conservation (for the plasma stress tensor); $\mathbf{vv}$ denotes the *dyadic/outer/tensor product* of $\mathbf{v}$ and $\mathbf{v}$. For brevity, we only present the first two moments for the Vlasov equation (i.e., $C(f) \equiv 0$). After computing the zeroth- and first-order moments of the Vlasov equation for a single (ion) species α, one gets the moment equations [87, 132]

$$\frac{\partial n_\alpha}{\partial t} + \nabla_{\mathbf{x}} \cdot (n_\alpha \mathbf{u}_\alpha) = 0, \tag{1.1.2a}$$

$$\frac{\partial}{\partial t} (m_\alpha n_\alpha \mathbf{u}_\alpha) + \nabla_{\mathbf{x}} \cdot (m_\alpha n_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha + \mathbf{P}_\alpha) = q_\alpha n_\alpha (\mathbf{E} + \mathbf{u}_\alpha \times \mathbf{B}), \tag{1.1.2b}$$

where the number density and bulk velocity are respectively

$$n_\alpha(\mathbf{x}, t) = \iiint_{\mathbb{R}^3} f_\alpha(\mathbf{x}, \mathbf{v}, t) d^3v, \qquad \mathbf{u}_\alpha(\mathbf{x}, t) = \frac{1}{n_\alpha(\mathbf{x}, t)} \iiint_{\mathbb{R}^3} \mathbf{v} f_\alpha(\mathbf{x}, \mathbf{v}, t) d^3v,$$

and the thermal pressure tensor is

$$\mathbf{P}_\alpha = \iiint_{\mathbb{R}^3} m_\alpha (\mathbf{v} - \mathbf{u}_\alpha)(\mathbf{v} - \mathbf{u}_\alpha) f_\alpha(\mathbf{x}, \mathbf{v}, t) d^3v.$$

$$\Updownarrow$$

$$n_\alpha \mathbf{U}_\alpha \coloneqq m_\alpha n_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha + \mathbf{P}_\alpha = \iiint_{\mathbb{R}^3} m_\alpha \mathbf{vv} f_\alpha(\mathbf{x}, \mathbf{v}, t) d^3v.$$

Multiplying by the mass m_α and summing moment equations (1.1.2) over all the charged species, one gets the conservation of mass and momentum [87, 132]

$$\frac{\partial \rho}{\partial t} + \nabla_{\mathbf{x}} \cdot (\rho \mathbf{u}) = 0, \tag{1.1.3a}$$

$$\frac{\partial}{\partial t} (\rho \mathbf{u}) + \nabla_{\mathbf{x}} \cdot (\rho \mathbf{uu} + \mathbf{P}) = \rho_c \mathbf{E} + \mathbf{J} \times \mathbf{B}, \tag{1.1.3b}$$

where the mass density and charge density are respectively

$$\rho = \sum_\alpha (m_\alpha n_\alpha), \qquad \rho_c = \sum_\alpha (q_\alpha n_\alpha),$$

the bulk velocity and momentum are respectively

$$\mathbf{u} = \frac{1}{\rho}\sum_{\alpha}(m_\alpha n_\alpha \mathbf{u}_\alpha), \qquad \rho\mathbf{u} = \sum_{\alpha}(m_\alpha n_\alpha \mathbf{u}_\alpha),$$

the electric current density is

$$\mathbf{J} = \sum_{\alpha}(q_\alpha n_\alpha \mathbf{u}_\alpha),$$

and the thermal pressure tensor is

$$\mathbf{P} = \sum_{\alpha}\iiint_{\mathbb{R}^3} m_\alpha(\mathbf{v}-\mathbf{u})(\mathbf{v}-\mathbf{u})f_\alpha(\mathbf{x},\mathbf{v},t)d^3v.$$

$$\Updownarrow$$

$$\rho\mathbf{U} \coloneqq \rho\mathbf{u}\mathbf{u} + \mathbf{P} = \sum_{\alpha}\iiint_{\mathbb{R}^3} m_\alpha \mathbf{v}\mathbf{v} f_\alpha(\mathbf{x},\mathbf{v},t)d^3v.$$

Further deriving up to the Nth-order moment similar to equations (1.1.2),(1.1.3) yields the **moment system/fluid model**. We note that a *fluid* is a many-particle system for which the first one, two or three moments of the distribution function form a sufficient and self-contained/closed description [132]. Observe that system (1.1.2) is in general not closed unless we know the stress tensor, $\mathbf{P}$. In many applications, the second-order moment is expressed as an equation of state in terms of the lower-order moments.

1.1.3 The moment closure problem

As seen in equations (1.1.2), the system is not closed. That is, it is an underdetermined system. The flux term in a given moment equation requires the conserved quantity from the next moment equation. For example, the continuity equation for mass ρ requires momentum $\rho\mathbf{u}$, which requires $\mathbf{P}$ in the momentum conservation equation, which requires more quantities the more moments one takes. This is known as the *moment closure problem*, and constructing a general way to close the moment system

is an open problem. Using a fluid model requires closure of the moment system (up to the Nth-order moment). Depending on the system, the flux term in the highest-order moment equation must be either known/assumed or determined by some other equation(s).

We mentioned earlier that there are slight differences between deriving the fluid equations from a kinetic model versus deriving them from Newton's laws of motion over a control volume. One difference is that in the continuum view other physical assumptions are often made. These physical assumptions inherently close the moment system. For example, the Navier-Stokes equation for linear momentum assumes a Newtonian fluid in which the viscous stresses and deformation/strain-rate are linearly dependent [114]. These physical assumptions are mathematically justified by the Chapman-Enskog theory, in which the transport terms are expressed in terms of molecular parameters [37, 132]. Thus, the Chapman-Enskog theory is a (classical) method that can be used to close moment systems derived from kinetic models. The takeaway is that the Chapman-Enskog theory bridges the two approaches for deriving the fluid equations.

Kinetic and fluid models are two hierarchical ways to represent a system. Although the moment equations are typically faster to solve numerically, there are many reasons why kinetic models are sometimes preferred. Kinetic models better describe the system, particularly in the plasma physics setting, since the distribution function describes the behavior of the particles; the fluid system only computes the macroscopic quantities. Furthermore, the fluid equations in the plasma physics setting tend to be better suited for Maxwellian distribution functions; the distribution functions in plasma physics settings are not always Maxwellian. And lastly, as mentioned earlier closing the moment system in a general way is an open problem; to close the system, certain physical assumptions might be necessary.

1.2 Eulerian, Lagrangian and Eulerian-Lagrangian frameworks

There are two standard frameworks in which we describe the dynamics of a system when solving partial differential equations: Eulerian and Lagrangian. Each one can be described by a physical motivation. Consider an arbitrary fluid flow as shown in Figure 1.1. There are two ways we can intuitively track the fluid flow. We could fix some control volume in space and observe how the flow evolves within the control volume (i.e., the *Eulerian framework*). Or, we could focus on tracking the movements of a fixed number of fluid particles (i.e., the *Lagrangian framework*). Notice in Figure 1.1 that the fluid particle being tracked in the Lagrangian framework follows a single pathline.

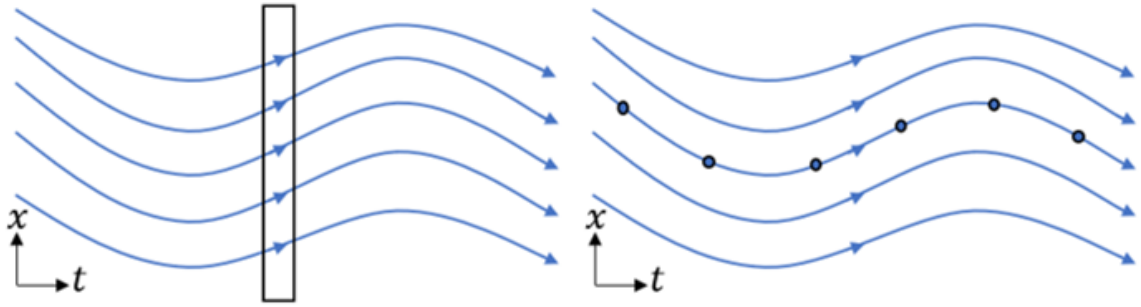


Figure 1.1: (left) The Eulerian framework; (right) The Lagrangian framework.

These physical viewpoints motivate and influence the design of numerical algorithms. For simplicity, consider solving for a solution $u(x,t)$, where $x \in [0,1]$ is space and $t \in [0,T_f]$ is time. Extending these descriptions to numerical frameworks, one could evolve the PDE on a *fixed* spatial mesh (Eulerian framework), or move some fixed spatial points over the paths on which the solution remains constant (Lagrangian framework).

1.2.1 Eulerian framework

Eulerian methods update the solution over a stationary mesh. The main advantage of Eulerian methods is that the fixed mesh allows high-order spatial resolution

methods to be utilized [43, 117, 120, 157]; specific examples of high-order spatial resolution methods are discussed in Chapter 2. These methods approximate the solution in space to high-order accuracy while also controlling spurious oscillations caused by Gibbs' phenomenon (i.e., steep oscillations observed when numerically approximating sharp gradients and jump discontinuities). Sharp gradients and jump discontinuities frequently occur in fluid and plasma simulations. Hence, there is a great desire to develop algorithms that can accommodate high-order spatial resolution methods.

Discretizing the solution $u(x,t)$ in both space and time, assume a spatial mesh of N_x nodes

$$0 = x_1 < x_2 < ... < x_{N_x-1} < x_{N_x} = 1,$$

and assume $N_t + 1$ successive incremental time-steps

$$0 = t^0 < t^1 < ... < t^{N_t-1} < t^{N_t} = T_f.$$

There are many numerical methods to obtain the updated solution $u_j^{n+1} \approx u(x_j, t^{n+1})$ from the current solution $u_j^n \approx u(x_j, t^n)$. Take for instance the Lax-Wendroff method when solving $u_t + u_x = 0$, $u(x,0) = u_0(x)$,

$$u_j^{n+1} = u_j^n - \Delta t \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + \frac{\Delta t^2}{2} \frac{u_{j-1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}, \tag{1.2.1}$$

in which u_j^{n+1} is dependent on u_{j-1}^n, u_j^n and u_{j+1}^n. We call $[x_{j-1}, x_{j+1}]$ the **numerical domain of dependence** since the updated solution at time t^{n+1} depends on the solution at time t^n within the domain $[x_{j-1}, x_{j+1}]$. Furthermore, the space-time curves along which the solution remains constant are called **characteristics**. As seen in Fig. 1.2, we can trace the characteristics backwards in time from time t^{n+1} to time t^n to obtain the **physical domain of dependence**. This leads us to the well-known CFL condition, stated below [118].

Theorem 1.1 (CFL condition [118])**.** A necessary condition for numerical stability is

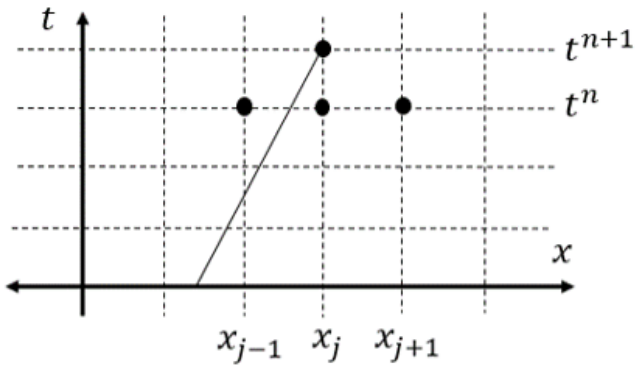


Figure 1.2: Tracing the characteristics backwards in time for the Lax-Wendroff method.

that the numerical domain of dependence must contain the physical domain of dependence.

Notice that depending on how fast the fluid particles are moving, the time-stepping size Δt might need to be much smaller than the mesh size Δx in order to satisfy the CFL condition. This is unideal since a smaller time-stepping size means more time-steps need to be taken, and hence we would require a greater computational run-time. Although Eulerian methods have the benefit of incorporating high-order spatial resolution methods that approximate the solution to high-order accuracy and control spurious oscillations, their efficiency is hindered by the CFL condition.

1.2.2 Lagrangian framework

In kinetic simulations, Lagrangian methods update the solution by following a finite number of points/particles exactly along the characteristics. Unlike the Eulerian methods, Lagrangian methods do not suffer from the CFL condition since the numerical domain of dependence exactly matches the physical domain of dependence. Popular Lagrangian methods used in kinetic simulations (e.g., particle in cell methods [68, 83, 95, 173]) have a fixed number of particles follow trajectories that drive the system. Following these trajectories could cause a higher density of particles. As such, a re-meshing step that re-positions the mesh/particles away from the shock is often required, increasing the computational complexity. Furthermore, Lagrangian methods are known

to have a significant amount of noise. Despite these challenges, Lagrangian methods remain a very popular choice primarily due to their computational efficiency in higher dimensions; in higher dimensions, the computational cost of a re-meshing step is very small relative to the overall complexity.

1.2.3 Eulerian-Lagrangian framework

Eulerian-Lagrangian and semi-Lagrangian methods reap the benefits of both Eulerian and Lagrangian methods by combining their frameworks [39, 55, 111, 133, 142, 144]. Generally speaking, an Eulerian-Lagrangian method works on a stationary background grid so that high-order spatial resolutions can be used (the Eulerian part), and characteristics stemming from the cell boundaries are then traced backward/forward in time to relax the CFL constraint (the Lagrangian part). As seen in Fig. 1.3, we can approximate the characteristics using linear space-time curves; or we can trace the characteristics exactly. Methods that trace the characteristics exactly are called semi-Lagrangian methods. Note that the Eulerian framework is the special case where the approximate characteristics are vertical space-time lines.

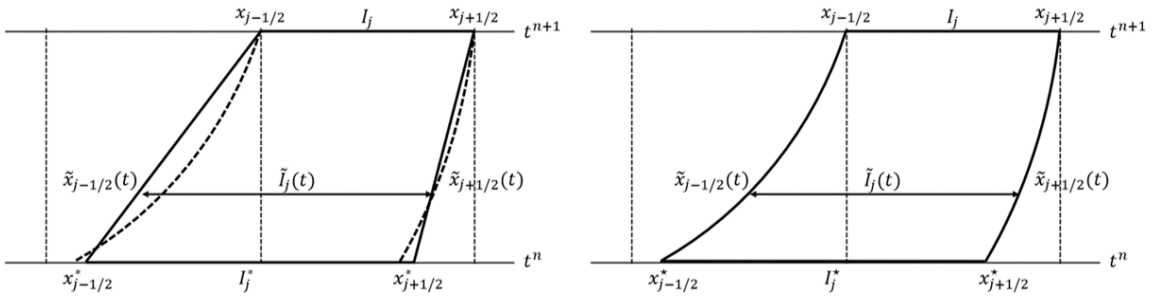


Figure 1.3: (left) Approximating the characteristics in the Eulerian-Lagrangian framework; (right) Following the characteristics exactly in the semi-Lagrangian framework.

After tracing the characteristics backward in time, the solution is computed at the traceback points $x^*_{j+\frac{1}{2}}$. Since the solution remains constant along characteristics, the solution at time t^{n+1} is easily obtained by evolving back up the approximate

characteristics. This procedure is repeated at each time step. By evolving the solution along approximate characteristics in a similar fashion to Lagrangian methods, the CFL condition is relaxed. As such, the largest allowable time-stepping size is significantly increased. Meanwhile, we can also utilize high-order spatial resolution methods because we are working on a fixed background mesh at time t^{n+1}, as seen in Fig. 1.3.

1.3 The proposed schemes

The three schemes that make up this dissertation are outlined. The first scheme emphasizes efficiency in time: a robust and high-order accurate Eulerian-Lagrangian method for convection-diffusion equations. The second scheme emphasizes efficiency in space: an Eulerian method for diffusion equations that takes advantage of low-rank structures in solutions. The third scheme emphasizes efficiency in space and structure preservation: an Eulerian method for the Vlasov-Fokker-Planck equation that takes advantage of low-rank structures in solutions.

1.3.1 An Eulerian-Lagrangian scheme for convection-diffusion equations

Motivation and existing methods

As discussed in Section 1.2.3, Eulerian-Lagrangian (EL) and semi-Lagrangian (SL) methods are attractive mostly due to their ability to allow large time steps. In particular, EL and SL schemes [30, 152, 179] have proven to be computationally effective when solving hyperbolic problems because of their ability to employ high spatial resolution schemes while admitting very large CFL with numerical stability. In fact, this is what has led to their recent popularity in the plasma physics community [39, 111, 142, 143, 144, 151]. Such methods have been developed in a wide variety of frameworks: discontinuous Galerkin [28, 55, 144, 151], finite difference [29, 39, 100, 121, 142, 143, 178], and finite volume [2, 15, 38, 46, 72, 98, 99, 133].

Although the SL and EL frameworks are similar in spirit, the SL framework assumes exact characteristic tracing and hence poses difficulties when considering nonlinear problems. Two other methods similar to EL and SL methods are the arbitrary Lagrangian-Eulerian (ALE) methods, where an arbitrary mesh velocity not necessarily aligned with the fluid velocity is defined [21, 22, 23, 24, 58, 94, 141], and moving mesh methods, where the PDE is first evolved in time and then followed by some mesh-redistribution procedure [122, 123, 129, 152, 161, 168]. The main difference between these two methods and the previously mentioned methods is that they move the mesh adaptively to focus resolving the solution around sharp transitions. By contrast, EL and SL methods evolve the equation by following characteristics.

Our goal was to develop a new high-order EL method in the finite volume framework using method-of-lines (MOL) RK time discretizations. Finite volume methods are attractive since they are naturally mass conservative, easy to physically interpret, and modifiable for nonuniform grids. Similar to the recent developments made by Huang, Arbogast, and Qiu [98, 99], we use approximate characteristics to define a traceback space-time region, and then use WENO reconstructions to evaluate the modified flux. Huang and Arbogast developed a re-averaging technique that allows high-order reconstruction of the solution at arbitrary points by applying a standard WENO scheme [43, 157] over a uniform reconstruction grid that is defined separately. They then used a natural continuous extension [183] of Runge-Kutta schemes, which requires the solution at several Gaussian nodes of the interval $[t^n, t^{n+1}]$, to evolve the solution along the approximate characteristics.

The proposed method

The novelty of our proposed method is twofold: (1) the partition of space-time regions formed by linear approximations of the characteristic curves, and (2) integrating the differential equation over the partitioned space-time regions, followed by rewriting the space-time integral form of the equation into a spatial-integral time-differential form. In this way, a MOL RK type method can be directly applied for time

discretization, thus avoiding the need to use a natural continuous extension of RK schemes. To be more precise, we construct linear approximate characteristics by using the Rankine-Hugoniot jump condition to define the traceback space-time regions. If the linear approximate characteristics are defined with zero velocity, then the proposed EL-RK-FV scheme reduces to the standard RK-FV method [157]. Whereas, when linear space-time curves adequately approximate the exact characteristics, a large time stepping size is still permitted. We use WENO-AO to perform a solution remapping of the uniform cell averages onto the possibly nonuniform traceback cells. As discussed in Section 2.1.1.2, the recently developed WENO-AO schemes [8, 11, 12] are robust and guarantee the existence of the linear weights at arbitrary points. We note that Chen, et al. used WENO-AO schemes in the SL framework [39], and Huang and Arbogast have recently used WENO-AO schemes in the Eulerian framework [7, 8]. RK methods are used to evolve the MOL system along the approximate characteristics. Explicit RK methods, such as the strong stability-preserving (SSP) RK methods [77], are used for convection equations, and implicit-explicit (IMEX) RK methods [10, 44, 92] are used for convection-diffusion equations. In the latter case, the non-stiff convective term is treated explicitly and the stiff diffusive term is treated implicitly. Dimensional splitting is used to extend the one-dimensional algorithm to solve multi-dimensional problems. The proposed method is high-order accurate, capable of resolving discontinuities without oscillations, mass conservative, and stable with large time stepping sizes.

1.3.2 A low-rank Eulerian scheme for diffusion equations

Motivation and existing methods

The high-dimensionality of kinetic models and physical systems poses a major challenge in computing numerical solutions. As the number of dimensions increases, classical discretization methods such as finite differences and finite elements experience exponential growth in computational cost and the degrees of freedom; this is commonly

known as the *curse of dimensionality*. To overcome the curse of dimensionality in solving high-dimensional equations, recent developments have taken advantage of low-rank structures inherent to some of these PDEs. For example, the solution to the multi-dimensional heat equation is low-rank in the sense that its Fourier coefficients quickly decay as the wave number increases. Even if the analytic solution is of high-rank (e.g., the Fourier series solution of the multi-dimensional heat equation), it can be approximated by a low-rank function (e.g., a truncated Fourier series solution). Townsend, et al. investigated why and when many datasets (e.g., in the form of matrices and tensors) that occur in real world applications are compressible and have low-rank structure [154, 175]. Given that many time-dependent systems of interest have low-rank solutions, particularly kinetic models, developing low-rank numerical methods could lead to significant computational savings.

Over the past couple of decades, significant progress has been made in developing low-rank tensor methods for solving time-dependent problems [41, 82, 112]. Numerical tensor decompositions offer great flexibility in reducing the storage and computational complexities of numerical methods. Popular tensor decompositions include the CANDECOMP/PARAFAC (CP) decomposition [106, 109, 112], tensor train (TT) decomposition [135, 136], and Tucker and hierarchical Tucker (HT) decompositions [48, 80, 89, 109, 112, 174]. Several such works have been developed and applied to nonlinear kinetic models: low-rank semi-Lagrangian method in the TT format [111], low-rank basis removal method in the HT format that conserves physical invariants with a projection-based scheme [85], low-rank CP method for Hamiltonian formulations [62], dynamical tensor approximations based on a functional tensor decomposition [52], and dynamical low-rank methods [45, 64, 65, 105, 107, 125].

Although there has been great progress in developing explicit low-rank methods for time-dependent problems, there is also a pressing need for high-accuracy implicit low-rank methods. Several kinetic models in plasma physics of interest to the national laboratories have stiff operators that describe the collisional interactions between particles, e.g., Fokker-Planck type equations (see Section 1.1). Due to the stiffness of these

collision operators, implicit time integrators are preferred over explicit time integrators. However, extending low-rank methods (e.g., DLR methods [32, 33, 63, 125]) to high-accuracy implicit methods is challenging since one needs to seek low-rank solutions in an implicit setting. Recently, Rodgers and Venturi developed an implicit rank-adaptive method in which the solution after each time-step is truncated by projecting onto a low-dimensional tensor manifold [150]. The authors refer to these algorithms as step-truncation algorithms [149], based on performing a truncation operation onto a tensor manifold after performing a single time-step with a conventional time integrator.

The proposed method

We propose a novel implicit low-rank method for solving two-dimensional diffusion equations. We represent the time-dependent solution in a low-rank framework. Similar in spirit to the unconventional integrator [33] and step-truncation methods [85, 86, 150], we strategically evolve the low-rank decomposition of the solution based on traditional implicit time-integrators. In particular, the solution is decomposed into one-dimensional time-dependent bases connected by time-dependent coefficients. We evolve these one-dimensional bases in a dimension-by-dimension fashion. The target dimension basis is updated by first freezing and correspondingly projecting the solution in all the non-target dimensions. Once the bases from all dimensions are updated, we then evolve the coefficients be a projection onto the subspace spanned by the updated bases in all dimensions. Finally, a SVD type truncation is applied to further compress the solution for optimal computational efficiency.

The first-order scheme can be viewed as an equivalent formulation to the unconventional DLR method [33]. However, the second-order schemes differ since we follow a multistage methodology in which the one-dimensional bases from previous stages are used to construct approximate bases that span rich enough vector spaces. The proposed method maintains low-rank structure in solutions and is extendable to higher-order implicit time integrators.

1.3.3 A low-rank Eulerian scheme for the Vlasov-Fokker-Planck equation

Motivation and existing methods

In a system of weakly coupled collisional plasmas, the Vlasov-Fokker-Planck (VFP) equation describes the evolution of each species' distribution function. When combined with Maxwell's equations that evolve the electromagnetic fields, the Vlasov-Fokker-Planck-Maxwell system provides a first-principles model for such plasmas. Such models have a wide array of applications that describe laboratory, space and astrophysical plasmas (e.g., field reversed configuration (FRC) thrusters [110], inertial confinement fusion (ICF) capsules [148, 164, 169]). As next-generation designs in these settings continue to progress and evolve, new concepts will heavily rely on a combination of forward-predictive modeling and experimental validations. Given the time-consuming and expensive nature of experimental iterations, numerical simulations could be used to accelerate and facilitate the design iteration procedure. Since kinetic models describing these plasmas are high-dimensional (up to six-dimensional phase space; three dimensions in space and three dimensions in velocity space – 3D3V), designing efficient algorithms is of the utmost importance. Moreover, enforcing physical properties such as conservation and structure preservation is highly desired to ensure physically correct solutions.

The VFP, Vlasov, and Fokker-Planck type equations have a rich computational history and have been solved in the Eulerian [36, 67, 71, 103, 104, 116, 139, 164, 165, 166, 167], Lagrangian [17, 47, 51, 70, 181], and semi-Lagrangian [56, 111, 142, 151, 156] frameworks. More recently, these kinetic models have also been solved using tensor and low-rank methods [45, 57, 64, 65, 66, 85, 111], and moving-mesh type methods based on thermal velocity-dependent mappings [69, 102, 115, 162]. As mentioned, conserving physical invariants [1, 6, 13, 64, 85, 155, 162, 163] and preserving physical structures [26, 27, 36, 49, 63, 96, 97, 103, 104, 139, 144, 151] (e.g., equilibrium-preserving, positivity-preserving, asymptotic-preserving, relative entropy dissipative) are critical for computing physically correct solutions, or rather, avoiding non-physical solutions.

Our goal is to develop a low-rank method for solving the VFP equation in 1D2V phase space that has structure-preserving qualities. We take most inspiration from the very popular Chang-Cooper discretization of the Fokker-Planck operator [26, 36, 131]. By discretizing the flux with a weighted sum of neighboring point values using specially constructed weights, Chang and Cooper obtain a method that is positivity-preserving, particle-conserving and equilibrium-conserving. In [139], Pareschi and Zanella generalize the Chang-Cooper discretization to more general Fokker-Planck and gradient flow operators while also extending to arbitrary accuracy. By incorporating the structure-preserving scheme [139] into a low-rank Eulerian framework, we hope to maintain their structure-preserving qualities.

The proposed method

We aim to solve the 1D2V VFP equation in 2V cylindrical coordinates. A single ion species is assumed, and we use a hybrid model in which the ions are treated kinetically with the VFP equation and electrons are treated as a fluid. Furthermore, we use the linearized Fokker-Planck operator, known as the Leonard-Bernstein-Fokker-Planck (LBFP) operator. Although linearized, the operator is still nonlinearly dependent on the solution since the coefficients are nonlinear functionals of the solution. The VFP equation is nonlinearly coupled with the fluid equation for electron energy.

Our proposed method has three major components that take inspiration from other works: (1) extending the structure-preserving Chang-Cooper (SPCC) method in [139] to discretize the 2V Fokker-Planck operator in cylindrical coordinates, (2) updating the solution using an implicit solver for linear systems of tensor product structure, and (3) a low-rank treatment similar in spirit to [84, 85, 86]. We note that the low-rank framework presented here is different from the dynamical low-rank (DLR) framework because we evolve the entire solution in one step. Whereas, DLR methods evolve the basis of each dimension separately. We discretize the solution in space and then solve a 2V system at each spatial node. First-order implicit-explicit (IMEX) Runge-Kutta scheme is used for temporal discretization.

First, the zeroth-, first- and second-order moments of the VFP equation are coupled with the fluid-electron energy equation to solve for the macroscopic quantities. These macroscopic quantities are then used to discretize the Fokker-Planck operator using the SPCC method. Second, the discretized VFP equation is set up as a linear system of tensor product structure and solved using the linear solver presented in [79]. (This is the *add basis* step since updating the solution increases the number of basis elements). Third, the SVD of the updated solution is truncated accordingly to some tolerance on the singular values. (This is the *remove basis* step since redundant basis elements are truncated out). The proposed method is low-rank and maintains structure-preserving qualities despite the SVD truncation.

1.4 Organization of the dissertation

This dissertation is organized as follows.

Chapter 2 presents the Eulerian-Lagrangian Runge-Kutta finite volume (EL-RK-FV) scheme for solving convection and convection-diffusion equations. First, the necessary technical components are reviewed. Second, the EL-RK-FV scheme is presented for pure convection problems. Third, the EL-RK-FV scheme is presented for convection-diffusion problems. Fourth, several numerical tests verify the convergence and robustness of the scheme. Fifth, conclusions and follow-up works are discussed.

Chapter 3 presents the implicit low-rank integrators for solving diffusion equations. First, the necessary technical components are reviewed. Second, the first- and second-order schemes are presented. Third, numerical tests verify the convergence and computational efficiency of the proposed method. Fourth, conclusions and follow-up works are discussed.

Chapter 4 presents the low-rank tensor scheme for solving the VFP equation. First, the necessary technical components are reviewed. Second, the discretization of the

VFP equation is presented. Third, the updating and truncating procedures for the solution are presented. Fourth, numerical results verify the low-rank structure and structure-preserving qualities of the method. Fifth, conclusions and follow-up works are discussed.

Bibliography lists the cited references herein.

Appendix A presents an illustrative second-order example of the EL-RK-FV scheme in Chapter 2.

Appendices B-C supplement Chapter 3. The appendices cover: (B) the second-order scheme with Crank-Nicolson, and (C) the second-order scheme with BDF2.

Appendices D-G supplement Chapter 4. The appendices cover: (D) the nondimensionalization of the VFP equation, (E) the derivation of the balance laws for the total mass, momentum and energy of the plasma model, (F) the Stenger quadrature nodes and weights, and (G) the quasi-Newton solver used to update the macroscopic quantities.

Appendix H presents the permissions to use the paper from which Chapter 2 is derived and taken.

Chapter 2

AN EULERIAN-LAGRANGIAN RUNGE-KUTTA FINITE VOLUME (EL-RK-FV) METHOD FOR SOLVING CONVECTION AND CONVECTION-DIFFUSION EQUATIONS

In this chapter, we are concerned with numerically solving convection-diffusion equations of the form

$$\begin{cases} u_t + \nabla \cdot \mathbf{F}(u) = \epsilon \Delta u + g(\mathbf{x}, t), & \mathbf{x} \in \mathcal{D}, \quad t > 0, \\ u(\mathbf{x}, t = 0) = u_0(\mathbf{x}), & \mathbf{x} \in \mathcal{D}, \end{cases} \tag{2.0.1}$$

where $u(\mathbf{x}, t)$ is the solution, $\mathbf{F}(u; \mathbf{x}, t)$ is the flux function, $g(\mathbf{x}, t)$ is the source term, $\mathcal{D} \subset \mathbb{R}^d$ is the spatial domain, $d \in \mathbb{N}$ is the number of dimensions, and $\epsilon \geq 0$ is the diffusion coefficient. An Eulerian-Lagrangian Runge-Kutta finite volume (EL-RK-FV) scheme utilizing weighted essentially non-oscillatory schemes with adaptive order (WENO-AO) for spatial reconstruction for solving pure convection problems was proposed by Chen, et al. [38]. Chen, et al. use the Rankine-Hugoniot jump condition to define linear approximate characteristics that go backwards in time. By projecting and evolving the solution back up along approximate characteristics, high-order accuracy is achieved and very large time-stepping sizes are allowed. Extension of the EL-RK-FV scheme to convection-diffusion problems was proposed by Nakao, et al. [133]. The proposed method was designed for one-dimensional problems of the form (2.0.1), with extension to higher-dimensional problems done via dimensional splitting.

This chapter is organized as follows. Section 2.1 reviews the technical material required to understand the proposed scheme (e.g., WENO reconstruction, Runge-Kutta methods, operator splitting, Gauss-Legendre quadrature). We then discuss the EL-RK-FV algorithm for pure convection problems in Section 2.2. In Section 2.3, we discuss

the EL-RK-FV algorithm, coupled with IMEX RK schemes, for convection-diffusion equations. Numerical performance of the EL-RK-FV algorithm is shown in Section 2.4 by applying the algorithm to several linear and nonlinear test problems. Concluding remarks on the EL-RK-FV scheme and follow-up works are made in Section 2.5. A majority of the content presented in Sections 2.2-2.5 is derived from [133].

2.1 Review of technical components

In this section, we review four technical components to the EL-RK-FV scheme: high-order spatial reconstruction, high-order time-stepping, operator splitting, and Gauss-Legendre quadrature.

2.1.1 Spatial reconstructions: WENO and WENO-AO

As discussed in Chapter 1, an Eulerian(-Lagrangian) framework allows for high-order spatial resolution methods that approximate the solution and control spurious oscillations. A very popular class of high-order spatial resolution methods that we use in this scheme is the weighted essentially non-oscillatory (WENO) methods. Although we only discuss WENO methods, other classes of high-order spatial resolution methods are mentioned in [117, 118, 157]. The material presented in this section is predominantly derived from [11, 43, 157].

Consider a *uniform* one-dimensional mesh over the interval $[a, b]$ consisting of $N_x + 1$ *evenly distributed* nodes (i.e., N_x cells),

$$a = x_{\frac{1}{2}} < x_{\frac{3}{2}} < ... < x_{N_x-\frac{1}{2}} < x_{N_x+\frac{1}{2}} = b.$$

Define the cells $I_j := [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ with centers $x_j = (x_{j-\frac{1}{2}} + x_{j+\frac{1}{2}})/2$ and widths $\Delta x_j = \Delta x$ for $j = 1, ..., N_x$.

The **cell average** of a function $u(x, t)$ at time t^n over cell I_j is denoted by

$$\overline{u}_j^n := \frac{1}{\Delta x} \int_{I_j} u(x, t^n) dx. \tag{2.1.1}$$

In finite volume methods, we assume knowledge of the cell averages of the solution $u(x,t)$ rather than the point values. Yet, most finite volume methods still require knowledge of the point values of the solution at the cell boundaries, $u(x_{j\pm\frac{1}{2}},t)$, for evaluating the flux difference. The process of approximating point values of a function $u(x,t)$ from its cell averages is called **spatial reconstruction**. The end product of the two WENO methods presented in the following subsections is ultimately a piecewise-polynomial that approximates $u(x,t)$ with high-order accuracy and controls spurious oscillations.

2.1.1.1 WENO5

We present the classic fifth-order WENO method, abbreviated WENO5 [43, 157]. A **p-point stencil** S_j is a set of p cells surrounding and including I_j. For example, I_j has three 3-point stencils: $\{I_{j-2}, I_{j-1}, I_j\}$, $\{I_{j-1}, I_j, I_{j+1}\}$ and $\{I_j, I_{j+1}, I_{j+2}\}$. Note that there are p possible $p-$point stencils.

Theorem 2.1 (Linear Reconstruction [157])**.** Let $\overline{u}_j$ be the cell averages of a smooth function $u(x)$ over a uniform grid consisting of cells I_j, $j = 1, 2, ..., N_x$. For each $p-$point stencil S_j, there exists a $p-1$ degree **reconstruction polynomial** $\mathcal{R}_j$ such that $\frac{1}{\Delta x}\int_{I_k}\mathcal{R}_j(x)dx = \overline{u}_k$ for all $I_k \in S_j$. Moreover, $\mathcal{R}_j(x_{j\pm\frac{1}{2}}) = u(x_{j\pm\frac{1}{2}}) + \mathcal{O}(\Delta x^p)$.

The outline of the proof is straightforward. The $p-1$ degree reconstruction polynomial has p unknown coefficients, and it must satisfy the cell averages of the solution on the p cells in the stencil. The system of p equations and p unknowns can then be solved for the coefficients of the (unique) reconstruction polynomial, and the coefficients will be in terms of the cell averages. Taylor expanding $u(x_{j\pm\frac{1}{2}})$ will then give the high-order accuracy. We refer the reader to [157] for a more detailed derivation.

The idea of WENO starts with the following observation. Consider any of the uniform cells I_j. Each of the 3-point stencils have unique third-degree reconstruction polynomials respectively denoted by $\mathcal{R}_j^{(1)}$, $\mathcal{R}_j^{(2)}$ and $\mathcal{R}_j^{(3)}$; and the centered 5-point stencil $\{I_{j-2}, I_{j-1}, I_j, I_{j+1}, I_{j+2}\}$ has a unique fifth-degree reconstruction polynomial

denoted by $\mathcal{R}_j^{(5)}$. Evaluating at the cell boundary $x_{j+\frac{1}{2}}$, there exist linear weights γ_1, γ_2 and γ_3 such that [157]

$$u(x_{j+\frac{1}{2}}) \approx \mathcal{R}_j^{(5)}(x_{j+\frac{1}{2}}) = \gamma_1 \mathcal{R}_j^{(1)}(x_{j+\frac{1}{2}}) + \gamma_2 \mathcal{R}_j^{(2)}(x_{j+\frac{1}{2}}) + \gamma_3 \mathcal{R}_j^{(3)}(x_{j+\frac{1}{2}}). \tag{2.1.2}$$

That is, the fifth-order approximation can be expressed as a linear combination of the three third-order approximations at the right cell boundary; this can be similarly done at the left cell boundary. It should be noted that the evaluation at the right cell boundary is the left limit $u^-_{j+\frac{1}{2}}$, and the evaluation at the left cell boundary is the right limit $u^+_{j-\frac{1}{2}}$. After some tedious algebra, the three third-order approximations at the right cell boundary are

$$u^{(1),-}_{j+\frac{1}{2}} = \frac{1}{3}\overline{u}_{j-2} - \frac{7}{6}\overline{u}_{j-1} + \frac{11}{6}\overline{u}_j \tag{2.1.3a}$$

$$u^{(2),-}_{j+\frac{1}{2}} = -\frac{1}{6}\overline{u}_{j-1} + \frac{5}{6}\overline{u}_j + \frac{1}{3}\overline{u}_{j+1} \tag{2.1.3b}$$

$$u^{(3),-}_{j+\frac{1}{2}} = \frac{1}{3}\overline{u}_j + \frac{5}{6}\overline{u}_{j+1} - \frac{1}{6}\overline{u}_{j+2} \tag{2.1.3c}$$

The three third-order approximations at the left cell boundary are

$$u^{(1),+}_{j-\frac{1}{2}} = -\frac{1}{6}\overline{u}_{j-2} + \frac{5}{6}\overline{u}_{j-1} + \frac{1}{3}\overline{u}_j \tag{2.1.4a}$$

$$u^{(2),+}_{j-\frac{1}{2}} = \frac{1}{3}\overline{u}_{j-1} + \frac{5}{6}\overline{u}_j - \frac{1}{6}\overline{u}_{j+1} \tag{2.1.4b}$$

$$u^{(3),+}_{j-\frac{1}{2}} = \frac{11}{6}\overline{u}_j - \frac{7}{6}\overline{u}_{j+1} + \frac{1}{3}\overline{u}_{j+2} \tag{2.1.4c}$$

The centered fifth-order approximation at the right cell boundary is

$$u^-_{j+\frac{1}{2}} = \frac{1}{30}\overline{u}_{j-2} - \frac{13}{60}\overline{u}_{j-1} + \frac{47}{60}\overline{u}_j + \frac{9}{20}\overline{u}_{j+1} - \frac{1}{20}\overline{u}_{j+2} \tag{2.1.5}$$

The coefficients of the linear reconstruction polynomials up to seventh-order accuracy can be found in Table 2.1 of [157]. The downside to linear reconstruction is

that it assumes a smooth function. In the case of non-smooth functions (e.g., Heaviside step function), we will observe spurious oscillations, as seen in Fig. 2.1. We can resolve this issue with WENO reconstruction, as seen in Fig. 2.1. Loosely speaking, **WENO reconstruction** perturbs the linear weights γ_k $(k = 1, 2, ..., p)$ for added stability and robustness. In the spirit of equation (2.1.2), WENO also takes some special weighted sum of lower-order reconstruction polynomials to achieve a higher-order reconstruction polynomial.

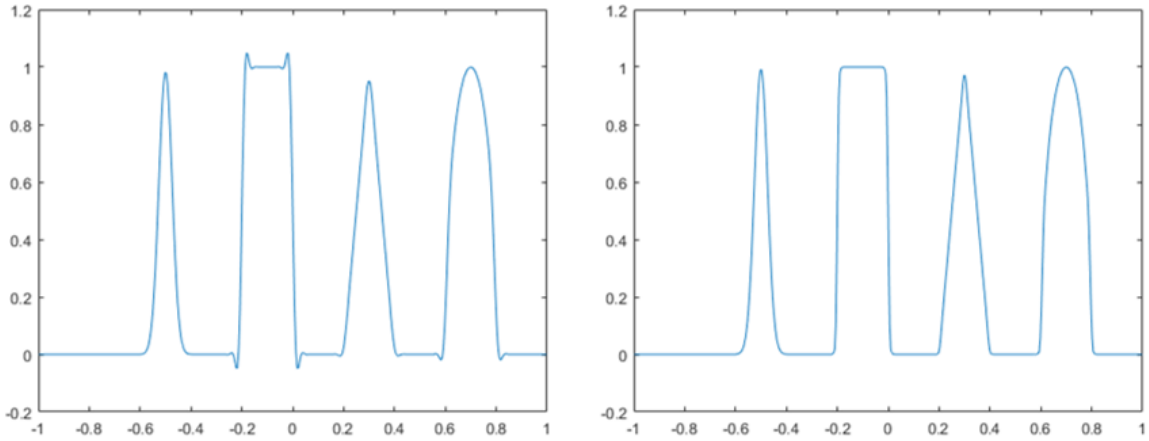


Figure 2.1: (left) Without WENO; (right) With WENO.

We present the fifth-order WENO reconstruction below and refer the reader to [157] for the general $p-$order WENO reconstruction formulas. For the purposes of this dissertation, the fine details of WENO reconstruction are not the primary focus. We refer the reader to [157] for a more rigorous construction of WENO methods. Dropping the limit sign for notational ease, the equations for **fifth-order WENO** (i.e., **WENO5**) are as follows [157]. Using the reconstructions in equation (2.1.4),

$$u_{j+\frac{1}{2}} = \omega_1 u^{(1)}_{j+\frac{1}{2}} + \omega_2 u^{(2)}_{j+\frac{1}{2}} + \omega_3 u^{(3)}_{j+\frac{1}{2}}, \tag{2.1.6}$$

where

$$\omega_k = \frac{\alpha_k}{\sum_{s=1}^{3} \alpha_s}, \qquad k = 1, 2, 3, \tag{2.1.7a}$$

$$\alpha_k = \frac{d_k}{(\epsilon + \beta_k)^2}, \qquad k = 1, 2, 3, \tag{2.1.7b}$$

with

$$d_1 = \frac{1}{10}, \qquad d_2 = \frac{3}{5}, \qquad d_3 = \frac{3}{10}, \tag{2.1.8a}$$

$$\begin{aligned}
\beta_1 &= \frac{13}{12}(\overline{u}_{j-2} - 2\overline{u}_{j-1} + \overline{u}_j)^2 + \frac{1}{4}(\overline{u}_{j-2} - 4\overline{u}_{j-1} + 3\overline{u}_j)^2, \\
\beta_2 &= \frac{13}{12}(\overline{u}_{j-1} - 2\overline{u}_j + \overline{u}_{j+1})^2 + \frac{1}{4}(\overline{u}_{j-1} - \overline{u}_{j+1})^2, \\
\beta_3 &= \frac{13}{12}(\overline{u}_j - 2\overline{u}_{j+1} + \overline{u}_{j+2})^2 + \frac{1}{4}(3\overline{u}_j - 4\overline{u}_{j+1} + \overline{u}_{j+2})^2.
\end{aligned} \tag{2.1.8b}$$

$\epsilon > 0$ is a small parameter that prevents the denominator of the smoothness indicators β_k $(k = 1, 2, 3)$ from being zero. The parameter is often set to $\epsilon = 1.0e - 06$ [43, 157]. Furthermore, the nonlinear weights ω_k $(k = 1, 2, 3)$ sum to one and are non-negative for consistency and stability.

2.1.1.2 WENO-AO(5,3)

Although the classic WENO scheme presented in the previous subsection is efficient, robust and high-order accurate, there are two important notes. First, equations (2.1.4) and (2.1.6) assume a uniform grid. The WENO scheme can be generalized to non-uniform grids, but the weights will not be the same for each cell [157]. Second, equations (2.1.4) and (2.1.6) only hold when approximating the solution at the cell boundaries. In fact, the linear weights are not guaranteed to exist nor be non-negative at arbitrary points [143, 157]. If we desire approximating the solution in a WENO-type fashion at arbitrary points within the domain, then a different WENO method is needed. One such method is WENO with adaptive order (WENO-AO) [11].

The WENO-AO method presented in [11] is constructed in a way such that the linear weights can exist at arbitrary points with a looser condition that the linear weights can now be negative. The stability and robustness come from what types of lower-order WENO methods we incorporate in our combination for a higher-order approximation.

The overarching idea of WENO-AO methods is to provide high-order accuracy for smooth solutions over a large center stencil and adaptively reduce to lower-order accuracy when the solution does not permit the high-order accuracy. This is done by creating a nonlinear hybridization between a large center stencil with high-order accuracy, and very stable lower-order WENO schemes (e.g., central WENO (CWENO) schemes [120]). Aside from the high-order accuracy and existence of linear weights at arbitrary points, the robustness of these WENO-AO schemes is particularly attractive for Eulerian-Lagrangian methods. The authors in [11] write WENO-AO(p,r) to denote an adaptive order that is at best pth order (from the large center stencil) and at worst rth order (from the stable lower order stencils). We present the formulas for WENO-AO(5,3) below and refer the reader to [11] for more general formulas and a rigorous derivation. We note that another similar method that derives the intuitive underpinnings of this WENO-AO reconstruction is found in [187]. The end product of WENO-AO methods is still a reconstruction polynomial $\mathcal{R}_j(x \in I_j)$ that we shall use for reconstruction. The equations for **WENO-AO(5,3)** are as follows [11].

The authors in [11] formulate their WENO reconstruction in the Legendre basis. The first few Legendre polynomials scaled over the domain $[-1/2, 1/2]$ are given below.

$$L_0(x) = 1, \quad L_1(x) = x, \quad L_2(x) = x^2 - \frac{1}{12}, \quad L_3(x) = x^3 - \frac{3}{20}x,$$
$$L_4(x) = x^4 - \frac{3}{14}x^2 + \frac{3}{560}, \quad L_5(x) = x^5 - \frac{5}{18}x^3 + \frac{5}{336}x.$$

Although the WENO and WENO-AO schemes presented in this subsection are over the domain $[-1/2, 1/2]$, one can easily perform a linear change of variables from/to a cell I_j to/from the interval $[-1/2, 1/2]$. The three very stable third-order WENO reconstruction polynomials are as follows [11].

$$\mathcal{R}_j^{(k),r3}(x) = a_0^{(k),r3} + a_1^{(k),r3} L_1(x) + a_2^{(k),r3} L_2(x), \qquad k = 1, 2, 3, \tag{2.1.10}$$

where

$$\begin{aligned} a_0^{(1),r3} &= \overline{u}_j, \\ a_1^{(1),r3} &= -2\overline{u}_{j-1} + \frac{1}{2}\overline{u}_{j-2} + \frac{3}{2}\overline{u}_j, \\ a_2^{(1),r3} &= \frac{1}{2}\overline{u}_{j-2} - \overline{u}_{j-1} + \frac{1}{2}\overline{u}_j, \end{aligned} \tag{2.1.11}$$

$$\begin{aligned} a_0^{(2),r3} &= \overline{u}_j, \\ a_1^{(2),r3} &= \frac{1}{2}\overline{u}_{j+1} - \frac{1}{2}\overline{u}_{j-1}, \\ a_2^{(2),r3} &= \frac{1}{2}\overline{u}_{j-1} - \overline{u}_j + \frac{1}{2}\overline{u}_{j+1}, \end{aligned} \tag{2.1.12}$$

$$\begin{aligned} a_0^{(3),r3} &= \overline{u}_j, \\ a_1^{(3),r3} &= -\frac{3}{2}\overline{u}_j + 2\overline{u}_{j+1} - \frac{1}{2}\overline{u}_{j+2}, \\ a_2^{(3),r3} &= \frac{1}{2}\overline{u}_j - \overline{u}_{j+1} + \frac{1}{2}\overline{u}_{j+2}. \end{aligned} \tag{2.1.13}$$

The smoothness indicators for each 3-point stencil are

$$\beta^{(k),r3} = \left(a_1^{(k),r3}\right)^2 + \frac{13}{3}\left(a_2^{(k),r3}\right)^2, \qquad k = 1, 2, 3. \tag{2.1.14}$$

The fifth-order WENO reconstruction polynomial using the center stencil is as follows [11].

$$\mathcal{R}_j^{r5}(x) = a_0^{r5} + a_1^{r5}L_1(x) + a_2^{r5}L_2(x) + a_3^{r5}L_3(x) + a_4^{r5}L_4(x), \tag{2.1.15}$$

where

$$
\begin{aligned}
a_0^{r5} &= \overline{u}_j, \\
a_1^{r5} &= -\frac{82}{120}\overline{u}_{j-1} + \frac{11}{120}\overline{u}_{j-2} + \frac{82}{120}\overline{u}_{j+1} - \frac{11}{120}\overline{u}_{j+2}, \\
a_2^{r5} &= \frac{40}{56}\overline{u}_{j-1} - \frac{3}{56}\overline{u}_{j-2} - \frac{74}{56}\overline{u}_j + \frac{40}{56}\overline{u}_{j+1} - \frac{3}{56}\overline{u}_{j-2}, \\
a_3^{r5} &= \frac{2}{12}\overline{u}_{j-1} - \frac{1}{12}\overline{u}_{j-2} - \frac{2}{12}\overline{u}_{j+1} + \frac{1}{12}\overline{u}_{j+2}, \\
a_4^{r5} &= -\frac{4}{24}\overline{u}_{j-1} + \frac{1}{24}\overline{u}_{j-2} + \frac{6}{24}\overline{u}_j - \frac{4}{24}\overline{u}_{j+1} + \frac{1}{24}\overline{u}_{j+2}.
\end{aligned}
\tag{2.1.16}
$$

The smoothness indicator for the center 5-point stencil is

$$
\beta^{r5} = \left(a_1^{r5} + \frac{1}{10}a_3^{r5}\right)^2 + \frac{13}{3}\left(a_2^{r5} + \frac{123}{455}a_4^{r5}\right)^2 + \frac{781}{20}\left(a_3^{r5}\right)^2 + \frac{1421461}{2275}\left(a_4^{r5}\right)^2. \tag{2.1.17}
$$

Taking a special combination of equations (2.1.10) and (2.1.15) yields the WENO-AO(5,3) reconstruction polynomial. When the smoothness indicators show that the center 5-point stencil is smooth, most if not all of the reconstruction will come from the fifth-order accurate WENO reconstruction. But, if the smoothness indicators show that the center 5-point stencil is non-smooth, then most if not all of the reconstruction will come from the very stable, third-order accurate WENO reconstructions. This non-linear hybridized high-order reconstruction is as follows.

Define two parameters, γ_{Hi} and γ_{Lo}, both less than unity. Typically, the parameters are set $\gamma_{Hi} \in [0.85, 0.95]$ and $\gamma_{Lo} \in [0.85, 0.95]$ [11]. The linear weights for the center 5-point stencil and three 3-point stencils are respectively given by

$$
\gamma^{r5} = \gamma_{Hi}, \quad \gamma_1^{r3} = \frac{(1-\gamma_{Hi})(1-\gamma_{Lo})}{2}, \quad \gamma_2^{r3} = (1-\gamma_{Hi})\gamma_{Lo}, \quad \gamma_3^{r3} = \gamma_1^{r3}. \tag{2.1.18}
$$

Note that the center 3-point stencil carries a higher linear weight than the other two 3-point stencils to help make the CWENO scheme centrally biased. Further note that

$\gamma_1^{r3} + \gamma_2^{r3} + \gamma_3^{r3} = 1 - \gamma_{Hi}$. Next, we define the parameter

$$\tau = \frac{1}{3}\left(\left|\beta^{r5} - \beta_1^{r3}\right| + \left|\beta^{r5} - \beta_2^{r3}\right| + \left|\beta^{r5} - \beta_3^{r3}\right|\right). \tag{2.1.19}$$

There are two choices for defining the un-normalized non-linear weights [19, 157]:

$$\begin{aligned} \alpha^{r5} &= \gamma^{r5}\left(1 + \frac{\tau^2}{(\beta^{r5} + \epsilon)^2}\right), \qquad \alpha_1^{r3} = \gamma_1^{r3}\left(1 + \frac{\tau^2}{\left(\beta_1^{r3} + \epsilon\right)^2}\right), \\ \alpha_2^{r3} &= \gamma_2^{r3}\left(1 + \frac{\tau^2}{\left(\beta_2^{r3} + \epsilon\right)^2}\right), \qquad \alpha_3^{r3} = \gamma_3^{r3}\left(1 + \frac{\tau^2}{\left(\beta_3^{r3} + \epsilon\right)^2}\right), \end{aligned} \tag{2.1.20a}$$

$$\begin{aligned} \alpha^{r5} &= \frac{\gamma^{r5}}{(\beta^{r5} + \epsilon)^2}, \qquad \alpha_1^{r3} = \frac{\gamma_1^{r3}}{\left(\beta_1^{r3} + \epsilon\right)^2}, \\ \alpha_2^{r3} &= \frac{\gamma_2^{r3}}{\left(\beta_2^{r3} + \epsilon\right)^2}, \qquad \alpha_3^{r3} = \frac{\gamma_3^{r3}}{\left(\beta_3^{r3} + \epsilon\right)^2}, \end{aligned} \tag{2.1.20b}$$

where $\epsilon > 0$ is small, typically $\epsilon = 1.0e - 12$. Equation (2.1.20b) is the same as in classic WENO5. The authors in [11] found equation (2.1.20a) to be a more stable option, whereas equation (2.1.20b) is a more accurate option. In practice, we often use equation (2.1.20a). The normalized non-linear weights are as follows.

$$\begin{aligned} \omega^{r5} &= \frac{\alpha^{r5}}{\alpha^{r5} + \alpha_1^{r3} + \alpha_2^{r3} + \alpha_3^{r3}}, \qquad \omega_1^{r3} = \frac{\alpha_1^{r3}}{\alpha^{r5} + \alpha_1^{r3} + \alpha_2^{r3} + \alpha_3^{r3}}, \\ \omega_2^{r3} &= \frac{\alpha_2^{r3}}{\alpha^{r5} + \alpha_1^{r3} + \alpha_2^{r3} + \alpha_3^{r3}}, \qquad \omega_3^{r3} = \frac{\alpha_3^{r3}}{\alpha^{r5} + \alpha_1^{r3} + \alpha_2^{r3} + \alpha_3^{r3}}. \end{aligned} \tag{2.1.21}$$

With this comes the WENO-AO(5,3) reconstruction polynomial, for which the linear weights exist at arbitrary points within the cell I_j (mapped to $[-1/2, 1/2]$ by a

linear change of variables).

$$\begin{aligned}\mathcal{R}_j(x) = \frac{\omega^{r5}}{\gamma^{r5}}\Big(\mathcal{R}_j^{r5}(x) - \gamma_1^{r3}\mathcal{R}_j^{(1),r3}(x) - \gamma_2^{r3}\mathcal{R}_j^{(2),r3}(x) - \gamma_3^{r3}\mathcal{R}_j^{(3),r3}(x)\Big) \\ + \omega_1^{r3}\mathcal{R}_j^{(1),r3}(x) + \omega_2^{r3}\mathcal{R}_j^{(2),r3}(x) + \omega_3^{r3}\mathcal{R}_j^{(3),r3}(x).\end{aligned} \tag{2.1.22}$$

2.1.2 Time discretizations: Runge-Kutta methods

Runge-Kutta (RK) methods are commonly used to solve initial value problems of the form

$$\begin{cases}\dfrac{du}{dt} = \mathcal{L}(u;t), \quad t > 0, \\ u(t_0) = u_0.\end{cases} \tag{2.1.23}$$

Traditionally, RK methods are placed into one of two categories: **explicit RK methods** or **implicit RK methods**. Discretizing in time, explicit methods evaluate the right-hand side of equation (2.1.23) at the current or known times, e.g., t^n. Whereas, implicit methods evaluate the right-hand side of equation (2.1.23) at the next or future times, e.g., t^{n+1}. Generally speaking, explicit RK methods are more straightforward to implement but could require very small time-stepping sizes to enforce numerical stability. On the other hand, implicit methods are generally more computationally expensive per time-step but allow larger time-stepping sizes in comparison. The method we use often boils down to the stiffness of the system and the region of stability of the RK method. For the purposes of this dissertation, we omit the finer theory and refer the reader to [145] for more details.

In practice, we use RK methods to solve partial differential equations $u_t = \mathcal{L}(u;x,t)$ after discretizing in space so that we are left with a function of u and t. Known as the **method of lines** approach, we then solve for a vector function $\mathbf{u} \in \mathbb{R}^{N_x \times 1}$ modeled by

$$\begin{cases}\dfrac{d\mathbf{u}}{dt} = \mathbf{L}(\mathbf{u};t), \quad t > 0, \\ \mathbf{u}(t_0) = \mathbf{u}_0.\end{cases} \tag{2.1.24}$$

The simplest RK methods are **forward Euler** (explicit) and **backward Euler**

(implicit), respectively given by

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{L}(\mathbf{u}^n; t^n), \tag{2.1.25a}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{L}(\mathbf{u}^{n+1}; t^{n+1}). \tag{2.1.25b}$$

Both forward Euler and backward Euler are first-order accurate in time, that is, $\mathbf{u}^n = \mathbf{u}(t^n) + \mathcal{O}(\Delta t)$. Note that equation (2.1.25b) requires solving a linear system; this is why implicit methods have a higher computational complexity per time-step. Of course, there are higher-order accurate RK methods. The rough idea to achieving high-order accuracy in time is to approximate the solution at sequential stages within the time interval $[t^n, t^{n+1}]$, starting at t^n and ending at t^{n+1}; this is why RK methods are classified as one-step, multi-stage methods. Most textbooks present a general $m-$stage RK method in the following form.

$$u^{n=0} = u_0, \tag{2.1.26}$$

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{m} b_i K_i, \qquad n \geq 0, \tag{2.1.27}$$

$$K_i = \mathcal{L}\left(u^n + \Delta t \sum_{j=1}^{m} a_{ij} K_j; t^n + c_i \Delta t \right), \qquad i = 1, 2, ..., m, \tag{2.1.28}$$

where $c_i = \sum_{j=1}^{m} a_{ij}$ for $1, 2, ..., m$, and $\sum_{i=1}^{m} b_i = 1$ for consistency. The RK method (2.1.26) is an explicit method if $a_{ij} = 0$ for all $j \geq i$, and is an implicit method otherwise. It is also common practice to organize the coefficients of a RK method in a **Butcher table**,

$\mathbf{c}$	$\mathbf{A}$
	$\mathbf{b}^t$

where $\mathbf{A} = (a_{ij})$, $\mathbf{c} = [c_1, ..., c_m]^t$ and $\mathbf{b} = [b_1, ..., b_m]^t$. For illustrative purposes, we outline the Butcher table of the very popular explicit fourth-order RK method.

Explicit fourth-order Runge-Kutta method:

$$u^{n+1} = u^n + \frac{\Delta t}{6}(K_1 + 2K_2 + 2K_3 + K_4), \tag{2.1.29}$$

where $K_1 = \mathcal{L}(u^n; t^n)$, $K_2 = \mathcal{L}(u^n + \frac{\Delta t}{2}K_1; t^{n+1/2})$, $K_3 = \mathcal{L}(u^n + \frac{\Delta t}{2}K_2; t^{n+1/2})$ and $K_4 = \mathcal{L}(u^n + \Delta t K_3; t^{n+1})$. The Butcher table for RK method (2.1.29) is given below.

	RK4			
0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/6	1/3	1/6

2.1.2.1 Strong stability-preserving Runge-Kutta methods

We have seen how RK methods can solve time-dependent systems. However, as partial differential equations and their solutions become more complicated, stability becomes a pressing concern when choosing a time-stepping method. Standard linear stability analysis [145] is sufficient for problems with smooth solutions. Whereas, a stronger measure of stability is required when solving problems with discontinuous solutions, such as those that arise in hyperbolic partial differential equations.

This stronger stability is as follows: assuming the forward Euler method is strongly stable under some (semi)norm when the time-stepping size is suitably restricted, we desire high-order time discretizations (e.g., Runge-Kutta methods) that maintain strong stability under the same norm but with a possibly different time-stepping size restriction. In other words, these so-called **strong stability-preserving (SSP)** methods aim to maintain the strong stability in the same (semi)norm as the

forward-Euler method [77, 158, 159]. Originally, the authors in [158, 159] let the relevant (semi)norm be the total variation (TV) seminorm,

$$TV(u^n) \coloneqq \sum_j |u^n_{j+1} - u^n_j|.$$

Subsection 2.1.1 discusses spatial reconstruction operators, $\mathcal{L}$, that discretize the solution in space and control spurious oscillations. We *assume* that the spatial discretization, when combined with the forward Euler method under some time-stepping restriction $\Delta t \leq \Delta t_{FE}$ (i.e., the CFL condition for forward Euler), satisfies the strong stability property

$$\left\| u^n + \Delta t \mathcal{L}(u^n) \right\| \leq \|u^n\|,$$

where $\|\cdot\|$ is the relevant (semi)norm. In the case where the TV seminorm is used, we call this the TV property.

Broadly speaking, by taking a convex combination of SSP operators under the time-stepping restriction $\Delta t \leq \Delta t_{FE}$, we attain a RK method that preserves the (assumed) strong stability of the forward Euler method combined with spatial discretization $\mathcal{L}(u)$. We refer the reader to [77] for a rigorous derivation of SSP RK methods. For the purposes of this dissertation, we provide the Butcher tables for the optimal explicit second- and third-order accurate SSP RK methods [77].

SSP RK2

0	0	0
1	1	0
	1/2	1/2

SSP RK3

0	0	0	0
1	1	0	0
1/2	1/4	1/4	0
	1/6	1/6	2/3

The class of SSP RK methods is very rich with explicit time discretizations, implicit time discretizations and variations of these methods. Additional SSP RK methods can be found in [44, 92].

2.1.2.2 Implicit-explicit Runge-Kutta methods

We can split equation (2.1.23) into its non-stiff and stiff terms,

$$\begin{cases} \dfrac{du}{dt} = \mathcal{F}(u;t) + \mathcal{G}(u;t), \quad t > 0, \\ u(t_0) = u_0, \end{cases} \tag{2.1.30}$$

where $\mathcal{F}(u;t)$ is the non-stiff term and $\mathcal{G}(u;t)$ is the stiff term. Discretization methods used for such problems are **implicit-explicit (IMEX) RK schemes** [10, 44, 92]. The intuition behind these schemes is straightforward – evolve the non-stiff term explicitly, and evolve the stiff term implicitly. As such, each stage in the RK method will involve explicitly evaluating the non-stiff term, and solving a linear system due to the stiff term. For simplicity, we only use the IMEX RK schemes outlined in [10]. As per the notation used by Ascher, et al., IMEX(s,σ,p) denotes using an $s-$stage diagonally-implicit Runge-Kutta (DIRK) scheme for $\mathcal{G}(u;t)$, using a $\sigma-$stage explicit RK scheme for $\mathcal{F}(u;t)$, and being of combined order p. DIRK methods are a special class of implicit RK methods and we leave their discussion to [4].

The IMEX(s,σ,p) RK schemes are expressed with two Butcher tables: one for the implicit RK method, and another for the explicit RK method. Note that the Butcher table for the implicit RK method is padded with zeros in the first column and row; this is to make the dimensions consistent with the Butcher table for the explicit RK method.

Implicit Scheme

0	0	0	0	$\dots$	0
c_1	0	a_{11}	0	$\dots$	0
c_2	0	a_{21}	a_{22}	$\dots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	0	a_{s1}	a_{s2}	$\dots$	a_{ss}
	0	b_1	b_2	$\dots$	b_s

Explicit Scheme

0	0	0	0	$\dots$	0
c_1	$\hat{a}_{21}$	0	0	$\dots$	0
c_2	$\hat{a}_{31}$	$\hat{a}_{32}$	0	$\dots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	$\hat{a}_{\sigma 1}$	$\hat{a}_{\sigma 2}$	$\hat{a}_{\sigma 3}$	$\dots$	0
	$\hat{b}_1$	$\hat{b}_2$	$\hat{b}_3$	$\dots$	$\hat{b}_\sigma$

More precisely, the IMEX(s,σ,p) scheme is as follows.

$$u^{n+1} = u^n + \Delta t \sum_{\mu=1}^{s} b_\mu K_\mu + \Delta t \sum_{\mu=1}^{\sigma} \hat{b}_\mu \hat{K}_\mu, \tag{2.1.31a}$$

$$K_\mu = \mathcal{G}(u^{(\mu)}; t^{(\mu)}), \qquad \mu = 1, 2, ..., s, \tag{2.1.31b}$$

$$\hat{K}_1 = \mathcal{F}(u^n; t^n), \tag{2.1.31c}$$

$$\hat{K}_{\mu+1} = \mathcal{F}(u^{(\mu)}; t^{(\mu)}), \qquad \mu = 1, 2, ..., s. \tag{2.1.31d}$$

Based on the IMEX RK method, the solution $u^{(\mu)}$ can be approximated by

$$u^{(\mu)} = u^n + \Delta t \sum_{\nu=1}^{\mu} a_{\mu,\nu} K_\nu + \Delta t \sum_{\nu=1}^{\mu} \hat{a}_{\mu+1,\nu} \hat{K}_\nu, \qquad \mu = 1, 2, ..., s. \tag{2.1.32}$$

Note that each stage of an IMEX RK method will require solving a linear system since K_μ is at time $t^{(\mu)}$. We provide the Butcher tables for several IMEX RK methods presented in [10]. By construction, each IMEX RK scheme has slightly different properties that are better suited for different problems. Some schemes might have better damping properties and stability regions, be stiffly-accurate, etc. Further details and other IMEX RK methods, including IMEX SSP RK methods, can be found in [10, 44, 92].

IMEX(1,1,1) – Implicit Table

0	0	0
1	0	1
	0	1

IMEX(1,1,1) – Explicit Table

0	0	0
1	1	0
	1	0

IMEX(1,2,2) – Implicit Table

0	0	0
1/2	0	1/2
	0	1

IMEX(1,2,2) – Explicit Table

0	0	0
1/2	1/2	0
	0	1

IMEX(2,2,2) – Implicit Table

0	0	0	0
γ	0	γ	0
1	0	$1-\gamma$	γ
	0	$1-\gamma$	γ

IMEX(2,2,2) – Explicit Table

0	0	0	0
γ	γ	0	0
1	δ	$1-\delta$	0
	δ	$1-\delta$	0

IMEX(2,2,2) Let $\gamma = 1 - \sqrt{2}/2$ and $\delta = 1 - 1/(2\gamma)$.

IMEX(2,3,3) – Implicit Table

0	0	0	0
γ	0	γ	0
$1-\gamma$	0	$1-2\gamma$	γ
	0	1/2	1/2

IMEX(2,3,3) – Explicit Table

0	0	0	0
γ	γ	0	0
$1-\gamma$	$\gamma-1$	$2(1-\gamma)$	0
	0	1/2	1/2

IMEX(2,3,3) Let $\gamma = (3 + \sqrt{3})/6$.

IMEX(2,3,2) – Implicit Table

0	0	0	0
γ	0	γ	0
1	0	$1-\gamma$	γ
	0	$1-\gamma$	γ

IMEX(2,3,2) – Explicit Table

0	0	0	0
γ	γ	0	0
1	δ	$1-\delta$	0
	0	$1-\gamma$	γ

IMEX(2,3,2) Let $\gamma = (2 - \sqrt{2})/2$ and $\delta = -2\sqrt{2}/3$.

IMEX(3,4,3) – Implicit Table

0	0	0	0	0
γ	0	γ	0	0
0.717933	0	0.282067	γ	0
1	0	1.208497	-0.644363	γ
	0	1.208497	-0.644363	γ

IMEX(3,4,3) – Explicit Table

0	0	0	0	0
γ	γ	0	0	0
0.717933	0.321279	0.396654	0	0
1	-0.105858	0.552929	0.552929	0
	0	1.208497	-0.644363	γ

IMEX(3,4,3) Let $\gamma = 0.435867$.

IMEX(4,4,3) – Implicit Table

0	0	0	0	0	0
1/2	0	1/2	0	0	0
2/3	0	1/6	1/2	0	0
1/2	0	-1/2	1/2	1/2	0
1	0	3/2	-3/2	1/2	1/2
	0	3/2	-3/2	1/2	1/2

IMEX(4,4,3) – Explicit Table

0	0	0	0	0	0
1/2	1/2	0	0	0	0
2/3	11/18	1/18	0	0	0
1/2	5/6	-5/6	1/2	0	0
1	1/4	7/4	3/4	-7/4	0
	1/4	7/4	3/4	-7/4	0

2.1.3 Operator splitting

When numerically solving equation (2.1.23) it is sometimes easier to decompose the differential operator $\mathcal{L}$ into a sum of simpler sub-operators and solve a sequence of simpler problems. Of course, this assumes that $\mathcal{L}$ can be decomposed in this manner. Consider the case of splitting $\mathcal{L}$ into the sum of two sub-operators $\mathcal{L}_1$ and $\mathcal{L}_2$. Equation (2.1.23) is rewritten as

$$\begin{cases} \dfrac{du}{dt} = \mathcal{L}_1(u;t) + \mathcal{L}_2(u;t), \quad t > 0, \\ u(t_0) = u_0. \end{cases} \tag{2.1.33}$$

Dimensional splitting methods solve equation (2.1.33), and hence equation (2.1.23), by alternating between solving the easier problems

$$\frac{du}{dt} = \mathcal{L}_1(u;t), \tag{2.1.34a}$$

$$\frac{du}{dt} = \mathcal{L}_2(u;t). \tag{2.1.34b}$$

For example, we might solve equation (2.1.34a) over an entire time-step Δt to get the intermediate solution u^*; and then use u^* as the initial condition when solving equation (2.1.34b) over an entire time-step Δt to get the updated solution u^{n+1}. Notice that we will have updated the solution with respect to each differential sub-operator for an entire time-step Δt. In this sense, we have reduced the problem to only dealing with

one sub-operator at a time. The example we just described is known as **Lie-Trotter splitting**, and this is a first-order splitting method since it introduces a temporal error $\mathcal{O}(\Delta t)$.

In this subsection, we present a few common splitting methods up to high-order accuracy. We only deal with the two sub-operator case $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$, but the extension to three or more sub-operators follows naturally. It should be noted that the more sub-operators you have, the greater the computational complexity, especially with higher-order splitting methods. For the sake of keeping this subsection focused on the implementation, we refer the reader to [119, 130] for details concerning the order of accuracy.

2.1.3.1 First-order Lie-Trotter splitting

- **Step 1 ($\mathcal{L}_1$ sub-operator).**

  Solve equation (2.1.34a) over a time-step Δt; that is, over $[t^n, t^{n+1}]$.

  $u^n \quad \longrightarrow \quad u^*$.

- **Step 2 ($\mathcal{L}_2$ sub-operator).**

  Solve equation (2.1.34b) over a time-step Δt; that is, over $[t^n, t^{n+1}]$.

  $u^* \quad \longrightarrow \quad u^{n+1}$.

2.1.3.2 Second-order Strang splitting

- **Step 1 ($\mathcal{L}_1$ sub-operator).**

  Solve equation (2.1.34a) over a time-step $\Delta t/2$; that is, over $[t^n, t^{n+1/2}]$.

  $u^n \quad \longrightarrow \quad u^*$.

- **Step 2 ($\mathcal{L}_2$ sub-operator).**

  Solve equation (2.1.34b) over a time-step Δt; that is, over $[t^n, t^{n+1}]$.

  $u^* \quad \longrightarrow \quad u^{**}$.

- **Step 3 ($\mathcal{L}_1$ sub-operator).**

Solve equation (2.1.34a) over a time-step $\Delta t/2$; that is, over $[t^{n+1/2}, t^{n+1}]$.

$u^{**} \quad \longrightarrow \quad u^{n+1}$.

2.1.3.3 Higher-order splitting

As seen with Strang splitting, higher-order splitting methods require several more steps alternating between solving equations (2.1.34a) and (2.1.34b). This is one downside to higher-order splitting methods. However, for the case of $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$, the computational cost of the splitting is usually reasonable. We only present a fourth-order splitting method [73, 182], but other high-order splitting methods can be found in [35, 73, 182]. It is important to note that splitting methods of order three or more must have at least one negative time-step [74]. When $\mathcal{L}$ is a spatial differential operator that contains a diffusive term, it is well-known that negative time integration leads to significant instabilities. In this case, we are limited to at best second-order Strang splitting.

Define two constants

$$\gamma_1 = \frac{1}{2 - 2^{1/3}} \approx 1.351207191959658 \quad \text{and} \quad \gamma_2 = \frac{-2^{1/3}}{2 - 2^{1/3}} \approx -1.702414383919315.$$

- **Step 1 ($\mathcal{L}_1$ sub-operator).**

  Solve equation (2.1.34a) over a time-step $\gamma_1 \Delta t/2$.

- **Step 2 ($\mathcal{L}_2$ sub-operator).**

  Solve equation (2.1.34b) over a time-step $\gamma_1 \Delta t$.

- **Step 3 ($\mathcal{L}_1$ sub-operator).**

  Solve equation (2.1.34a) over a time-step $(\gamma_1 + \gamma_2)\Delta t/2$.

- **Step 4 ($\mathcal{L}_2$ sub-operator).**

  Solve equation (2.1.34b) over a time-step $\gamma_2 \Delta t$.

- **Step 5 ($\mathcal{L}_1$ sub-operator).**

  Solve equation (2.1.34a) over a time-step $(\gamma_1 + \gamma_2)\Delta t/2$.

- **Step 6 ($\mathcal{L}_2$ sub-operator).**

  Solve equation (2.1.34b) over a time-step $\gamma_1 \Delta t$.

- **Step 7 ($\mathcal{L}_1$ sub-operator).**

  Solve equation (2.1.34a) over a time-step $\gamma_1 \Delta t/2$.

Note that steps 2 and 6 require steps larger than Δt, and steps 3, 4, and 5 require steps backwards in time.

2.1.4 Gauss-Legendre quadrature

Quadratures are a class of methods that explicitly approximate definite integrals. Common quadratures for functions of a single variable include Gauss-Legendre quadrature over finite intervals $[a, b]$, Gauss-Laguerre quadrature over unbounded intervals $[0, +\infty)$, and Gauss-Hermite quadrature over unbounded intervals $(-\infty, +\infty)$. There exist several different quadrature methods, each with their own advantages [60, 101, 145, 147, 171]. We only present the Gauss-Legendre quadrature formula and refer the reader to [101, 145, 147] for derivations and more details.

Considering the domain $[-1, 1]$, we desire an approximation to the definite integral $\int_{-1}^{1} f(x)dx$. Broadly speaking, the Gauss-Legendre quadrature approximates the integral $\int_{-1}^{1} f(x)dx$ with

$$\sum_{k=1}^{n} w_k f(x_k), \tag{2.1.35}$$

where the **quadrature nodes** $\{x_k : k = 1, 2, ..., n\}$ are the roots of the Legendre polynomial $L_n(x) \in \mathcal{P}_n$, and the **quadrature weights** are

$$w_k = \frac{2}{(1 - x_k)^2 \left(L_n'(x_k)\right)^2}, \qquad k = 1, 2, ..., n. \tag{2.1.36}$$

We assume that the Legendre polynomials are normalized over the interval $[-1, 1]$ so that $L_n(1) = 1$ for all $n \geq 1$. The Legendre polynomials are defined by the formula

$$L_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}(x^2 - 1)^n, \tag{2.1.37}$$

which can be easily proved by induction. We note that the Legendre polynomials are orthogonal with respect to the L^2 inner product with $\|L_n\|_2^2 = 2/(2n+1)$.

One can easily approximate an integral over a general bounded interval $\int_a^b f(x)dx$ by using quadrature formula (2.1.35) under the linear mapping $\phi : [a,b] \to [-1,1]$. Or, one could extend this idea to construct a composite quadrature. The n-point Gauss-Legendre composite quadrature (2.1.35) has $2n$-order of accuracy. In other words, if the interval $[a,b]$ is partitioned into M uniform and disjoint sub-intervals $[a,b] = \cup_{m=1}^{M} I_m$ each of width $h = (b-a)/M$, then

$$\left| \int_a^b f(x)dx - \sum_{m=1}^{M} G_m \right| = \mathcal{O}(h^{2n}),$$

where n is the number of quadrature nodes/weights and G_m is the n-point Gauss-Legendre quadrature (2.1.35) of sub-interval I_m. Table 2.1 presents the Gauss-Legendre quadrature nodes and weights over the interval $[-1,1]$.

Table 2.1: Nodes and weights for the order-n Gauss-Legendre quadrature over $[-1,1]$.

Number of nodes, n	Nodes, x_k	Weights, w_k
2	$\pm\frac{1}{\sqrt{3}}$	1
3	0	8/9
	$\pm\sqrt{\frac{3}{5}}$	5/9
4	$\pm$0.339981043584856	0.652145154862546
	$\pm$0.861136311594053	0.347854845137454
5	0	0.568888888888889
	$\pm$0.538469310105683	0.478628670499366
	$\pm$0.906179845938664	0.236926885056189
6	$\pm$0.238619186083197	0.4679139345726910
	$\pm$0.661209386466265	0.360761573048139
	$\pm$0.932469514203152	0.171324492379170

2.2 The EL-RK-FV method for pure convection problems

The spirit of the EL-RK-FV method is best demonstrated by starting with a pure convection problem in one dimension, i.e., equation (2.0.1) with $\epsilon = 0$ and $g(x) = 0$. Let the flux be denoted $f(x)$. We first discuss the formulation of the scheme in Section 2.2.1, followed by discussion of high-order spatial reconstruction in Section 2.2.2 and time discretization in Section 2.2.3.

2.2.1 Scheme formulation

We discretize the spatial domain $[a, b]$ into N_x intervals with $N_x + 1$ uniformly distributed nodes

$$a = x_{\frac{1}{2}} < x_{\frac{3}{2}} < ... < x_{N_x-\frac{1}{2}} < x_{N_x+\frac{1}{2}} = b.$$

Define the cells $I_j \coloneqq [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ with centers $x_j = (x_{j-\frac{1}{2}} + x_{j+\frac{1}{2}})/2$ and widths $\Delta x_j = \Delta x$ for $j = 1, ..., N_x$. We let

$$\Delta t = \frac{CFL\Delta x}{\max|f'(u)|}, \tag{2.2.1}$$

where CFL defines the time stepping size. In contrast to Eulerian methods, which evolve the solution on a stationary mesh, our EL algorithm proposes tracing the characteristics *backwards* in time from t^{n+1} to t^n to partition a set of space-time regions based on the computational grid. Since tracing characteristics in the nonlinear case is often nontrivial, we consider computing *approximations of characteristics* that are linear space-time curves. In particular, the *approximate characteristic speeds* at nodes $x_{j+\frac{1}{2}}$ and time t^{n+1} are defined using the Rankine-Hugoniot jump condition at time t^n,

$$\nu_{j+\frac{1}{2}} = \begin{cases} \dfrac{f(\overline{u}^n_{j+1}) - f(\overline{u}^n_j)}{\overline{u}^n_{j+1} - \overline{u}^n_j}, & \overline{u}^n_{j+1} \neq \overline{u}^n_j, \\ f'(\overline{u}^n_j), & \overline{u}^n_{j+1} = \overline{u}^n_j, \end{cases} \tag{2.2.2}$$

where $\overline{u}^n_j$ and $\overline{u}^n_{j+1}$ are the cell averages at time t^n. In practice we tested if $|\overline{u}^n_{j+1} - \overline{u}^n_j| < \epsilon$, with $\epsilon = 1.0e-8$, in which case we took $\overline{u}$ to be the average of $\overline{u}^n_j$ and $\overline{u}^n_{j+1}$. As seen

in Figure 2.2, the (upstream) traceback nodes can be defined by

$$x^*_{j+\frac{1}{2}} \coloneqq x_{j+\frac{1}{2}} - \nu_{j+\frac{1}{2}}\Delta t, \quad j = 1, ..., N. \tag{2.2.3}$$

Further define $\tilde{x}_{j+\frac{1}{2}}(t) = x^*_{j+\frac{1}{2}} + \nu_{j+\frac{1}{2}}(t - t^n)$ with $t^n \le t \le t^{n+1}$, for $j = 0, 1, 2, ..., N$. The approximate characteristics are given by the space-time tracelines

$$\mathcal{S}_{\text{left}} \coloneqq \{(\tilde{x}_{j-\frac{1}{2}}(t), t) : t^n \le t \le t^{n+1}\} \quad \text{and} \quad \mathcal{S}_{\text{right}} \coloneqq \{(\tilde{x}_{j+\frac{1}{2}}(t), t) : t^n \le t \le t^{n+1}\}.$$

Note that $x^*_{j+\frac{1}{2}} = \tilde{x}_{j+\frac{1}{2}}(t^n)$, $x_{j+\frac{1}{2}} = \tilde{x}_{j+\frac{1}{2}}(t^{n+1})$, and the (upstream) traceback cells $I^*_j = \tilde{I}_j(t^n)$ are in general nonuniform. We define the space-time domain Ω_j as the region bounded by I_j, I^*_j, $\mathcal{S}_{\text{left}}$, and $\mathcal{S}_{\text{right}}$, as seen in Figure 2.2.

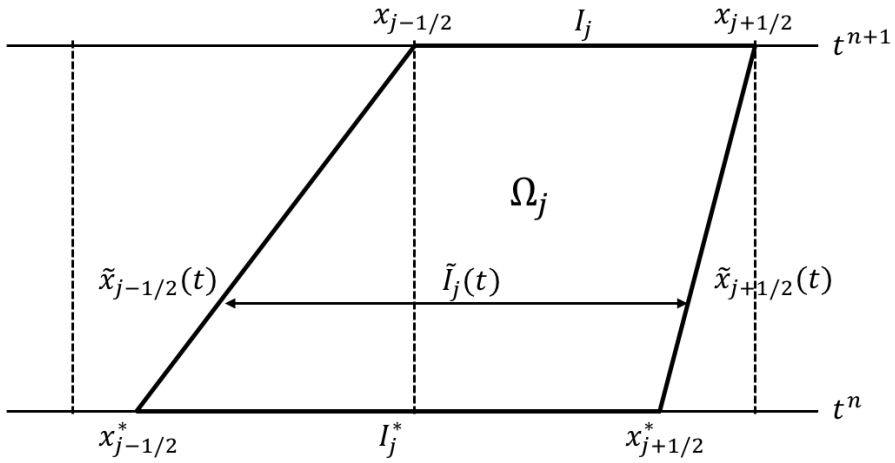


Figure 2.2: The space-time region Ω_j.

With the constructed space-time region Ω_j, we rewrite the one-dimensional pure convection problem in divergence form $\nabla_{t,x} \cdot (u, f(u))^T = 0$, integrate it over Ω_j, and

apply the divergence theorem to get

$$\int_{I_j} u(x,t^{n+1})dx - \int_{I_j^*} u(x,t^n)dx = -\left[\int_{t^n}^{t^{n+1}} \left(f(u(\tilde{x}_{j+\frac{1}{2}}(t),t)) - \nu_{j+\frac{1}{2}} u(\tilde{x}_{j+\frac{1}{2}}(t),t)\right)dt - \int_{t^n}^{t^{n+1}} \left(f(u(\tilde{x}_{j-\frac{1}{2}}(t),t)) - \nu_{j-\frac{1}{2}} u(\tilde{x}_{j-\frac{1}{2}}(t),t)\right)dt\right]. \tag{2.2.4}$$

We rewrite the time-integral form (2.2.4) into the time-differential form to get

$$\frac{d}{dt}\int_{\tilde{I}_j(t)} u(x,t)dx = -\Big[F_{j+\frac{1}{2}}(t) - F_{j-\frac{1}{2}}(t)\Big], \tag{2.2.5}$$

where $F_{j+\frac{1}{2}}(t) \coloneqq f(u(\tilde{x}_{j+\frac{1}{2}}(t),t)) - \nu_{j+\frac{1}{2}} u(\tilde{x}_{j+\frac{1}{2}}(t),t)$ is called the *modified flux function*. Choosing any appropriate monotone numerical flux function

$$\hat{F}_{j+\frac{1}{2}}(t) = \hat{F}_{j+\frac{1}{2}}(u^-_{j+\frac{1}{2}}, u^+_{j+\frac{1}{2}}; t) \tag{2.2.6}$$

(e.g., Lax-Friedrichs flux) for the modified flux function, equation (2.2.5) can be rewritten as the following semi-discrete finite volume scheme:

$$\frac{d}{dt}\int_{\tilde{I}_j(t)} u(x,t)dx = -\Big[\hat{F}_{j+\frac{1}{2}}(t) - \hat{F}_{j-\frac{1}{2}}(t)\Big]. \tag{2.2.7}$$

Here, the starting condition is obtained by a solution remapping onto a traceback grid, discussed in Section 2.2.2; $\hat{F}_{j+\frac{1}{2}}(t)$ are computed from (2.2.6) with reconstructed values from neighboring cell averages at time t, discussed in Section 2.2.3. We evolve equation (2.2.7) by the MOL approach with explicit RK schemes, discussed in Section 2.2.4.

2.2.2 Solution remapping onto a traceback grid

Referring back to Figure 2.2 and equation (2.2.4), we see that the solution will need to be projected onto the traceback grid in order to compute the *traceback cell*

averages

$$\tilde{u}_j^n \coloneqq \frac{1}{\Delta x_j^*} \int_{I_j^*} u(x, t^n) dx, \qquad j = 1, 2, ..., N, \tag{2.2.8}$$

that is, the starting condition for (2.2.7). Hence, we desire a procedure for an *integral reconstruction* that in general uses known uniform cell averages $\{\overline{u}_j(t) : j = 1, 2, ..., N\}$ to approximate the desired cell averages $\{\tilde{u}_j(t) : j = 1, 2, ..., N\}$ defined by

$$\tilde{u}_j(t) \coloneqq \frac{1}{\Delta \tilde{x}_j(t)} \int_{\tilde{I}_j(t)} u(x, t) dx, \qquad j = 1, 2, ..., N. \tag{2.2.9}$$

Unless otherwise stated, overlines (e.g., $\overline{u}_j(t)$) denote *uniform* cell averages, and tildes (e.g., $\tilde{u}_j(t)$) denote *nonuniform* cell averages.

Since discontinuities and sharp gradients can occur when solving pure convection problems, we use high-resolution schemes to control spurious oscillations, e.g., weighted essential non-oscillatory (WENO) methods [11, 43, 120, 157]. Again, we assume to only be given the uniform cell averages at some time t. One might apply the well known WENO procedure presented in [43, 157] to obtain the *reconstruction polynomials* $\mathcal{R}_j(x \in I_j)$ in terms of the neighboring uniform cell averages (at time t) $\overline{u}_i$, $i = j - p, ..., j + q$. Referring to Figure 2.2 for the sake of demonstration, we might then approximate the cell averages $\tilde{u}_j^*$ by

$$\tilde{u}_j^* \approx \frac{1}{\Delta x_j^*} \left(\int_{I_j^* \cap I_{j-1}} \mathcal{R}_{j-1}(x) dx + \int_{I_j^* \cap I_j} \mathcal{R}_j(x) dx \right). \tag{2.2.10}$$

However, the linear weights in the WENO reconstruction are not guaranteed to exist or be positive at arbitrary points. To alleviate this issue, we instead use the WENO schemes with adaptive order (WENO-AO) presented in [11] since the linear weights exist at arbitrary points.

The overarching idea of WENO-AO methods is to provide high order accuracy for smooth solutions over a large center stencil and adaptively reduce to lower order accuracy when the solution does not permit the high order accuracy. This is done

by creating a nonlinear hybridization between a large center stencil with high order accuracy, and very stable lower order WENO schemes (e.g., CWENO schemes [120]). Aside from the high order accuracy and existence of linear weights at arbitrary points, the robustness of these WENO-AO schemes is particularly attractive for our purposes. The authors in [11] write WENO-AO(p,r) to denote an adaptive order that is at best pth order (from the large center stencil) and at worst rth order (from the stable lower order stencils). For our purposes we use WENO-AO(5,3). The end product of WENO and WENO-AO methods is ultimately a reconstruction polynomial $\mathcal{R}_j(x \in I_j)$ that we shall use for reconstruction.

Equation (2.2.10) is valid when using WENO-AO reconstruction polynomials $\mathcal{R}_j(x \in I_j)$. Let I_ℓ and I_r denote the uniform cells that the *left* and *right* traceback points $\tilde{x}_{j-\frac{1}{2}}(t)$ and $\tilde{x}_{j+\frac{1}{2}}(t)$ reside in, respectively. Here, subscripts ℓ and r denote the indices of their respective uniform cells. Then

$$\int_{\tilde{I}_j(t)} u(x,t)dx \approx \int_{\tilde{I}_j(t)\cap I_\ell} \mathcal{R}_\ell(x)dx + \Delta x \sum_{i=\ell+1}^{r-1} \overline{u}_i(t) + \int_{\tilde{I}_j(t)\cap I_r} \mathcal{R}_r(x)dx. \tag{2.2.11}$$

We summarize the integral reconstruction procedure using WENO-AO below.

Algorithm 2.1. Integral reconstruction using WENO-AO

Input: uniform cell averages $\overline{u}_j(t)$ on the background grid of nodes $x_{j+\frac{1}{2}}$.

Output: possibly nonuniform cell averages $\tilde{u}_j(t)$ on the traceback grid of nodes $\tilde{x}_{j+\frac{1}{2}}(t)$.

do $j = 1, 2, ..., N$

Locate the uniform background cells that $\tilde{x}_{j-\frac{1}{2}}(t)$ and $\tilde{x}_{j+\frac{1}{2}}(t)$ reside in.

Call these cells I_ℓ and I_r, respectively.

if $\ell = r$

Compute $\tilde{u}_j(t) \approx \dfrac{1}{\Delta\tilde{x}_j(t)} \displaystyle\int_{\tilde{I}_j(t)} \mathcal{R}_\ell(x)dx.$

else

do $k = \ell, r$

Compute $\displaystyle\int_{\tilde{I}_j(t)\cap I_k} \mathcal{R}_k(x)dx.$

end do

Compute $\tilde{u}_j(t) \approx \dfrac{1}{\Delta\tilde{x}_j(t)} \left(\displaystyle\int_{\tilde{I}_j(t)\cap I_\ell} \mathcal{R}_\ell(x)dx + \Delta x \sum_{i=\ell+1}^{r-1} \overline{u}_i(t) + \int_{\tilde{I}_j(t)\cap I_r} \mathcal{R}_r(x)dx \right).$

end if

end do

2.2.3 Reconstruction of point values

Referring back to Figure 2.2 and equations (2.2.4)-(2.2.7), we also need to reconstruct the point values $u(\tilde{x}^-_{j+\frac{1}{2}}(t), t)$ and $u(\tilde{x}^+_{j+\frac{1}{2}}(t), t)$ for the modified flux function. However, the WENO-AO schemes in [11] assume a uniform grid for convenience and efficiency. We wish to avoid using nonuniform WENO methods since the linear weights need to be recomputed every step and will quickly become expensive. However, nonuniform WENO methods for two-dimensional problems [12, 184, 185, 186, 188] might become a more reasonable and realistic choice when dealing with non-splitting algorithms. Yet, we still need to reconstruct the left and right limits on the generally nonuniform traceback grid. We propose using the continuous piecewise-linear transformation from the nonuniform grid consisting of nodes $\tilde{x}_{j+\frac{1}{2}}(t)$ to the uniform background grid; performing a WENO-AO reconstruction on the uniform grid; and mapping back to the nonuniform grid to obtain the desired limits. We call this the *nonuniform-to-uniform transformation*. Given a fixed $t \in [t^n, t^{n+1}]$, consider the linear bijection $\phi_j : \tilde{I}_j(t) \to I_j$ for $j = 1, 2, ..., N$. Letting $x \in \tilde{I}_j(t)$ and $\xi \in I_j$,

$$\int_{\tilde{I}_j(t)} u(x,t)dx = |J| \int_{I_j} u(x(\xi),t)d\xi, \tag{2.2.12}$$

where $|J|(\xi) = |dx/d\xi|(\xi)$ is the Jacobian of the bijection ϕ_j. In particular, the Jacobian is constant for the linear interpolant,

$$|J| = \left| \frac{d}{d\xi} \left(\frac{\Delta\tilde{x}_j(t)}{\Delta x}\xi - \frac{x_{j+\frac{1}{2}}\tilde{x}_{j-\frac{1}{2}}(t) + x_{j-\frac{1}{2}}\tilde{x}_{j+\frac{1}{2}}(t)}{\Delta x} \right) \right| = \frac{\Delta\tilde{x}_j(t)}{\Delta x}. \tag{2.2.13}$$

Since we want to apply WENO-AO over the uniform grid, we define the *auxiliary uniform cell averages* as

$$\breve{u}_j(t) \coloneqq \frac{1}{\Delta x}\int_{I_j} u(x(\xi),t)d\xi = \frac{1}{\Delta x|J|}\int_{\tilde{I}_j(t)} u(x,t)dx = \frac{1}{\Delta\tilde{x}_j(t)}\int_{\tilde{I}_j(t)} u(x,t)dx = \tilde{u}_j(t). \tag{2.2.14}$$

Note that we have shown the auxiliary uniform cell averages (on the uniform background grid) are identical to the nonuniform cell averages. Hence, under this continuous piecewise-linear mapping, we can directly use the nonuniform cell averages at time t in the (uniform) WENO-AO procedure to obtain the left and right limits $u(\tilde{x}^-_{j+\frac{1}{2}}(t),t)$ and $u(\tilde{x}^+_{j+\frac{1}{2}}(t),t)$. We further note that the high-order accuracy is preserved with such a strategy only when there is a smooth mapping with equation (2.2.12). Theoretical justification for the existence of such a mapping is highly nontrivial. Yet, our numerical tests verify that high-order spatial accuracy is still achieved under this continuous, piecewise linear, C^0 mapping so long as the (approximate) characteristic field that controls the traceback grid is smooth. We found that the accuracy drops to first or second order when the (approximate) characteristic field that controls the traceback grid is not smooth (e.g., a fixed traceback grid with alternating cell lengths $\frac{4}{3}\Delta x : \frac{2}{3}\Delta x$ or $\frac{9}{5}\Delta x : \frac{1}{5}\Delta x$). However, since the traceback grid is determined by the smooth velocity field via the Rankine-Hugoniot jump condition, the high-order accuracy is still observed.

Algorithm 2.2. Reconstruction using WENO-AO with the nonuniform-to-uniform transformation

---

Input: nonuniform cell averages $\tilde{u}_j(t)$ on the traceback grid.

Output: left and right limits $u(\tilde{x}^-_{j+\frac{1}{2}}(t), t)$ and $u(\tilde{x}^+_{j+\frac{1}{2}}(t), t)$.

do $j = 1, 2, ..., N$

Calculate the (uniform) WENO-AO reconstruction polynomial $\mathcal{R}_j(x \in I_j)$ in terms of the neighboring auxiliary uniform cell averages $\breve{u}_i(t) = \tilde{u}_i(t)$, $i = j - p, ..., j + q$.

Compute the left limit $u(\tilde{x}^-_{j+\frac{1}{2}}(t), t) \approx \mathcal{R}_j(x_{j+\frac{1}{2}})$.

Compute the right limit $u(\tilde{x}^+_{j-\frac{1}{2}}(t), t) \approx \mathcal{R}_j(x_{j-\frac{1}{2}})$.

end do

2.2.4 Time evolution with explicit Runge-Kutta methods

Algorithms 2.1 and 2.2 now allow us to perform (integral) reconstruction on a traceback grid consisting of nodes $\tilde{x}_{j+\frac{1}{2}}(t)$. With these two tools we can evolve equation (2.2.7) using any explicit RK method. Recall that our primary goal is to achieve high order accuracy while also taking large time steps. In our numerical tests we use WENO-AO(5,3) for the spatial approximation, although higher order WENO-AO methods can certainly be used. As such, we would like to use high-order time stepping methods. In the cases where the solution is smooth, the standard fourth-order RK method suffices. However, if the solution is discontinuous (e.g., a travelling Heaviside step function), then we require an explicit SSP RK method [77]. As mentioned in Section 2.1.2.1, explicit SSP RK methods are especially attractive when numerically solving hyperbolic conservation laws because they maintain strong stability while also achieving high order accuracy in time. When applicable, we use the optimal three-stage, third-order explicit

SSP RK method outlined below for demonstration purposes:

$$
\begin{aligned}
\Delta\tilde{x}_j^{(0)}\tilde{u}_j^{(0)} &= \Delta x_j^*\overline{u}_j^*,\\
\Delta\tilde{x}_j^{(1)}\tilde{u}_j^{(1)} &= \Delta\tilde{x}_j^{(0)}\tilde{u}_j^{(0)} - \Delta t\big(\hat{F}^{(0)}_{j+\frac{1}{2}}(u^-,u^+;t^n) - \hat{F}^{(0)}_{j-\frac{1}{2}}(u^-,u^+;t^n)\big),\\
\Delta\tilde{x}_j^{(2)}\tilde{u}_j^{(2)} &= \Delta\tilde{x}_j^{(0)}\tilde{u}_j^{(0)} - \frac{\Delta t}{4}\Big[\big(\hat{F}^{(0)}_{j+\frac{1}{2}}(u^-,u^+;t^n) - \hat{F}^{(0)}_{j-\frac{1}{2}}(u^-,u^+;t^n)\big)\\
&\qquad + \big(\hat{F}^{(1)}_{j+\frac{1}{2}}(u^-,u^+;t^{n+1}) - \hat{F}^{(1)}_{j-\frac{1}{2}}(u^-,u^+;t^{n+1})\big)\Big], \qquad (2.2.15)\\
\Delta x_j\overline{u}_j^{n+1} &= \Delta\tilde{x}_j^{(0)}\tilde{u}_j^{(0)} - \frac{2\Delta t}{3}\big(\hat{F}^{(2)}_{j+\frac{1}{2}}(u^-,u^+;t^{n+\frac{1}{2}}) - \hat{F}^{(2)}_{j-\frac{1}{2}}(u^-,u^+;t^{n+\frac{1}{2}})\big)\\
&\qquad - \frac{\Delta t}{6}\Big[\big(\hat{F}^{(0)}_{j+\frac{1}{2}}(u^-,u^+;t^n) - \hat{F}^{(0)}_{j-\frac{1}{2}}(u^-,u^+;t^n)\big)\\
&\qquad + \big(\hat{F}^{(1)}_{j+\frac{1}{2}}(u^-,u^+;t^{n+1}) - \hat{F}^{(1)}_{j-\frac{1}{2}}(u^-,u^+;t^{n+1})\big)\Big],
\end{aligned}
$$

where $\Delta\tilde{x}_j^{(k)} = \tilde{x}_{j+\frac{1}{2}}(t^{(k)}) - \tilde{x}_{j-\frac{1}{2}}(t^{(k)})$ denotes the cell lengths at stage k, $\hat{F}^{(k)}_{j\pm\frac{1}{2}}(u^-,u^+;t)$ denotes using the cell averages $\tilde{u}_j^{(k)}$ from stage k to approximate the limits $u^-_{j\pm\frac{1}{2}}$ and $u^+_{j\pm\frac{1}{2}}$ in the numerical flux function at time t using Algorithms 2.1 and 2.2.

Remark 2.1. If the approximate characteristics are defined such that $\nu_{j+\frac{1}{2}} = 0$ for all j, then the EL-RK-FV scheme reduces to the standard RK-FV scheme [157]. Referring to Figure 2.2, the approximate characteristics would be vertical lines.

Remark 2.2. In the presence of shocks, the approximate characteristics will intersect even under reasonable time-stepping sizes. The proposed EL-RK-FV scheme can handle shocks only under very small time-stepping sizes, making the scheme not ideal when going to large times after shock formation. The EL-RK-FV scheme is modified and extended to allow large time-stepping sizes in the presence of shocks in [40, 180]. In particular, it is proved that the first-order version of the modified scheme with forward Euler time discretization is total-variation-diminishing and maximum-principle-preserving under a time-stepping size at least twice as large as the CFL constraint for an Eulerian first-order finite volume method [180]. The method is extended to high-order accuracy with WENO type spatial reconstruction and RK time discretization in [40].

Remark 2.3. Once the space-time partition is defined, the EL-RK-FV scheme evolves from time t^n to time t^{n+1} in a similar fashion to ALE methods. In the ALE mindset, one can imagine that $\nu_{j+\frac{1}{2}} = 0$ fixes the grid point over this time interval (Eulerian), and $\nu_{j+\frac{1}{2}} \neq 0$ moves the grid point according to the approximate characteristic speed (Lagrangian). In this sense, the EL-RK-FV and ALE frameworks achieve the same goal, that is, if the grid speed is chosen to be the characteristic speed. However, the goal of ALE methods is to move the grid to better resolve sharp gradients. Whereas, the EL-RK-FV method is designed to approximate the characteristics with the goal of large time-stepping sizes. By tracing backwards in time and working on a fixed background mesh, the EL-RK-FV method allows very large time-stepping sizes *without* the need to evolve the mesh.

2.2.5 Two-dimensional problems

We now consider the two-dimensional equation

$$u_t + f(u)_x + g(u)_y = 0. \tag{2.2.16}$$

The spatial domain is discretized into $N_x N_y$ cells, $I_{i,j} \coloneqq I_i \times I_j$ with centers $x_{i,j} = ((x_{i-\frac{1}{2}} + x_{i+\frac{1}{2}})/2, (y_{j-\frac{1}{2}} + y_{j+\frac{1}{2}})/2)$, where the spatial discretizations in x and y are respectively given by

$$0 = x_{\frac{1}{2}} < x_{\frac{3}{2}} < ... < x_{N_x-\frac{1}{2}} < x_{N_x+\frac{1}{2}} = 2\pi$$

and

$$0 = y_{\frac{1}{2}} < y_{\frac{3}{2}} < ... < y_{N_y-\frac{1}{2}} < y_{N_y+\frac{1}{2}} = 2\pi.$$

The CFL number is defined as

$$\Delta t = \frac{CFL}{\dfrac{\max|f'(u)|}{\Delta x} + \dfrac{\max|g'(u)|}{\Delta y}}, \tag{2.2.17}$$

and the uniform cell averages at time t^n are defined by

$$\tilde{\bar{u}}_{i,j}^n := \frac{1}{\Delta x \Delta y} \int_{I_{i,j}} u(x, y, t^n) dx dy. \tag{2.2.18}$$

In the two-dimensional case, we also want to compute the *interval averages* over the interval I_i at a fixed y, or over the interval I_j at a fixed x. Define the uniform *interval averages* at time t^n with one variable fixed by

$$\bar{u}_{i|y}^n := \frac{1}{\Delta x} \int_{I_i} u(x, y, t^n) dx, \tag{2.2.19a}$$

$$\tilde{u}_{j|x}^n := \frac{1}{\Delta y} \int_{I_j} u(x, y, t^n) dy. \tag{2.2.19b}$$

Note that *only in this subsection* will the superscript tilde denote uniform interval averages in y, not nonuniform interval averages.

As discussed in Section 2.1.3, dimensional splitting methods are commonly used to solve two (or higher) dimensional problems. In the current setting, splitting methods solve equation (2.2.16) by alternating between solving the easier problems

$$u_t + f(u)_x = 0, \tag{2.2.20a}$$

$$u_t + g(u)_y = 0. \tag{2.2.20b}$$

Strang splitting uses $\tilde{\bar{u}}_{i,j}^n$ to solve (2.2.20a) over a half time step $\Delta t/2$ for intermediate solution $\tilde{\bar{u}}_{i,j}^*$; uses $\tilde{\bar{u}}_{i,j}^*$ to solve (2.2.20b) over a full time step Δt for intermediate solution $\tilde{\bar{u}}_{i,j}^{**}$; and uses $\tilde{\bar{u}}_{i,j}^{**}$ to solve (2.2.20a) over another half time step $\Delta t/2$ for solution $\tilde{\bar{u}}_{i,j}^{n+1}$. As such, we can reduce the two-dimensional problem to solving several (easier) one-dimensional problems. Since the one-dimensional EL-RK-FV algorithm evolves interval averages, we need a way to go between two-dimensional cell averages and one-dimensional interval averages. We note that the nonlinear weights in the WENO-AO method [11] are the same for all nodes within the same cell.

2.2.5.1 Going from/to cell averages to/from interval averages

Consider the interval averages as functions of y and x, respectively defined by

$$\psi_i(y) \coloneqq \bar{u}^n_{i|y} = \frac{1}{\Delta x}\int_{I_i} u(x,y,t^n)dx, \tag{2.2.21a}$$

$$\phi_j(x) \coloneqq \tilde{u}^n_{j|x} = \frac{1}{\Delta y}\int_{I_j} u(x,y,t^n)dy. \tag{2.2.21b}$$

For any given $i = 1, 2, ..., N_x$ or $j = 1, 2, ..., N_y$, consider the respective Gauss-Legendre quadrature nodes $\{x^{(i)}_k \in I_i : k = 1, ..., K\}$ and $\{y^{(j)}_l \in I_j : l = 1, ..., L\}$. Observing that the cell averages at time t^n can be expressed as the interval averages of $\psi(y)$ and $\phi(x)$,

$$\tilde{\bar{u}}^n_{i,j} = \frac{1}{\Delta y}\int_{I_j} \psi_i(y)dy = \frac{1}{\Delta x}\int_{I_i} \phi_j(x)dx, \tag{2.2.22}$$

we can use WENO-AO to go from cell averages to interval averages evaluated at the Gauss-Legendre nodes,

$$\tilde{\bar{u}}^n_{i,j} \longrightarrow \psi_i(y^{(j)}_l) = \bar{u}^n_{i|l}, \tag{2.2.23a}$$

$$\tilde{\bar{u}}^n_{i,j} \longrightarrow \phi_j(x^{(i)}_k) = \tilde{u}^n_{j|k}. \tag{2.2.23b}$$

Algorithm 2.3 presents how to implement WENO-AO to go from cell averages to $x-$interval averages at the fixed $y-$Gauss-Legendre nodes. A similar algorithm can be done to go from cell averages to $y-$interval averages at the fixed $x-$Gauss-Legendre nodes.

Algorithm 2.3. Going from cell averages to $x-$interval averages

---

Input: uniform cell averages $\tilde{\bar{u}}_{i,j}$.

Output: uniform $x-$interval averages $\bar{u}_{i|l}$ at fixed Gauss-Legendre nodes $\{y^{(j)}_l \in I_j : l = 1, ..., L\}$.

do $i = 1, 2, ..., N_x$

do $j = 1, 2, ..., N_y$

Calculate the WENO-AO reconstruction polynomial $\mathcal{R}_j^{(i)}(y \in I_j)$ in terms of the neighboring averages $\tilde{\bar{u}}_{i,k}$, $k = j-p, ..., j+q$.

do $l = 1, ..., L$

Store $\bar{u}_{i|l} = \psi_i(y_l^{(j)}) \approx \mathcal{R}_j^{(i)}(y_l^{(j)})$.

end do

end do

end do

The uniform $x-$interval averages at a fixed Gauss-Legendre node $y_l^{(j)}$ are $\{\bar{u}_{i|l} = \psi_i(y_l^{(j)}) : l = 1, 2, ..., N_x\}$.

Computing the cell averages from the interval averages at the Gauss-Legendre nodes is straightforward using a Gaussian quadrature. Let ξ and w denote the standard Gauss-Legendre nodes and weights over the interval $[-1, 1]$, respectively. Without loss of generality, we can go from $x-$interval averages to cell averages via

$$\begin{aligned}
\tilde{\bar{u}}_{i,j} &= \frac{1}{\Delta y}\int_{I_j} \psi(y)dy \\
&= \frac{1}{2}\int_{-1}^{1} \psi\Big(y_{j-\frac{1}{2}} + \frac{\Delta y}{2}(y'+1)\Big)dy' \\
&\approx \frac{1}{2}\sum_{l=1}^{L} w_l \psi\Big(y_{j-\frac{1}{2}} + \frac{\Delta y}{2}(\xi_l+1)\Big) \\
&= \frac{1}{2}\sum_{l=1}^{L} w_l \psi(y_l^{(j)}), \qquad \text{where } y_l^{(j)} = y_{j-\frac{1}{2}} + \frac{\Delta y}{2}(\xi_l+1).
\end{aligned} \tag{2.2.24}$$

2.2.5.2 A demonstration with Strang splitting

For demonstrative purposes, we outline the EL-RK-FV algorithm for two-dimensional problems via Strang splitting. Since Strang splitting is only second-order in time, higher-order splitting methods might be preferred. The fourth-order splitting method in Section 2.1.3 developed by Forest and Ruth [73] and Yoshida [182] follows similarly. Rossmanith and Seal used this fourth-order splitting method in the semi-Lagrangian

framework in [151].

Algorithm 2.4. Strang splitting for the EL-RK-FV method

Input: uniform cell averages $\tilde{\bar{u}}^n_{i,j}$.

Output: uniform cell averages $\tilde{\bar{u}}^{n+1}_{i,j}$.

Step 1 (x-direction).

Solve equation (2.2.20a) for a half time step $\Delta t/2$; that is, over $[t^n, t^{n+1/2}]$.

Use Algorithm 2.3 to get $x-$interval averages; that is, $\tilde{\bar{u}}^n_{i,j} \longrightarrow \bar{u}^n_{i|l}$.

do $l = 1, 2, ..., N_y \cdot L$

For each Gauss-Legendre node, update the $x-$interval averages by applying the 1D algorithm, that is, $\bar{u}^n_{i|l} \longrightarrow \bar{u}^*_{i|l}$.

end do

Use equation (2.2.24) to get updated cell averages; that is, $\bar{u}^*_{i|l} \longrightarrow \tilde{\bar{u}}^*_{i,j}$.

Step 2 (y-direction).

Solve equation (2.2.20b) for a full time step Δt, that is, over $[t^n, t^{n+1}]$.

Use Algorithm 2.3 analogue to get $y-$interval averages, that is, $\tilde{\bar{u}}^*_{i,j} \longrightarrow \tilde{u}^*_{j|k}$.

do $k = 1, 2, ..., N_x \cdot K$

For each Gauss-Legendre node, update the $y-$interval averages by applying the 1D algorithm, that is, $\tilde{u}^*_{j|k} \longrightarrow \tilde{u}^{**}_{j|k}$.

end do

Use equation (2.2.24) analogue to get updated cell averages, that is, $\tilde{u}^{**}_{j|k} \longrightarrow \tilde{\bar{u}}^{**}_{i,j}$.

Step 3 (x-direction).

Solve equation (2.2.20a) for a half time step $\Delta t/2$, that is, over $[t^{n+1/2}, t^{n+1}]$.

Use Algorithm 2.3 to get $x-$interval averages, that is, $\tilde{\bar{u}}^{**}_{i,j} \longrightarrow \bar{u}^{**}_{i|l}$.

do $l = 1, 2, ..., N_y \cdot L$

For each Gauss-Legendre node, update the $x-$interval averages by applying the 1D algorithm, that is, $\bar{u}^{**}_{i|l} \longrightarrow \bar{u}^{n+1}_{i|l}$.

end do

Use equation (2.2.24) to get updated cell averages; that is, $\bar{u}^{n+1}_{i|l} \longrightarrow \tilde{\bar{u}}^{n+1}_{i,j}$.

2.3 The EL-RK-FV method for convection-diffusion equations

Throughout this section, overlines (e.g., $\overline{u}_j(t)$) denote *uniform* cell averages, and tildes (e.g., $\tilde{u}_j(t)$) denote *nonuniform* cell averages. We discuss the EL-RK-FV algorithm for the convection-diffusion equation (2.0.1). Since we use dimensional splitting methods for two-dimensional problems, it suffices to discuss the 1D EL-RK-FV algorithm for problems of the form

$$u_t + f(u)_x = \epsilon u_{xx} + g(x,t), \tag{2.3.1}$$

where we impose periodic boundary conditions. Rewriting equation (2.3.1) in divergence form, integrating over the same space-time region Ω_j as in the $\epsilon = 0$ case, applying the divergence theorem and fundamental theorem of calculus, integrating over $[t^n, t^{n+1}]$, and converting to the time-differential form, we get the semi-discrete formulation

$$\frac{d}{dt}\int_{\tilde{I}_j(t)} u(x,t)dx = -\left[\hat{F}_{j+\frac{1}{2}}(t) - \hat{F}_{j-\frac{1}{2}}(t)\right] + \epsilon\int_{\tilde{I}_j(t)} u_{xx}(x,t)dx + \int_{\tilde{I}_j(t)} g(x,t)dx, \tag{2.3.2}$$

where $\hat{F}_{j+\frac{1}{2}}(t)$ is any monotone numerical flux (e.g., Lax-Friedrichs flux) and $F_{j+\frac{1}{2}}(t)$ is the modified flux function defined in (2.2.5).

2.3.1 Computing the uniform cell averages of u_{xx}

When evolving equation (2.3.2) we will need to compute the uniform cell averages of $u_{xx}(x,t)$, defined by

$$\overline{u}_{xx,j}(t) \coloneqq \frac{1}{\Delta x}\int_{I_j} u_{xx}(x,t)dx. \tag{2.3.3}$$

We use the centered five-point stencil $\{j-2, j-1, j, j+1, j+2\}$ for linear reconstruction. Let $P(x \in I_j)$ be the linear reconstruction polynomial over the centered five-point stencil. After some algebra, we find that the uniform cell averages $\overline{u}_{xx,j}(t)$ can be expressed in terms of the uniform cell averages $\overline{u}_j(t)$ with fourth-order accuracy using the equation

$$\overline{u}_{xx,j}(t) = \frac{1}{\Delta x^2}\begin{bmatrix} -\frac{1}{12} & \frac{4}{3} & -\frac{5}{2} & \frac{4}{3} & -\frac{1}{12} \end{bmatrix}\begin{bmatrix} \overline{u}_{j-2}(t) \\ \overline{u}_{j-1}(t) \\ \overline{u}_j(t) \\ \overline{u}_{j+1}(t) \\ \overline{u}_{j+2}(t) \end{bmatrix}. \tag{2.3.4}$$

For implementation it is easier to express equation (2.3.4) in matrix form (assuming periodic or zero boundary conditions). We denote this matrix $\mathbf{D}_4$ in Algorithm 2.5 stated later on. Computing the nonuniform cell averages $\tilde{u}_{xx,j}(t)$ can now be done using Algorithm 2.1.

2.3.2 Time evolution with implicit-explicit Runge-Kutta methods

We can split equation (2.3.1) into its non-stiff and stiff terms,

$$u_t = \mathcal{F}(u;x,t) + \mathcal{G}(u;x,t), \tag{2.3.5}$$

where $\mathcal{F}(u;x,t) = -f(u)_x$ is the non-stiff convective term, and $\mathcal{G}(u;x,t) = \epsilon u_{xx} + g(x,t)$ is the stiff diffusive (and source) term. Discretization methods used for such problems

are implicit-explicit (IMEX) RK schemes [10, 44, 92]. The intuition behind these schemes is straightforward – evolve the non-stiff term explicitly, and evolve the stiff term implicitly. As such, each stage in the RK method will involve explicitly evaluating the non-stiff term, and solving a linear system due to the stiff term.

All we need are the possibly nonuniform cell averages at each RK stage $\mu = 1, ..., s$ over the space-time regions Ω_j. There are two approaches one can take to approximate the possibly nonuniform cell averages at each RK stage: (1) have a single space-time partition for multiple RK stages and directly update the solution to the possibly nonuniform traceback grid at each intermediate RK stage, or (2) have multiple space-time partitions for intermediate RK stages, and compute the uniform cell averages at time $t^{(\mu)}$ and project them onto the possibly nonuniform traceback grid via Algorithm 2.1. Although the first approach is more intuitive, it is computationally expensive since the system to be solved (from the implicit part) will depend on the measure of each cell. We choose the second approach, as computing the uniform cell averages at time $t^{(\mu)}$ requires solving a system dependent only on the background uniform mesh size Δx. Since we choose the second approach, we will need to: (1) solve for the *uniform* cell averages at each consecutive stage, and (2) project the uniform cell averages onto the possibly nonuniform traceback grid via Algorithm 2.1. Recall the uniform cell averages $\overline{u}^{(\mu)}_{xx,j}$ can be computed from the uniform cell averages $\overline{u}^{(\mu)}_{j}$ using equation (2.3.4).

For simplicity, we only use the IMEX RK schemes outlined in [10]. As per the notation used by Ascher, et al., IMEX(s,σ,p) denotes using an $s-$stage diagonally-implicit Runge-Kutta (DIRK) scheme for $\mathcal{G}(u;x,t)$, using a $\sigma-$stage explicit RK scheme for $\mathcal{F}(u;x,t)$, and being of combined order p. Consider the semi-discrete formulation (2.3.2) rewritten as

$$\frac{d}{dt}\int_{\tilde{I}_j(t)} u(x,t)dx = \mathcal{F}(u;t) + \mathcal{G}(u;t), \tag{2.3.6}$$

where we redefine $\mathcal{F}(u;t) = -\left[\hat{F}_{j+\frac{1}{2}}(t) - \hat{F}_{j-\frac{1}{2}}(t)\right]$ and $\mathcal{G}(u;t) = \epsilon\int_{\tilde{I}_j(t)} u_{xx}(x,t)dx + \int_{\tilde{I}_j(t)} g(x,t)dx$. Defining $U^{(\mu)} \coloneqq \int_{\tilde{I}_j(t^{(\mu)})} u(x,t^{(\mu)})dx$, the IMEX($s$,$\sigma$,$p$) scheme over the

space-time region Ω_j is as follows:

$$U^{n+1} = U^n + \Delta t \sum_{\mu=1}^{s} b_\mu K_\mu + \Delta t \sum_{\mu=1}^{\sigma} \hat{b}_\mu \hat{K}_\mu, \tag{2.3.7a}$$

$$K_\mu = \mathcal{G}(U^{(\mu)}; t^{(\mu)}), \qquad \mu = 1, 2, ..., s, \tag{2.3.7b}$$

$$\hat{K}_1 = \mathcal{F}(U^n; t^n), \tag{2.3.7c}$$

$$\hat{K}_{\mu+1} = \mathcal{F}(U^{(\mu)}; t^{(\mu)}), \qquad \mu = 1, 2, ..., s. \tag{2.3.7d}$$

More precisely,

$$K_\mu = \epsilon \int_{\tilde{I}_j(t^{(\mu)})} u_{xx}(x, t^{(\mu)}) dx + \int_{\tilde{I}_j(t^{(\mu)})} g(x, t^{(\mu)}) dx, \tag{2.3.8a}$$

$$\hat{K}_{\mu+1} = -\left[\hat{F}_{j+\frac{1}{2}}(t^{(\mu)}) - \hat{F}_{j-\frac{1}{2}}(t^{(\mu)})\right]. \tag{2.3.8b}$$

Based on the IMEX RK method, the solution $U^{(\mu)}$ over the traceback grid can be approximated by

$$U^{(\mu)} = U^n + \Delta t \sum_{\nu=1}^{\mu} a_{\mu,\nu} K_\nu + \Delta t \sum_{\nu=1}^{\mu} \hat{a}_{\mu+1,\nu} \hat{K}_\nu, \qquad \mu = 1, 2, ..., s. \tag{2.3.9}$$

Recall that we choose *not* to directly update the solution over Ω_j. Instead, we opt to solve for the uniform cell averages at each RK stage and project them onto the corresponding possibly nonuniform traceback grid. We define sub-space-time regions ${}_\mu\Omega_j$ for the μ-th stage of the IMEX RK scheme, as shown in Figures 2.3 and 2.4. The space-time region ${}_\mu\Omega_j$ traces back to time t^n (using the same approximate characteristic speeds as in Ω_j) from time $t^{(\mu)}$. Hence, at time $t^{(\mu)}$ on the sub-space-time region ${}_\mu\Omega_j$ the grid is uniform. Lower left subscript μ denotes values confined to the sub-space-time region ${}_\mu\Omega_j$. For example, ${}_2U^n$ are the possibly nonuniform definite integrals (and hence the possibly nonuniform cell averages) at time t^n over the traceback cell in ${}_2\Omega_j$, as seen in Figure 2.4.

The desired uniform cell averages $\overline{u}_j^{(\mu)} = {}_\mu U^{(\mu)}/\Delta x$ at stage μ can be obtained by

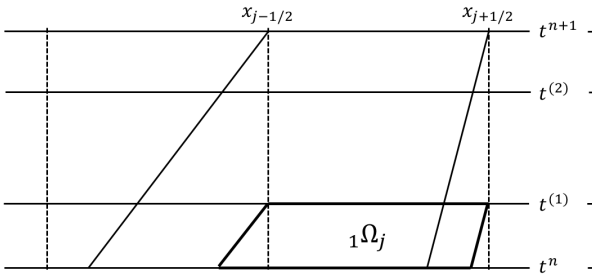


Figure 2.3: The space-time region ${}_1\Omega_j$.

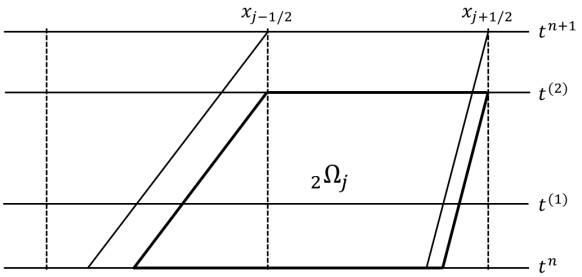


Figure 2.4: The space-time region ${}_2\Omega_j$.

computing the first $\mu-$stages of the IMEX RK scheme over the sub-space-time region ${}_\mu\Omega_j$, given below.

$$ {}_\mu U^{(\mu)} = {}_\mu U^n + \Delta t \sum_{\nu=1}^{\mu} (a_{\mu,\nu})({}_\mu K_\nu) + \Delta t \sum_{\nu=1}^{\mu} (\hat{a}_{\mu+1,\nu})({}_\mu \hat{K}_\nu) \tag{2.3.10a} $$

$$ {}_\mu K_\nu = \mathcal{G}({}_\mu U^{(\nu)}; t^{(\nu)}), \qquad \nu = 1, 2, ..., \mu, \tag{2.3.10b} $$

$$ {}_\mu \hat{K}_1 = \mathcal{F}({}_\mu U^n; t^n), \tag{2.3.10c} $$

$$ {}_\mu \hat{K}_{\nu+1} = \mathcal{F}({}_\mu U^{(\nu)}; t^{(\nu)}), \qquad \nu = 1, 2, ..., \mu. \tag{2.3.10d} $$

More precisely,

$$ {}_\mu K_1 = \epsilon \int_{{}_\mu \tilde{I}_j(t^{(\nu)})} u_{xx}(x, t^{(\nu)}) dx + \int_{{}_\mu \tilde{I}_j(t^{(\nu)})} g(x, t^{(\nu)}) dx, \tag{2.3.11a} $$

$$ {}_\mu \hat{K}_{\nu+1} = -\left[\hat{F}_{j+\frac{1}{2}}(t^{(\nu)}) - \hat{F}_{j-\frac{1}{2}}(t^{(\nu)}) \right]. \tag{2.3.11b} $$

Note that ${}_\mu \hat{K}_{\nu+1}$ uses the cell averages restricted to the sub-space-time regions ${}_\mu\Omega_j$. Further note that equation (2.3.10) can recycle and reuse the uniform cell averages already computed during stages $1, 2, ..., \mu - 1$. Simply project the uniform cell averages ($\overline{u}_j^{(\nu)}$ or $\overline{u}_{xx,j}^{(\nu)}$, $\nu = 1, 2, ..., \mu - 1$) onto the traceback grids formed by the space-time regions ${}_\mu\Omega_j$. Although IMEX RK schemes are straightforward, it is easy to get lost in the notation. To help demonstrate the EL-RK-FV algorithm coupled with an IMEX RK scheme, we present the IMEX(2,2,2) case in Appendix A.

Algorithm 2.5. EL-RK-FV algorithm coupled with an IMEX RK scheme [10]

Input: uniform cell averages $\overline{u}_j^n$.

Output: uniform cell averages $\overline{u}_j^{n+1}$.

Compute the possibly nonuniform traceback cell averages $\tilde{u}_j(t^n)$ using Algorithm 2.1.

Store $\hat{K}_1 = \mathcal{F}(U^n; t^n)$. This is the traceback grid in Ω_j.

do $\mu = 1, 2, ..., s$

1. Compute the uniform cell averages $\overline{u}_j^{(\mu)}$ at time $t^{(\mu)}$ with equation (2.3.10).

   i. Restrict yourself to the space-time region ${}_\mu\Omega_j$.

   ii. Compute the values ${}_\mu K_\nu$ and ${}_\mu\hat{K}_{\nu+1}$ at each stage $\nu = 1, 2, ..., \mu - 1$ using Algorithm 2.1 to compute the necessary nonuniform cell averages, and Algorithm 2.2 for the modified flux function. Definite integrals of $g(x,t)$ can be computed using a Gaussian quadrature.

   iii. Recalling equation (2.3.4), solve equation (2.3.10) by setting it up as the linear system

$$\left(\mathbf{I} - \frac{a_{\mu,\mu}\epsilon\Delta t}{\Delta x^2}\mathbf{D}_4\right) {}_\mu\vec{U}^{(\mu)} = {}_\mu\vec{U}^n + \Delta t \sum_{\nu=1}^{\mu-1} (a_{\mu,\nu})({}_\mu\vec{K}_\nu) + \Delta t \sum_{\nu=1}^{\mu} (\hat{a}_{\mu+1,\nu})({}_\mu\vec{\hat{K}}_\nu) + a_{\mu,\mu}\Delta t\,\vec{g}(x, t^{(\mu)}),$$

   where $\vec{g}_j(x, t^{(\mu)}) = \int_{I_j} g(x, t^{(\mu)})dx$.

   iv. Store the uniform cell averages $\overline{u}_j^{(\mu)} = {}_\mu U_j^{(\mu)}/\Delta x$.

2. Compute and store the uniform cell averages $\overline{u}_{xx,j}^{(\mu)}$ using equation (2.3.4).

3. Compute the possibly nonuniform traceback cell averages $\tilde{u}_j^{(\mu)}$ using Algorithm 2.1.

4. Compute the possibly nonuniform traceback cell averages $\tilde{u}_{xx,j}^{(\mu)}$ using Algorithm 2.1.

5. Compute and store K_μ and $\hat{K}_{\mu+1}$. Note that we are back on the possibly nonuniform traceback grid consisting of cells $\tilde{I}_j(t^{(\mu)})$.

end do

Compute $U^{n+1} = U^n + \Delta t \sum_{\mu=1}^{s} b_\mu K_\mu + \Delta t \sum_{\mu=1}^{\sigma} \hat{b}_\mu \hat{K}_\mu$.

Compute the uniform cell averages $\overline{u}_j^{n+1} = U_j^{n+1}/\Delta x$.

2.3.3 Mass conservation

We now show that the 1D EL-RK-FV algorithm is mass conservative when coupled with any IMEX RK scheme in [10]. Since we extend the EL-RK-FV algorithm to higher dimensions via dimensional splitting in this paper, showing mass conservation for the 1D problem suffices. Note that mass conservation of the $\epsilon = 0$ case can be proved just as easily.

Proposition 2.1. (Mass conservative [133]). The 1D EL-RK-FV algorithm coupled with any IMEX RK scheme in [10] for (non)linear convection-diffusion equations is mass conservative, assuming the source term $g(x,t) = 0$ and periodic boundary conditions.

Proof. Making use of the semi-discrete form of the convection-diffusion equation,

$$\begin{aligned}\int_{I_j} u(x,t^{n+1})dx &= \int_{\tilde{I}_j(t^n)} u(x,t^n)dx - \int_{t^n}^{t^{n+1}} \left\{ \hat{F}_{j+\frac{1}{2}}(t) - \hat{F}_{j-\frac{1}{2}}(t) - \epsilon \int_{\tilde{I}_j(t)} u_{xx}(x,t)dx \right\} dt \\ &= \int_{\tilde{I}_j(t^n)} u(x,t^n)dx - \int_{t^n}^{t^{n+1}} \Bigg\{ \hat{F}_{j+\frac{1}{2}}(t) - \hat{F}_{j-\frac{1}{2}}(t) \\ &\qquad - \epsilon \left(u_x(\tilde{x}_{j+\frac{1}{2}}(t),t) - u_x(\tilde{x}_{j-\frac{1}{2}}(t),t) \right) \Bigg\} dt. \end{aligned} \tag{2.3.12}$$

Summing over all $j = 1, 2, ..., N_x$ and making use of the periodic boundary conditions,

$$\int_a^b u(x,t^{n+1})dx = \int_a^b u(x,t^n)dx. \tag{2.3.13}$$

□

Remark 2.4. By virtue of approximating the exact characteristics, the CFL condition is much more relaxed when compared to the standard RK-FV method. On the other hand, Algorithm 2.2 is more expensive than a standard WENO procedure on a uniform grid. The computational savings of the EL-RK-FV method will be more significant for solving convection-diffusion equations, as the computational overhead in the remapping algorithm is less significant compared with the implicit solver for the stiff diffusion term. In our computational experiments, we find that the proposed algorithm is most advantageous for convection-dominated problems. Moreover, it is advantageous to take larger time steps with numerical stability for convection-diffusion problems when the convective and diffusive terms are of similar magnitudes.

2.4 Numerical tests

In this section, we present results applying the proposed EL-RK-FV algorithm to various benchmark problems. In particular, we include error tables and error plots to investigate the spatial and temporal convergence of the algorithm. Mass conservation is also numerically verified by applying the proposed algorithm to the 0D2V (zero dimensions in physical space and two dimensions in velocity space) Leonard-Bernstein Fokker-Planck equation. We assume a uniform mesh, apply WENO-AO(5,3) in Algorithms 2.1 and 2.2, use three Gauss-Legendre nodes in Algorithm 2.3, and use the fourth-order approximation given by equation (2.3.4) for the diffusive term. Unless otherwise stated, for the time-stepping we use the fourth-order RK method for pure convection problems, and IMEX(2,3,3) for convection-diffusion equations. We also use second-order Strang splitting for two-dimensional convection-diffusion equations. Although higher-order splitting methods can be used for pure convection problems, it is well-known that negative time integration can lead to significant instabilities when dealing with a diffusive term.

There are three sources of error: spatial approximation, time-stepping, and splitting. Depending on the CFL number and test problem, these three sources of

error will influence the observed order of convergence. We compute the L^1, L^2, and L^∞ errors (in one-dimension),

$$\|\overline{u} - \overline{u}_{exact}\|_1 = \Delta x \sum_{j=1}^{N_x} |\overline{u}_j - \overline{u}_{exact,j}| \tag{2.4.1a}$$

$$\|\overline{u} - \overline{u}_{exact}\|_2 = \sqrt{\Delta x \sum_{j=1}^{N_x} |\overline{u}_j - \overline{u}_{exact,j}|^2} \tag{2.4.1b}$$

$$\|\overline{u} - \overline{u}_{exact}\|_\infty = \max_{1\le j\le N_x} |\overline{u}_j - \overline{u}_{exact,j}| \tag{2.4.1c}$$

Note that for the norms defined above, $\|\overline{u} - \overline{u}_{exact}\|_1 \le |\mathcal{D}| \|\overline{u} - \overline{u}_{exact}\|_\infty$.

2.4.1 Pure convection problems: one-dimensional tests

Example 2.1. (1D transport with constant coefficient)

$$u_t + u_x = 0, \qquad x \in [0, 2\pi] \tag{2.4.2}$$

with periodic boundary conditions and exact solution $u(x,t) = \sin(x-t)$. The errors provided in Table 2.2 verify the convergence of the EL-RK-FV method when using WENO-AO(5,3) and forward Euler. As expected, we see fifth-order convergence despite the large CFL number. There is no temporal error for the convective part since the characteristics are traced exactly and hence $F_{j+\frac{1}{2}}(t) = 0$ for all $t \in [t^n, t^{n+1}]$ and $j = 1, 2, ..., N_x$.

Example 2.2. (1D transport with variable coefficient in space)

$$u_t + (\sin(x)u)_x = 0, \qquad x \in [0, 2\pi] \tag{2.4.3}$$

with periodic boundary conditions and exact solution

$$u(x,t) = \frac{\sin\left(2\arctan\left(e^{-t}\tan(x/2)\right)\right)}{\sin(x)}.$$

Table 2.2: Convergence study with spatial mesh refinement for equation (2.4.2) with forward Euler at $T_f = 1$.

$CFL = 8$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	1.09E-08	-	4.83E-09	-	2.86E-09	-
100	3.34E-10	5.03	1.48E-10	5.03	8.34E-11	5.10
200	9.86E-12	5.08	4.37E-12	5.08	2.51E-12	5.06
400	2.80E-13	5.14	1.43E-13	4.94	1.91E-13	3.72

As seen in Table 2.3, we observe the high-order convergence. As the CFL number (and hence the time step) increases, the temporal error starts to play more of a role, as evidenced by the fourth-order convergence. We verify the high-order temporal convergence in Figure 2.5 by fixing the mesh $N_x = 400$ and varying the CFL from 0.2 to 20.

Table 2.3: Convergence study with spatial mesh refinement for equation (2.4.3) with RK4 at $T_f = 1$.

$CFL = 0.3$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	2.76E-04	-	2.53E-04	-	3.42E-04	-
100	3.05E-06	6.50	2.53E-06	6.64	2.94E-06	6.87
200	9.78E-08	4.96	7.90E-08	5.00	9.86E-08	4.90
400	3.24E-09	4.91	2.59E-09	4.93	3.23E-09	4.93
$CFL = 8$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	6.81E-02	-	4.11E-02	-	4.51E-02	-
100	1.18E-03	5.85	6.23E-04	6.04	6.38E-04	6.15
200	5.89E-05	4.32	3.63E-05	4.10	5.46E-05	3.54
400	3.71E-06	3.99	2.57E-06	3.82	4.31E-06	3.66

Example 2.3. (1D transport with variable coefficient in time)

$$u_t + \left(\frac{u}{t+1}\right)_x = 0, \qquad x \in [0, 2\pi] \tag{2.4.4}$$

with periodic boundary conditions and exact solution $u(x,t) = \exp(-5(x - \log(t+1) - \pi)^2)$. Periodic boundary conditions are a valid assumption for sufficiently thin Gaussian curves and small enough times. The expected high-order convergence for both small and large CFL numbers is seen in Table 2.4. Fixing the mesh $N_x = 400$ and varying the CFL number from 0.2 to 20, fifth-order convergence in time is seen in Figure 2.6. Observe that there are two optimal CFL numbers for this mesh.

Table 2.4: Convergence study with spatial mesh refinement for equation (2.4.4) with RK4 at $T_f = 1$.

$CFL = 0.95$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	6.36E-03	-	5.80E-03	-	8.23E-03	-
100	2.37E-04	4.75	2.28E-04	4.67	5.33E-04	3.95
200	1.71E-06	7.12	1.30E-06	7.46	2.36E-06	7.82
400	4.40E-08	5.28	3.84E-08	5.08	6.02E-08	5.29
$CFL = 8$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	6.71E-02	-	5.72E-02	-	8.15E-02	-
100	7.74E-04	6.44	6.48E-04	6.46	1.06E-03	6.27
200	1.52E-05	5.67	1.26E-05	5.69	1.76E-05	5.91
400	9.65E-07	3.98	8.11E-07	3.96	9.61E-07	4.20

2.4.2 Pure convection problems: two-dimensional tests

Example 2.4. (2D transport with constant coefficient)

$$u_t + u_x + u_y = 0, \qquad x, y \in [-\pi, \pi] \tag{2.4.5}$$

with periodic boundary conditions and exact solution $u(x,y,t) = \sin(x + y - 2t)$. The expected high-order convergence is shown in Table 2.5 when using Strang splitting. Just like equation (2.4.2), there is no temporal error for the convective part since the characteristics are traced exactly.

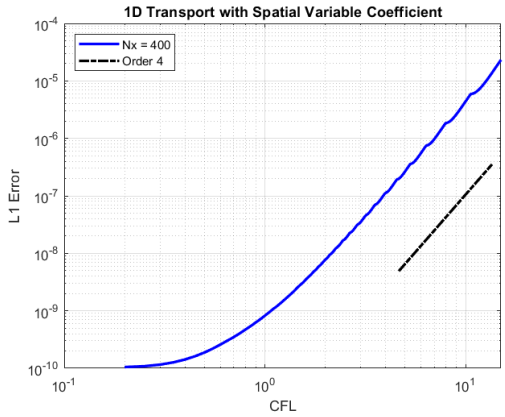


Figure 2.5: Error plot corresponding to equation (2.4.3) using RK4 with final time $T_f = 0.5$.

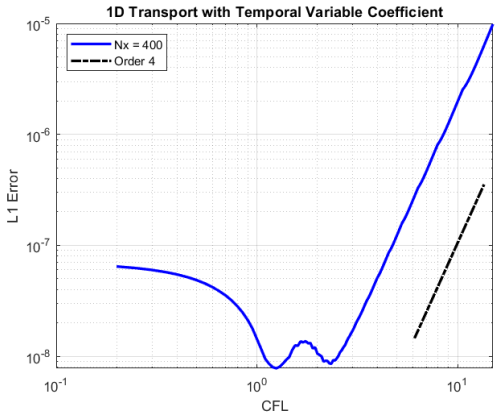


Figure 2.6: Error plot corresponding to equation (2.4.4) using RK4 with final time $T_f = 0.5$.

Table 2.5: Convergence study with spatial mesh refinement for equation (2.4.5) with forward Euler and $CFL = 8$ at $T_f = 1$.

Strang splitting						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
100	4.24E-09	-	7.49E-10	-	1.69E-10	-
200	1.27E-10	5.05	2.25E-11	5.05	5.15E-12	5.03
300	1.60E-11	5.11	2.84E-12	5.11	8.23E-13	4.52
400	3.57E-12	5.21	6.88E-13	4.92	3.68E-13	2.80

Example 2.5. (Rigid body rotation)

$$u_t - yu_x + xu_y = 0, \qquad x, y \in [-\pi, \pi] \tag{2.4.6}$$

with periodic boundary conditions. We choose the exact solution $u(x, y, t) = u(x, y, t = 0) = \exp(-3(x^2 + y^2))$ for convergence tests. The convergence results are presented in Tables 2.6 and 2.7. Strang splitting dominates the error and we observe the expected second-order convergence. Whereas, the spatial error dominates when using fourth-order splitting as evidenced by the fifth-order convergence. The error plot using a fixed mesh $N_x = N_y = 200$ and varying the CFL number from 0.1 to 50 is shown in Figure 2.7. Second-order convergence in time is observed when using Strang splitting, and

fourth-order convergence is observed when using fourth-order splitting. We note that comparable convergence results were observed for the non-symmetric initial condition $u(x, y, t = 0) = \exp(-3x^2 - 2y^2)$. To demonstrate the effectiveness of WENO-AO controlling spurious oscillations, we choose the initial condition $u(x, y, t = 0) = 1$ if $x, y \in [-\pi/2, \pi/2]$; $u(x, y, t = 0) = 0$ otherwise. With a fixed mesh $N_x = N_y = 100$ and $CFL = 2.2$, we compute the solution up to time $T_f = 2\pi$ using SSP RK3. The discontinuities are smoothed out and no spurious oscillations occur.

Table 2.6: Convergence study with spatial mesh refinement for equation (2.4.6) with RK4 and $CFL = 0.95$ at $T_f = 0.5$.

Strang splitting						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
100	3.57E-05	-	1.54E-05	-	2.43E-05	-
200	1.83E-06	4.29	9.94E-07	3.95	1.23E-06	4.31
300	8.01E-07	2.04	4.35E-07	2.04	4.72E-07	2.36
400	4.50E-07	2.01	2.44E-07	2.01	2.55E-07	2.13
Fourth-order splitting						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
100	1.05E-04	-	4.66E-05	-	8.95E-05	-
200	1.08E-06	6.60	6.13E-07	6.25	7.74E-07	6.85
300	1.39E-07	5.06	8.12E-08	4.99	1.02E-07	5.00
400	3.32E-08	4.99	1.93E-08	4.99	2.43E-08	4.99

Example 2.6. (Swirling deformation)

$$u_t - \left(\cos^2(x/2)\sin(y)g(t)u\right)_x + \left(\sin(x)\cos^2(y/2)g(t)u\right)_y = 0, \qquad x, y \in [-\pi, \pi] \tag{2.4.7}$$

When testing convergence we set $g(t) = \cos(\pi t/T_f)\pi$ and choose the initial condition to be the smooth (with C^5 smoothness) cosine bell

$$u(x, y, t = 0) = \begin{cases} r_0^b \cos^6\left(\frac{r^b(x,y)\pi}{2r_0^b}\right), & \text{if } r^b(x, y) < r_0^b, \\ 0, & \text{otherwise,} \end{cases} \tag{2.4.8}$$

Table 2.7: Convergence study with spatial mesh refinement for equation (2.4.6) with RK4 and $CFL = 8$ at $T_f = 0.5$.

Strang splitting						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
100	4.97E-04	-	2.68E-04	-	2.78E-04	-
200	1.24E-04	2.00	6.74E-05	1.99	6.86E-05	2.02
300	5.59E-05	1.97	3.03E-05	1.97	3.08E-05	1.97
400	3.20E-05	1.94	1.73E-05	1.94	1.76E-05	1.94
Fourth-order splitting						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
100	4.95E-05	-	2.46E-05	-	6.82E-05	-
200	4.84E-07	6.68	2.92E-07	6.39	4.77E-07	7.16
300	6.17E-08	5.08	3.86E-08	4.99	6.19E-08	5.03
400	1.47E-08	4.99	9.22E-09	4.88	1.47E-08	5.00

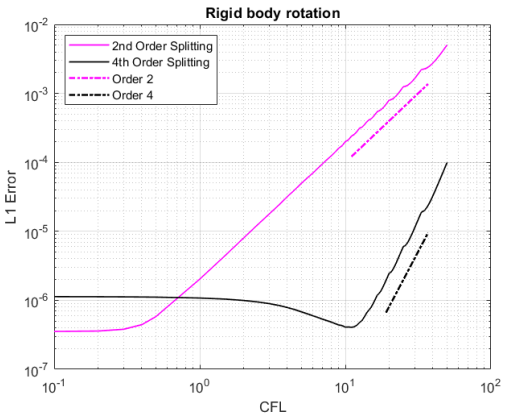


Figure 2.7: Error plot for (2.4.6) with RK4 at $T_f = 0.5$. $N_x = N_y = 200$.

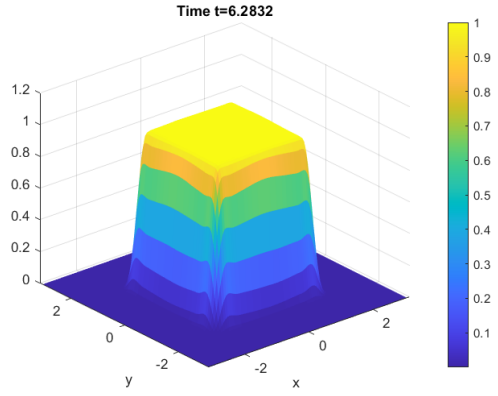


Figure 2.8: Plot of the numerical solution to (2.4.6) with SSP RK3 and $CFL = 2.2$ at $T_f = 2\pi$. $N_x = N_y = 100$.

where $r_0^b = 0.3\pi$ and $r^b(x, y) = \sqrt{(x - x_0^b)^2 + (y - y_0^b)^2}$ with $(x_0^b, y_0^b) = (0.3\pi, 0)$. The convergence results under spatial mesh refinement are presented in Tables 2.8 and 2.9. Surprisingly, high-order convergence is observed in all test cases, even for the large CFL number of 8. In particular, we observed super-convergence for $CFL = 8$ when using Strang splitting. We got comparable convergence results when letting the initial condition be: (1) a cosine bell of C^3 smoothness, and (2) the cosine bell (2.4.8) but

with $x_0^b = 0.6\pi$ and the width in the $x-$direction scaled by a factor of $1/2$.

The temporal orders of convergence are shown in Figure 2.9 using a fixed mesh $N_x = N_y = 200$ and varying the CFL number from 0.1 to 25. When using Strang splitting the temporal convergence switches from second-order to third-order, indicating that for very large CFL numbers the splitting error does not dominate the time-stepping error as much. Fourth-order convergence is observed when using fourth-order splitting. To demonstrate the effectiveness of WENO-AO in controlling spurious oscillations we set $g(t) = 1$ and choose the initial condition [117]

$$u(x, y, t = 0) = \begin{cases} 1, & \text{if } r^b(x, y) < r_0^b, \\ 0, & \text{otherwise,} \end{cases} \tag{2.4.9}$$

where $r_0^b = 8\pi/5$ and $r^b(x, y) = \sqrt{(x - x_0^b)^2 + (y - y_0^b)^2}$ with $(x_0^b, y_0^b) = (\pi, \pi)$. With a fixed mesh $N_x = N_y = 100$ and $CFL = 8$, we compute the solution up to time $T_f = 5\pi$ using SSP RK3 and Strang splitting. The discontinuities are smoothed out and no spurious oscillations occur.

Table 2.8: Convergence study with spatial mesh refinement for equation (2.4.7) with RK4 and $CFL = 0.95$ at $T_f = 1.5$.

Strang splitting						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
100	5.17E-03	-	6.11E-03	-	2.04E-02	-
200	1.69E-04	4.94	1.69E-04	5.18	4.80E-04	5.41
300	3.12E-05	4.16	3.85E-05	3.64	1.41E-04	3.01
400	8.29E-06	4.61	1.12E-05	4.66	4.66E-05	3.86
Fourth-order splitting						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
100	1.42E-02	-	1.73E-02	-	5.25E-02	-
200	3.98E-04	5.16	3.70E-04	5.54	9.15E-04	5.84
300	7.69E-05	4.06	9.02E-05	3.48	3.24E-04	2.56
400	2.20E-05	4.35	2.89E-05	3.96	1.17E-04	3.52

Table 2.9: Convergence study with spatial mesh refinement for equation (2.4.7) with RK4 and $CFL = 8$ at $T_f = 1.5$.

Strang splitting						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
100	1.90E-03	-	2.11E-03	-	6.83E-03	-
200	1.02E-04	4.23	8.88E-05	4.57	2.47E-04	4.79
300	1.90E-05	4.13	1.79E-05	3.94	6.36E-05	3.35
400	2.82E-06	6.63	4.11E-06	5.12	1.98E-05	4.05
Fourth-order splitting						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
100	3.89E-03	-	4.31E-03	-	1.43E-02	-
200	1.30E-04	4.90	1.34E-04	5.01	4.42E-04	5.01
300	2.41E-05	4.16	3.32E-05	3.44	1.40E-04	2.84
400	6.54E-06	4.53	1.02E-05	4.10	4.81E-05	3.70

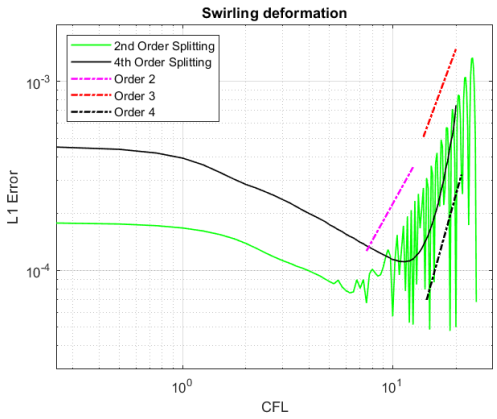


Figure 2.9: Error plot for (2.4.7) with RK4 at $T_f = 1.5$. $N_x = N_y = 200$.

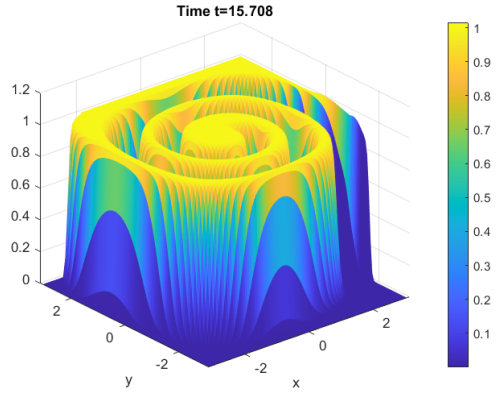


Figure 2.10: Plot of the numerical solution to (2.4.7) with $g(t) = 1$, SSP RK3 and $CFL = 8$ at $T_f = 5\pi$. $N_x = N_y = 100$.

2.4.3 Convection-diffusion equations: one-dimensional tests

Example 2.7. (1D equation with constant coefficient)

$$u_t + u_x = \epsilon u_{xx}, \qquad x \in [0, 2\pi] \tag{2.4.10}$$

with periodic boundary conditions and exact solution $u(x,t) = \sin(x-t)\exp(-\epsilon t)$. We set $\epsilon = 1$. The convergence results under spatial mesh refinement are shown in Table 2.10 for $CFL = 0.95$ and $CFL = 8$. In both cases we observe the expected third-order convergence since we are using IMEX(2,3,3) for the time-stepping. Note that there is no temporal error for the convective part since the characteristics are traced exactly and hence $F_{j+\frac{1}{2}}(t) = 0$ for all $t \in [t^n, t^{n+1}]$ and $j = 1, 2, ..., N_x$. Figure 2.11 shows the expected third-order convergence in time using fixed mesh $N_x = 400$ and varying the CFL number from 0.1 to 15.

Table 2.10: Convergence study with spatial mesh refinement for equation (2.4.10) with IMEX(2,3,3) at $T_f = 1$.

$CFL = 0.95$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	1.87E-04	-	8.26E-05	-	4.66E-05	-
100	2.55E-05	2.87	1.13E-05	2.87	6.36E-06	2.87
200	3.34E-06	2.93	1.48E-06	2.93	8.35E-07	2.93
400	4.31E-07	2.95	1.91E-07	2.95	1.08E-07	2.95
800	5.44E-08	2.99	2.41E-08	2.99	1.36E-08	2.99
$CFL = 8$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	6.87E-02	-	3.04E-02	-	1.72E-02	-
100	1.09E-02	2.66	4.82E-03	2.66	2.72E-03	2.66
200	1.63E-03	2.74	7.22E-04	2.74	4.07E-04	2.74
400	2.27E-04	2.84	1.01E-04	2.84	5.69E-05	2.84
800	3.03E-05	2.91	1.34E-05	2.91	7.57E-06	2.91

Example 2.8. (1D equation with variable coefficient)

$$u_t + (\sin(x)u)_x = \epsilon u_{xx} + g, \qquad x \in [0, 2\pi] \tag{2.4.11}$$

with periodic boundary conditions and $g(x,t) = \sin(2x)\exp(-\epsilon t)$ and exact solution $u(x,t) = \sin(x)\exp(-\epsilon t)$. We set $\epsilon = 1$. Table 2.11 shows the convergence results under spatial mesh refinement. Third-order convergence in space is observed for $CFL = 0.95$.

Whereas, the convergence for $CFL = 8$ is roughly order 2.6 since the time-stepping start to dominate. We note that the order of convergence for IMEX(2,3,3) under increasing the CFL number dips slightly below three for larger CFL numbers. We use fixed mesh $N_x = 400$ and vary the CFL number from 0.1 to 15 for the error plot showing the temporal order of convergence in Figure 2.12.

Table 2.11: Convergence study with spatial mesh refinement for equation (2.4.11) with IMEX(2,3,3) at $T_f = 1$.

$CFL = 0.95$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	2.10E-03	-	9.99E-04	-	7.70E-04	-
100	3.99E-04	2.39	1.88E-04	2.41	1.44E-04	2.42
200	6.41E-05	2.64	2.99E-05	2.65	2.27E-05	2.66
400	9.84E-06	2.70	4.57E-06	2.71	3.44E-06	2.73
800	1.30E-06	2.92	6.04E-07	2.92	4.53E-07	2.92
$CFL = 8$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	7.50E-01	-	4.34E-01	-	3.92E-01	-
100	1.26E-01	2.58	6.60E-02	2.72	6.63E-02	2.56
200	1.74E-02	2.85	8.31E-03	2.99	6.50E-03	3.35
400	3.09E-03	2.50	1.46E-03	2.51	1.12E-03	2.53
800	5.03E-04	2.62	2.36E-04	2.63	1.81E-04	2.63

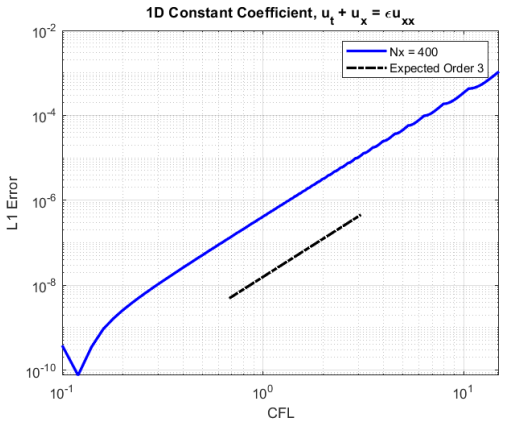


Figure 2.11: IMEX(2,3,3), $\epsilon = 1$, Final time $T_f = 0.5$.

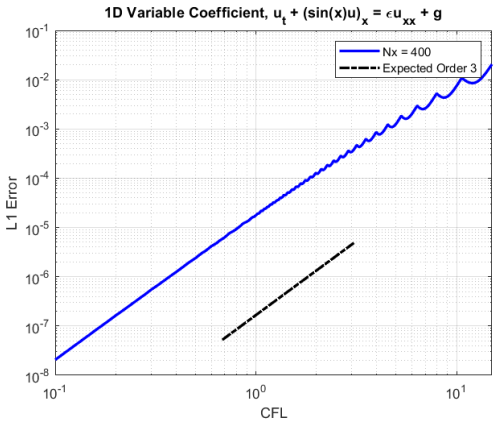


Figure 2.12: IMEX(2,3,3), $\epsilon = 1$, Final time $T_f = 0.5$.

Example 2.9. (1D viscous Burgers' equation)

$$u_t + \left(\frac{u^2}{2}\right)_x = \epsilon u_{xx}, \qquad x \in [0, 2] \tag{2.4.12}$$

with periodic boundary conditions. As in [153], we set $\epsilon = 0.1$ and choose the initial condition $u(x, t = 0) = 0.2 \sin(\pi x)$. The exact solution is

$$u(x,t) = 2\epsilon\pi \frac{\sum_{n=1}^{\infty} c_n \exp(-n^2\pi^2\epsilon t) n \sin(n\pi x)}{c_0 + \sum_{n=1}^{\infty} c_n \exp(-n^2\pi^2\epsilon t)\cos(n\pi x)},$$

where

$$c_0 = \int_0^1 \exp(-(1 - \cos(\pi x))/(10\pi\epsilon))dx$$

and

$$c_n = 2\int_0^1 \exp(-(1 - \cos(\pi x))/(10\pi\epsilon))\cos(n\pi x)dx \quad \text{for } n = 1, 2, 3, ...$$

We computed the first ten Fourier coefficients in Mathematica® for the exact solution; the eleventh Fourier coefficient was less than machine precision.

Table 2.12 shows the expected orders of convergence under spatial mesh refinement. Third-order convergence is observed for $CFL = 0.95$. Whereas, the convergence for $CFL = 8$ is slightly below order three since the time-stepping error starts to dominate. W note that the order of convergence for IMEX(2,3,3) under increasing the CFL number dips slightly below three for larger CFL numbers. The error plot in Figure 2.13 showing third-order convergence in time uses mesh $N_x = 400$ and CFL numbers varying from 0.1 to 15.

Example 2.10. (The 0D1V Leonard-Bernstein (linearized) Fokker-Planck equation)

$$f_t - \frac{1}{\epsilon}((v_x - \overline{v}_x)f)_{v_x} = \frac{1}{\epsilon} D f_{v_x v_x}, \qquad v_x \in [-2\pi, 2\pi] \tag{2.4.13}$$

Table 2.12: Convergence study with spatial mesh refinement for equation (2.4.12) with IMEX(2,3,3) at $T_f = 1$.

$CFL = 0.95$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	9.50E-05	-	8.43E-05	-	1.17E-04	-
100	1.48E-05	2.68	1.32E-05	3.67	1.87E-05	2.65
200	2.42E-06	2.62	2.20E-06	2.59	3.21E-06	2.55
400	3.27E-07	2.88	3.00E-07	2.87	4.42E-07	2.86
800	4.28E-08	2.94	3.95E-08	2.93	5.86E-08	2.92
$CFL = 8$						
N_x	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	1.26E-02	-	1.31E-02	-	2.24E-02	-
100	2.79E-03	2.17	2.50E-03	2.38	3.70E-03	2.60
200	4.73E-04	2.56	4.16E-04	2.59	5.64E-04	2.71
400	1.28E-04	1.89	1.15E-04	1.85	1.64E-04	1.78
800	1.97E-05	2.70	1.78E-05	2.70	2.56E-05	2.68

with zero boundary conditions and equilibrium solution the Maxwellian

$$f_M(v_x) = \frac{n}{\sqrt{2\pi RT}} \exp\left(-\frac{(v_x - \overline{v}_x)^2}{2RT}\right), \tag{2.4.14}$$

where $\epsilon = 1$, gas constant $R = 1/6$, temperature $T = 3$, thermal velocity $v_{th} = \sqrt{2RT} = \sqrt{2D} = 1$, number density $n = \pi$, and bulk velocity $\overline{v}_x = 0$. These quantities were chosen for scaling convenience. When testing convergence we set the initial distribution $f(v_x, t = 0) = f_M(v_x)$. Table 2.13 shows the convergence results, for which we use IMEX(4,4,3) for the time-stepping; we show the results using IMEX(4,4,3) because it gave slightly better convergence than IMEX(2,3,3). We observe fourth-order convergence under spatial mesh refinement for $CFL = 0.95$. Whereas, for $CFL = 8$ the time-stepping error starts to dominate and we observe third-order convergence. The error plot in Figure 2.14 showing third-order convergence in time uses a fixed mesh $N_{v_x} = 400$ and CFL numbers varying from 0.1 to 15. We note that although high-order convergence is observed, the proposed EL-RK-FV algorithm is not equilibrium-preserving.

Table 2.13: Convergence study with spatial mesh refinement for equation (2.4.13) with IMEX(4,4,3) at $T_f = 1$.

$CFL = 0.95$						
N_{v_x}	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	8.02E-04	-	5.19E-04	-	5.65E-04	-
100	6.12E-05	3.71	3.83E-05	3.76	4.27E-05	3.73
200	4.41E-06	3.79	2.63E-06	3.87	2.84E-06	3.91
400	2.99E-07	3.88	1.73E-07	3.93	1.80E-07	3.98
800	2.03E-08	3.88	1.13E-08	3.94	1.10E-08	4.04
$CFL = 8$						
N_{v_x}	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	9.33E-03	-	4.52E-03	-	3.63E-03	-
100	1.34E-03	2.80	6.60E-04	2.78	5.63E-04	2.69
200	1.91E-04	2.81	9.57E-05	2.79	8.15E-05	2.79
400	3.21E-05	2.57	1.61E-05	2.57	1.34E-05	2.61
800	4.13E-06	2.96	2.09E-06	2.95	1.74E-06	2.94

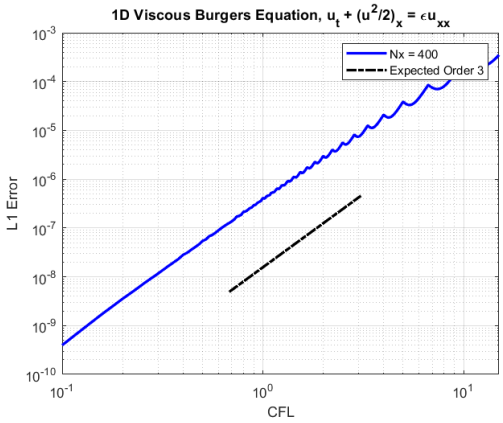


Figure 2.13: IMEX(2,3,3), $\epsilon = 0.1$, Final time $T_f = 0.5$.

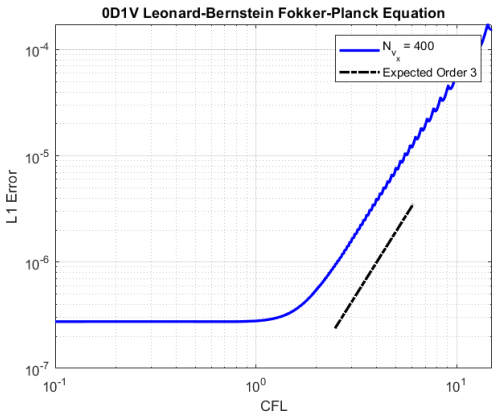


Figure 2.14: IMEX(4,4,3), Final time $T_f = 0.5$.

2.4.4 Convection-diffusion equations: two-dimensional tests

Example 2.11. (2D equation with constant coefficient)

$$u_t + u_x + u_y = \epsilon(u_{xx} + u_{yy}), \qquad x, y \in [0, 2\pi] \tag{2.4.15}$$

with periodic boundary conditions and exact solution $u(x, y, t) = \exp(-2\epsilon t)\sin(x + y - 2t)$. We set $\epsilon = 1$. Third-order convergence under spatial mesh refinement is seen in Table 2.14 for $CFL = 0.95$ and $CFL = 8$. As with equation (2.4.10), there is no temporal error for the convective part since the characteristics are traced exactly. Note that the error is larger for $CFL = 8$ than $CFL = 0.95$ since this problem also has diffusion. Figure 2.15 shows the third-order convergence in time using fixed mesh $N_x = N_y = 400$ and varying the CFL number from 6 to 20.

Table 2.14: Convergence study with spatial mesh refinement for equation (2.4.15) with IMEX(2,3,3) and Strang splitting at $T_f = 0.5$.

$CFL = 0.95$						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	6.74E-05	-	1.19E-05	-	2.68E-06	-
100	1.02E-05	2.72	1.81E-06	2.72	4.07E-07	2.72
200	1.41E-06	2.86	2.49E-07	2.86	5.62E-08	2.86
400	1.86E-07	2.92	3.29E-08	2.92	7.41E-09	2.92
$CFL = 8$						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	3.92E-02	-	6.94E-03	-	1.56E-03	-
100	5.82E-03	2.75	1.03E-03	2.75	2.32E-04	2.75
200	8.09E-04	2.85	1.43E-04	2.85	3.22E-05	2.85
400	1.07E-04	2.92	1.90E-05	2.92	4.27E-06	2.92

Example 2.12. (Rigid body rotation with diffusion)

$$u_t - yu_x + xu_y = \epsilon(u_{xx} + u_{yy}) + g, \qquad x, y \in [-2\pi, 2\pi] \tag{2.4.16}$$

with periodic boundary conditions, $g(x, y, t) = (6\epsilon - 4xy - 4\epsilon(x^2 + 9y^2))\exp(-(x^2 + 3y^2 + 2\epsilon t))$, and exact solution $u(x, y, t) = \exp(-(x^2 + 3y^2 + 2\epsilon t))$. We set $\epsilon = 1$. Table 2.15 shows the order of convergence when using IMEX(4,4,3). We use IMEX(4,4,3) instead of IMEX(2,3,3) because the latter choice, along with the Strang splitting, showed an order of convergence less than two for large CFL numbers. The expected second-order

convergence in time (due to Strang splitting) is seen in Figure 2.16 assumes fixed mesh $N_x = N_y = 400$ and CFL numbers varying from 6 to 20.

Table 2.15: Convergence study with spatial mesh refinement for equation (2.4.16) with IMEX(4,4,3) and Strang splitting at $T_f = 0.5$.

$CFL = 0.95$						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	3.22E-03	-	1.42E-03	-	1.44E-03	-
100	3.92E-04	3.04	1.75E-04	3.02	2.00E-04	2.85
200	7.27E-05	2.43	3.14E-05	2.47	3.54E-05	2.50
400	1.65E-05	2.14	7.11E-06	2.15	7.80E-06	2.18
$CFL = 5$						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	2.21E-02	-	8.81E-03	-	7.44E-03	-
100	5.98E-03	1.89	2.48E-03	1.83	2.46E-03	1.60
200	1.61E-03	1.89	6.81E-04	1.86	7.14E-04	1.79
400	4.24E-04	1.93	1.81E-04	1.91	1.94E-04	1.88

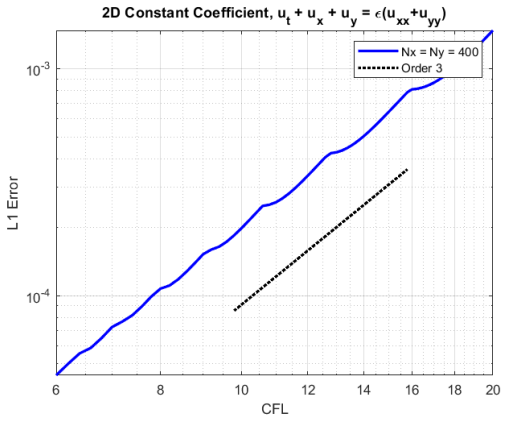


Figure 2.15: IMEX(2,3,3), $\epsilon = 1$, Final time $T_f = 0.5$.

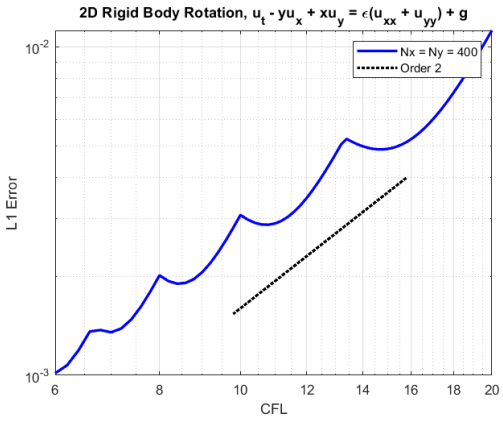


Figure 2.16: IMEX(4,4,3), $\epsilon = 1$, Final time $T_f = 0.1$.

Example 2.13. (Swirling deformation with diffusion)

$$u_t - \left(\cos^2(x/2)\sin(y)f(t)u\right)_x + \left(\sin(x)\cos^2(y/2)f(t)u\right)_y = \epsilon(u_{xx}+u_{yy}), \qquad x, y \in [-\pi, \pi] \tag{2.4.17}$$

When testing the convergence we set $f(t) = \cos(\pi t/T_f)\pi$, $\epsilon = 1$, and choose the initial condition to be the cosine bell in equation (2.4.8). Since there is no analytic solution, we use a reference solution computed with a mesh size of 400×400 and $CFL = 0.1$. The convergence results under spatial mesh refinement are presented in Table 2.16. The splitting error seems to dominate the time-stepping error for $CFL = 0.95$ as evidenced by the apparent second-order convergence. Whereas, the time-stepping error seems to contribute more for $CFL = 8$. Due to the interplay between the time-stepping and splitting errors, the temporal order 2.4 is also observed in Figure 2.17, for which we use fixed mesh $N_x = N_y = 400$ and CFL numbers varying from 6 to 20.

Table 2.16: Convergence study with spatial mesh refinement for equation (2.4.17) with IMEX(2,3,3) and Strang splitting at $T_f = 0.1$.

$CFL = 0.95$						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	4.98E-04	-	2.57E-04	-	3.38E-04	-
100	9.13E-05	2.45	4.58E-05	2.49	6.05E-05	2.48
200	1.91E-05	2.26	9.56E-06	2.26	1.22E-05	2.31
400	4.48E-06	2.09	2.19E-06	2.13	2.61E-06	2.23
$CFL = 8$						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	5.93E-02	-	3.60E-02	-	5.82E-02	-
100	2.05E-02	1.53	1.16E-02	1.63	1.65E-02	1.82
200	3.14E-03	2.71	1.67E-03	2.80	2.04E-03	3.01
400	6.08E-04	2.37	3.13E-04	2.41	4.16E-04	2.30

Example 2.14. (2D viscous Burgers' equation)

$$u_t + \left(\frac{u^2}{2}\right)_x + \left(\frac{u^2}{2}\right)_y = \epsilon(u_{xx} + u_{yy}) + g, \qquad x, y \in [-\pi, \pi] \tag{2.4.18}$$

with periodic boundary conditions. As in [177], we set $\epsilon = 0.1$, $g(x, y, t) = \exp(-4\epsilon t)\sin(2(x + y))$, and suppose the solution $u(x, y, t) = \exp(-2\epsilon t)\sin(x + y)$. The convergence results are presented in Table 2.17. The splitting error seems to dominate the time-stepping error

for $CFL = 0.95$ as evidenced by the second-order convergence. Whereas, the time-stepping error seems to contribute more for $CFL = 8$ since the order is between two and three. The temporal order of convergence in the L^1 norm is roughly 2.3, as seen in Figure 2.18, for which we use fixed mesh $N_x = N_y = 400$ and CFL numbers varying from 6 to 20.

Table 2.17: Convergence study with spatial mesh refinement for equation (2.4.18) with IMEX(2,3,3) and Strang splitting at $T_f = 0.5$.

$CFL = 0.95$						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	3.01E-04	-	5.21E-05	-	1.19E-05	-
100	7.12E-05	2.08	1.30E-05	2.00	3.42E-06	1.80
200	1.76E-05	2.02	3.34E-06	1.96	9.42E-07	1.86
400	4.53E-06	1.95	8.60E-07	1.96	2.49E-07	1.92
$CFL = 8$						
$N_x = N_y$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	8.11E-02	-	1.60E-02	-	5.80E-03	-
100	1.02E-02	2.99	2.00E-03	3.00	7.78E-04	2.90
200	1.61E-03	2.66	3.01E-04	2.74	9.81E-05	2.99
400	3.47E-04	2.21	5.97E-05	2.33	1.35E-05	2.86

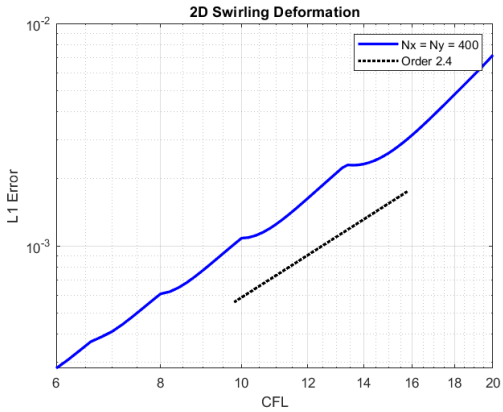


Figure 2.17: IMEX(2,3,3), $\epsilon = 1$, Final time $T_f = 0.1$.

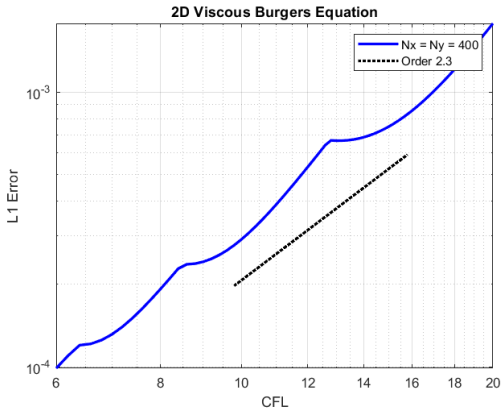


Figure 2.18: IMEX(2,3,3), $\epsilon = 0.1$, Final time $T_f = 0.2$.

Example 2.15. (The 0D2V Leonard-Bernstein (linearized) Fokker-Planck equation)

$$f_t - \frac{1}{\epsilon}((v_x - \overline{v}_x)f)_{v_x} - \frac{1}{\epsilon}((v_y - \overline{v}_y)f)_{v_y} = \frac{1}{\epsilon}D(f_{v_xv_x} + f_{v_yv_y}), \qquad v_x, v_y \in [-2\pi, 2\pi] \tag{2.4.19}$$

with zero boundary conditions and equilibrium solution the Maxwellian

$$f_M(v_x, v_y) = \frac{n}{2\pi RT}\exp\left(-\frac{(v_x - \overline{v}_x)^2 + (v_y - \overline{v}_y)^2}{2RT}\right), \tag{2.4.20}$$

where $\epsilon = 1$, gas constant $R = 1/6$, temperature $T = 3$, thermal velocity $v_{th} = \sqrt{2RT} = \sqrt{2D} = 1$, number density $n = \pi$, and bulk velocities $\overline{v}_x = \overline{v}_y = 0$. These quantities were chosen for scaling convenience. When testing the spatial and temporal orders of accuracy we set the initial distribution $f(v_x, v_y, t = 0) = f_M(v_x, v_y)$. Table 2.19 shows the convergence results under spatial mesh refinement. We observe higher fourth-order convergence in space for $CFL = 0.95$. The time-stepping and splitting errors start to dominate the spatial error for larger CFL numbers, as observed for $CFL = 8$. Figure 2.19 shows the temporal order of convergence is roughly 2.6 for fixed mesh $N_{v_x} = N_{v_y} = 400$ and CFL numbers varying from 6 to 20. We again note the interplay between the third-order time-stepping and second-order splitting.

When testing for relaxation of the system, we choose the initial distribution $f(v_x, v_y, t = 0) = f_{M1}(v_x, v_y) + f_{M2}(v_x, v_y)$, that is, the sum of two randomly generated Maxwellians such that the total macro-parameters are preserved. The number density, bulk velocities, and temperature of each Maxwellian are listed in Table 2.18. We set $\overline{v}_y = 0$ so that the two generated Maxwellians are shifted only along the v_x axis.

	f_{M1}	f_{M2}
n	1.990964530353041	1.150628123236752
$\overline{v}_x$	0.4979792385268875	-0.8616676237412346
$\overline{v}_y$	0	0
T	2.46518981703837	0.4107062104302872

Table 2.18: $n = \pi$, $\overline{\mathbf{v}} = \mathbf{0}$, and $T = 3$.

Ideally, the macro-parameters we want to conserve are number density (n), momentum ($n\overline{\mathbf{v}}$), and energy ($\frac{1}{2}n\overline{\mathbf{v}}^2 + nT$), where in two dimensions we define

$$n = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\mathbf{v})dv_y dv_x, \tag{2.4.21a}$$

$$\overline{\mathbf{v}} = \frac{1}{n}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbf{v}f(\mathbf{v})dv_y dv_x, \tag{2.4.21b}$$

$$T = \frac{1}{2Rn}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\mathbf{v}-\overline{\mathbf{v}})^2 f(\mathbf{v})dv_y dv_x. \tag{2.4.21c}$$

It is important to note that although temperature might look like a conserved quantity since $\overline{\mathbf{v}} = \mathbf{0}$, and hence energy reduces to nT, in fact it is not. Figures 2.20(f) and 2.21 show the solution using fixed mesh $N_{v_x} = N_{v_y} = 200$ and $CFL = 6$. Although we computed the solution up to time $T_f = 20$, there was no difference (to the naked eye) after time $t = 3$. Although the solution appears to reach equilibrium, we again note that the proposed EL-RK-FV algorithm is not equilibrium-preserving. Figure 2.20(a) verifies mass conservation, but Figure 2.20(b) implies that the numerical solution has some negative values and is not positivity-preserving. Referring to Figure 2.20, momentum and energy are not conserved. As seen in Figure 2.20(d), the bulk velocity in the v_y-direction is on the order of machine epsilon because we constructed the two Maxwellians in Table 2.18 such that $\overline{v}_{M1,y} = \overline{v}_{M2,y} = 0$. Hence, there is no drift velocity in the v_y-direction.

2.5 Conclusions and follow-up work

A new EL-RK-FV method was presented for solving convection and convection-diffusion equations [38, 133]. Whereas SL methods require solving for the exact characteristics, which is often highly nontrivial for nonlinear problems, our EL method computes linear space-time curves as the approximate characteristics. WENO-AO schemes allowed us to perform spatial reconstruction at arbitrary points which was essential since the traceback grid was not necessarily the (uniform) background grid. By working with the time-differential form, we could use a method-of-lines approach.

Table 2.19: Convergence study with spatial mesh refinement for equation (2.4.19) with IMEX(2,3,3) and Strang splitting at $T_f = 0.5$.

$CFL = 0.95$						
$N_{v_x} = N_{v_y}$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	9.07E-04	-	4.22E-04	-	5.49E-04	-
100	7.19E-05	3.66	3.15E-05	3.74	4.36E-05	3.66
200	5.35E-06	3.75	2.15E-06	3.87	2.93E-06	3.89
400	3.54E-07	3.92	1.37E-07	3.98	1.82E-07	4.01
$CFL = 8$						
$N_{v_x} = N_{v_y}$	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
50	5.70E-03	-	1.84E-03	-	1.26E-03	-
100	1.08E-03	2.40	3.53E-04	2.39	2.82E-04	2.16
200	1.69E-04	2.67	5.68E-05	2.64	5.02E-05	2.49
400	2.73E-05	2.63	9.30E-06	2.61	8.63E-06	2.54

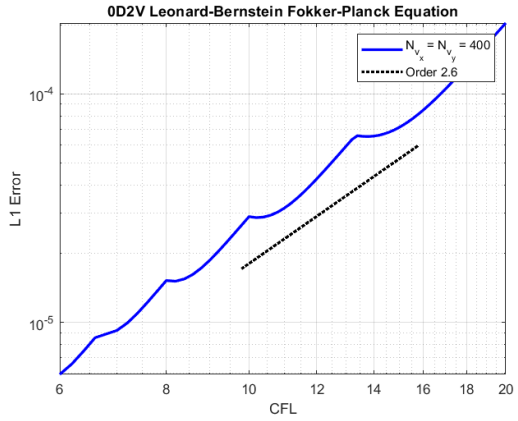


Figure 2.19: IMEX(2,3,3), Final time $T_f = 0.1$.

Explicit RK methods were used for pure convection problems, and IMEX RK methods were used for convection-diffusion equations. Dimensional splitting was used for higher-dimensional problems. Several one- and two-dimensional test problems demonstrated the algorithm's robustness, high-order accuracy, and ability to allow extra large time steps. Ongoing and future work includes modifying the algorithm to handle shocks and rarefaction waves [40, 180], and developing a non-splitting version of the EL-RK-FV algorithm.

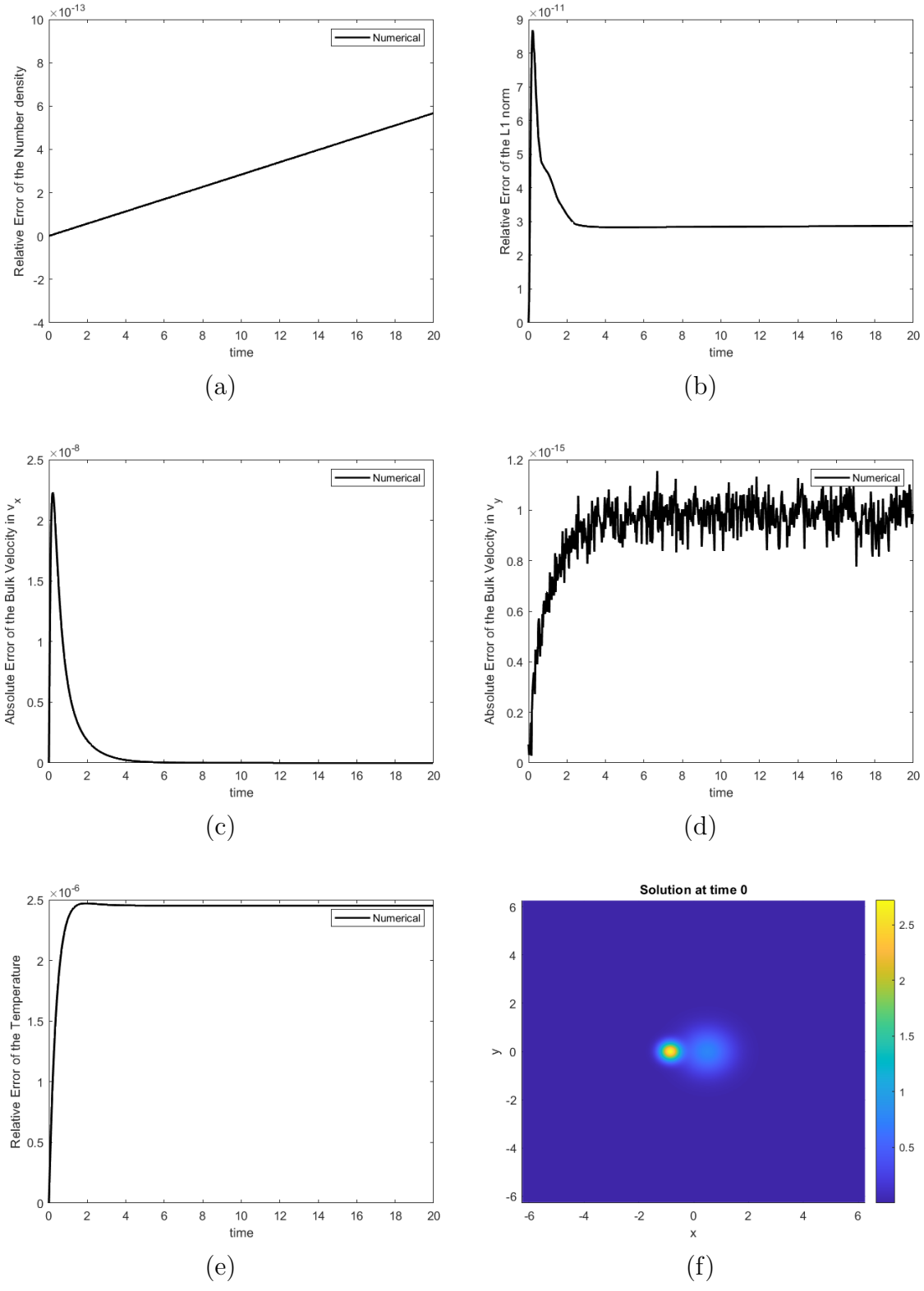


Figure 2.20: Figures (a)-(e): Relative macro-parameters for equation (2.4.19) with initial distribution of two Maxwellians defined by Table 2.18. Mesh $N_{v_x} = N_{v_y} = 200$, $CFL = 6$. Figure (f): The initial distribution.

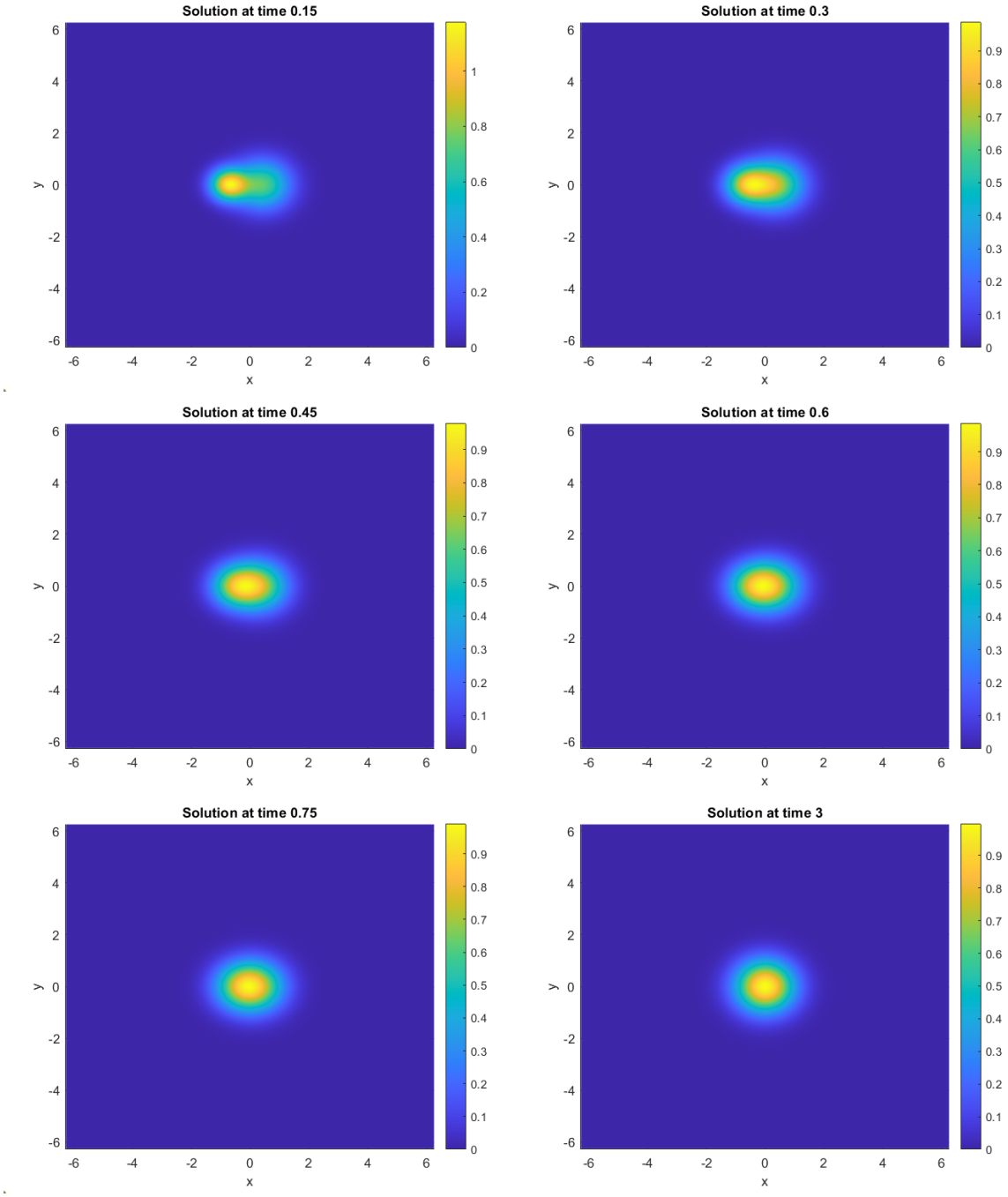


Figure 2.21: Various snapshots of the numerical solution to equation (2.4.19) with initial distribution of two Maxwellians defined by Table 2.18. Mesh $N_{v_x} = N_{v_y} = 200$, $CFL = 6$. Times: 0.15, 0.30, 0.45, 0.60, 0.75, 3.

Chapter 3

IMPLICIT LOW-RANK INTEGRATORS FOR SOLVING DIFFUSION EQUATIONS

In this chapter, we are concerned with efficiently solving diffusion equations of the form

$$\begin{cases} u_t = \nabla \cdot (\mathbf{D} \cdot \nabla u), & \mathbf{x} \in \Omega, \quad t > 0, \\ u(\mathbf{x}, t = 0) = u_0(\mathbf{x}), & \mathbf{x} \in \Omega, \end{cases} \tag{3.0.1}$$

where $u(\mathbf{x}, t)$ is the solution, $\Omega \subset \mathbb{R}^d$ is the spatial domain, $d \in \mathbb{N}$ is the number of dimensions, and $\mathbf{D}$ is the anisotropic diffusion tensor. Within this chapter we only deal with the two-dimensional case in Cartesian coordinates in which $\mathbf{D}$ is diagonal,

$$u_t = d_1^2 u_{xx} + d_2^2 u_{yy}, \qquad (x, y) \in (0, 1)^2, \qquad t > 0, \tag{3.0.2}$$

for constants $d_1, d_2 > 0$.

A novel implicit low-rank method for solving diffusion equations is presented. Traditional implicit time-integrators are used to evolve the solution. The main novelty of the proposed method is that we represent the time-dependent solution in a low-rank framework and strategically evolve the low-rank decomposition of the solution based on traditional implicit time-integrators. In particular, the solution is decomposed into one-dimensional time-dependent bases connected by time-dependent coefficients. Our unique strategy is to evolve these one-dimensional bases in a dimension-by-dimension fashion. The target dimension basis is updated by first freezing and correspondingly projecting the solution in all the non-target dimensions. Once the bases from all dimensions are updated, we then evolve the coefficients be a projection onto the subspace

spanned by the updated bases in all dimensions. Finally, a SVD type truncation is applied to further compress the solution for optimal computational efficiency. Backward Euler method is used for the first-order scheme. Second-order schemes are also presented using second-order stiffly-accurate diagonally implicit Runge-Kutta methods, Crank-Nicolson method, and second-order backward differentiation formula.

The chapter is organized as follows. Section 3.1 reviews the technical material required to understand the proposed method (e.g., matrix and tensor decompositions, low-rank tensor approaches for time-dependent PDEs, von Neumann stability analysis). Section 3.2 outlines the proposed method, including the first-order scheme, second-order scheme and computational complexity. Section 3.3 presents the numerical results verifying convergence and efficiency. Section 3.4 ends with concluding remarks and ongoing work.

3.1 Review of technical components

In this section, we review three technical components for the development of low-rank time integrators: tensor decompositions, low-rank tensor approaches for time-dependent PDEs, and von Neumann stability analysis. The matrix and tensor decompositions presented in this section demonstrate how data can be compressed if there is low-rank structure. The two presented low-rank tensor approaches for time-dependent PDEs offer a foundation for understanding the proposed method. Lastly, the von Neumann analysis is used to assess the stability and predicted solution behavior of various implicit time-discretizations.

3.1.1 Tensor decompositions

Large-scale scientific simulations pose several challenges when storing, computing and analyzing the avalanche of data they produce. In our case, data takes the form of a solution to a high-dimensional partial differential equation, $u(x_1, ..., x_d)$. Most classical techniques for two-dimensional equations store the solution in two-dimensional arrays, or rather, order-2 tensors (i.e., matrices). When solving a high-dimensional

equation, the solution can naturally be stored as a high-order tensor. An **order-d tensor $\mathscr{X}$** can be thought of as a $d-$index array.

Seeing how quickly memory can become an issue, this necessitates tensor decompositions that can compress the data and reduce the storage requirement and computational complexity [106, 109, 112, 135, 174]. A good literature survey of some developments in low-rank tensor approximations for scientific computing is [82, 88, 109].

In the following subsections, we review two classical matrix decompositions typically taught in a numerical linear algebra course, followed by a popular high-order tensor decomposition. We present the material in the context of storing a solution $u(x_1, ..., x_d)$.

3.1.1.1 Singular value decomposition (SVD)

The material from this subsection is predominantly taken from [76, 172]. Consider a solution $u(x, y)$ stored as a matrix $\mathbf{U} \in \mathbb{R}^{N_x \times N_y}$, where $U_{ij} = u(x_i, y_j)$ for $i = 1, 2, ..., N_x$ and $j = 1, 2, ..., N_y$. Assume for now that we know the rank of $\mathbf{U}$ is $r \leq \min(N_x, N_y)$.

Informally speaking, we can find an orthonormal basis for colspace($\mathbf{U}$), say $\{\mathbf{v}_{x,1}, \mathbf{v}_{x,2}, ..., \mathbf{v}_{x,r}\}$, ordered from "most important vector" to "least important vector". The weights of the vectors in this ordered basis are respectively $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$. There is a nice geometric interpretation for what we mean by "important" described in [172] that we omit here. The takeaway is that we have an ordered basis for colspace($\mathbf{U}$), or rather, an ordered basis for the $x-$dimension.

A natural question arises – is there an ordered basis for the $y-$dimension? In fact, there is an orthonormal basis $\{\mathbf{v}_{y,1}, \mathbf{v}_{y,2}, ..., \mathbf{v}_{y,r}\}$ in y that corresponds to our ordered basis in x. It turns out the vectors $\{\mathbf{v}_{x,1}, \mathbf{v}_{x,2}, ..., \mathbf{v}_{x,r}\}$ are the eigenvectors of $\mathbf{U}\mathbf{U}^T$, and the weights $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$ are the square roots of the nonzero eigenvalues of $\mathbf{U}\mathbf{U}^T$ [172]. Moreover, the vectors $\{\mathbf{v}_{y,1}, \mathbf{v}_{y,2}, ..., \mathbf{v}_{y,r}\}$ are the eigenvectors of $\mathbf{U}^T\mathbf{U}$, and the weights $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$ are the square roots of the nonzero

eigenvalues of $\mathbf{U}^T\mathbf{U}$ [172]. In this sense, the orthonormal bases are connected by the weights σ_k, $k = 1, 2, ..., r$. All together, we have that

$$\mathbf{U} = \sigma_1 \mathbf{v}_{x,1}\mathbf{v}_{y,1}^T + \sigma_2 \mathbf{v}_{x,2}\mathbf{v}_{y,2}^T + ... + \sigma_r \mathbf{v}_{x,r}\mathbf{v}_{y,r}^T.$$

The is known as the **reduced singular value decomposition** of matrix $\mathbf{U}$. Letting $\mathbf{V}_x \in \mathbb{R}^{N_x \times r}$ be the matrix with columns $\mathbf{v}_{x,k}$ (known as the **left singular vectors**), $\mathbf{V}_y \in \mathbb{R}^{N_y \times r}$ be the matrix with columns $\mathbf{v}_{y,k}$ (known as the **right singular vectors**), and $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$ be the diagonal matrix with diagonal entries σ_k for $k = 1, 2, ..., r$ (known as the **singular values**), the reduced SVD can be expressed more compactly as

$$\mathbf{U} = \mathbf{V}_x \mathbf{\Sigma} \mathbf{V}_y^T = \sum_{k=1}^{r} \sigma_k \mathbf{v}_{x,k}\mathbf{v}_{y,k}^T. \tag{3.1.1}$$

Remark 3.1. We can alternatively derive the reduced SVD of a matrix from a linear mappings viewpoint. As in the geometric interpretation described in [172], we could consider an orthogonal (not unit vectors) basis $\{\mathbf{v}_{x,1}, \mathbf{v}_{x,2}, ..., \mathbf{v}_{x,r}\}$ with lengths $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$. Let $\{\mathbf{v}_{y,1}, \mathbf{v}_{y,2}, ..., \mathbf{v}_{y,r}\}$ be the orthonormal basis that $\mathbf{U}$ maps to $\{\mathbf{v}_{x,1}, \mathbf{v}_{x,2}, ..., \mathbf{v}_{x,r}\}$. Normalizing and redefining the basis vectors $\mathbf{v}_{x,k} \coloneqq \mathbf{v}_{x,k}/\sigma_k$, we have the reduced SVD (3.1.1).

Remark 3.2 (Full singular value decomposition)**.** In the case that $r < N_x$ or $r < N_y$, we might also desire orthonormal bases for all of $\mathbb{R}^{N_x}$ and $\mathbb{R}^{N_y}$. However, we do not want to affect the reduced SVD that has been constructed. We can extend the bases from the reduced SVD to form our desired orthonormal bases, and then append zeros to $\mathbf{\Sigma}$. Define the following matrices:

$$\tilde{\mathbf{V}}_x = \begin{bmatrix} \mathbf{V}_x & \mathbf{V}'_x \end{bmatrix}_{N_x \times N_x}, \quad \tilde{\mathbf{\Sigma}} = \begin{bmatrix} \mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{N_x \times N_y}, \quad \tilde{\mathbf{V}}_y = \begin{bmatrix} \mathbf{V}_y & \mathbf{V}'_y \end{bmatrix}_{N_y \times N_y},$$

where $\mathbf{V}'_x$ and $\mathbf{V}'_y$ are the appended bases. The **full singular value decomposition**

(SVD) of a matrix $\mathbf{U} \in \mathbb{R}^{N_x \times N_y}$ is

$$\mathbf{U} = \tilde{\mathbf{V}}_x \tilde{\mathbf{\Sigma}} \tilde{\mathbf{V}}_y^T. \tag{3.1.2}$$

There is a very important interpretation of the SVD – data compression of a matrix $\mathbf{U}$. What if $\mathbf{U}$ is a very large matrix that is naturally low-rank in the sense that the first $r' \ll \text{rank}(\mathbf{U})$ singular values are more dominant? The degree of dominance could be measured by some small tolerance ϵ (e.g., $\epsilon = \mathcal{O}(1.0E-10)$). For example, we can use a rank r' matrix to approximate the full matrix $\mathbf{U}$ with an error specified below in Theorem 3.1. The full-rank matrix $\mathbf{U}$ has a storage complexity of $N_x N_y$, but a rank-r' approximation has a storage complexity of $r'(N_x + N_y + 1) \ll N_x N_y$ to store the first r' singular values/vectors. Only keeping the first r' rank-1 matrices in equation (3.1.1),

$$\mathbf{U} \approx \mathbf{U}_{r'} \coloneqq \sum_{k=1}^{r'} \sigma_k \mathbf{v}_{x,k} \mathbf{v}_{y,k}^T. \tag{3.1.3}$$

Theorem 3.1 (Low-rank approximations from the SVD [61, 172])**.** Let $\mathbf{U} \in \mathbb{R}^{N_x \times N_y}$ with $r = \text{rank}(\mathbf{U})$. Further let $1 \leq r' \leq r$ and define $\mathbf{U}_{r'}$ as in equation (3.1.3). The best rank-r' approximation of $\mathbf{U}$ is given by the leading r' factors of the SVD, that is,

$$\|\mathbf{U} - \mathbf{U}_{r'}\| = \inf_{\substack{\mathbf{W} \in \mathbb{R}^{N_x \times N_y} \\ \text{rank}(\mathbf{U}_{r'}) \leq r'}} \|\mathbf{U} - \mathbf{W}\|.$$

In particular,

$$\|\mathbf{U} - \mathbf{U}_{r'}\|_2 = \begin{cases} \sigma_{r'+1}, & 1 \leq r' < r, \\ 0, & r' = r, \end{cases}$$

$$\|\mathbf{U} - \mathbf{U}_{r'}\|_F = \begin{cases} \sqrt{\sigma_{k'+1}^2 + \ldots + \sigma_r^2}, & 1 \leq r' < r, \\ 0, & r' = r, \end{cases}$$

where $\|\cdot\|_2$ denotes the L^2 norm, and $\|\cdot\|_F$ denotes the Frobenius norm.

Takeaway 3.1. The reduced SVD can be used to approximate $\mathbf{U}$ by an ordered sum of r' rank-1 matrices. Moreover, this is the best rank-r' approximation. This leads to significant reduction in storage complexity when the formal rank of $\mathbf{U}$ is large.

Takeaway 3.2. The reduced SVD decomposes $\mathbf{U}$ into orthonormal bases for each dimension ordered from "most important" to "least important."

There is a vast literature on computing the (reduced) SVD of a matrix. Certain algorithms are better suited for matrices with structure (e.g., tridiagonal matrices), special matrices (e.g., Hankel matrices), matrices where $N_x \gg N_y$, and so on. It has been common practice over the past several decades to compute the SVD of a matrix using a two-phase process [42, 172]. **Phase 1** transforms the matrix into a bidiagonal form; and **Phase 2** diagonalizes the bidiagonal matrix from Phase 1. Both phases together produce the singular values (from the diagonalization) and the singular vectors (if desired) in an efficient and stable manner.

The computational complexity of Phase 1 is typically larger than that of Phase 2 since the latter works with a bidiagonal form of size $N_y \times N_y$ [42, 172]. Phase 2 requires $\mathcal{O}(N_y^2)$ flops if only the singular values are required [42, 172], although the more recent divide and conquer methods also compute the singular vectors in $\mathcal{O}(N_y^2)$ flops.

Given that Phase 1 usually dominates the computational complexity, we list a few algorithms for Phase 1 presented in [42, 172]. We refer the reader to [42] for a collection of algorithms for Phase 2. For the sake of this dissertation, the fine details of each algorithm are not necessary and can be found in [42, 76, 172]. The simulations contained within this dissertation use MATLAB's svd function to compute the (reduced) SVD of a matrix. MATLAB's svd function uses LAPACK to compute the SVD, and the routines used by LAPACK can be found in [5]. Although the algorithms presented here are not necessarily what MATLAB uses in our simulations, they still provide a good sense of the computational complexities that we expect to observe.

Algorithms for Phase 1 [172]:

- One-step (Golub-Kahan) bidiagonalization, $\sim 4N_xN_y^2 - \frac{4}{3}N_y^3$ flops.
- Two-step (Lawson-Hanson-Chan) bidiagonalization, $\sim 2N_xN_y^2 + 2N_y^3$ flops.

  Better than Golub-Kahan bidiagonalization if $N_x > \frac{5}{3}N_y$.
- Three-step bidiagonalization, $\sim 4N_xN_y^2 - \frac{4}{3}N_y^3 - \frac{2}{3}(N_x - N_y)^3$ flops.

  Provides a smooth transition from one-step and two-step bidiagonalizations for $N_y < N_x < 2N_y$ *but the improvement achieved is small.*

As we will see in Section 3.1.2.2, the low-rank solutions involved in this chapter will require computing the SVD of size $N_x \times r$ (or size $r \times r$) matrices. In this case, we expect the computational complexity to be roughly $\mathcal{O}(N_x r^2)$ flops, where $r \ll N_x$. Although not discussed here, we note that randomization methods have gained recent popularity as an effective way to compute a near-best low-rank approximation of a matrix [25, 90].

3.1.1.2 QR factorization

As with Section 3.1.1.1, the material from this subsection is predominantly taken from [76, 172]. Consider a matrix $\mathbf{U} \in \mathbb{R}^{N_x \times N_y}$ with $N_x \geq N_y$, and let its column vectors be denoted $\mathbf{u}_j$, $j = 1, 2, ..., N_y$. The **reduced QR factorization** of a matrix $\mathbf{U} \in \mathbb{R}^{N_x \times N_y}$ is

$$\mathbf{U} = \mathbf{QR}, \tag{3.1.4}$$

where the columns of $\mathbf{Q} \in \mathbb{R}^{N_x \times N_y}$ are orthonormal, and $\mathbf{u}_j \in \text{span}(\mathbf{q}_1, ..., \mathbf{q}_j)$ for $j = 1, 2, ..., N_y$. The matrix $\mathbf{R} \in \mathbb{R}^{N_y \times N_y}$ is upper-triangular so that

$$\mathbf{u}_j = r_{1j}\mathbf{q}_1 + r_{2j}\mathbf{q}_2 + ... + r_{jj}\mathbf{q}_j, \qquad j = 1, 2, ..., N_y.$$

Remark 3.3. If $\mathbf{U}$ is full-rank, then $\text{span}(\mathbf{u}_1, ..., \mathbf{u}_j) = \text{span}(\mathbf{q}_1, ..., \mathbf{q}_j)$ for $j = 1, 2, ..., N_y$.

Remark 3.4. If $\mathbf{U}$ is rank-deficient, then at least one of the diagonal entries of $\mathbf{R}$ will be zero. For example, assume $\mathbf{u}_1$ and $\mathbf{u}_2$ are linearly independent, but that $\mathbf{u}_3 \in \text{span}(\mathbf{u}_1, \mathbf{u}_2) = \text{span}(\mathbf{q}_1, \mathbf{q}_2)$. There is no contribution from $\mathbf{q}_3$ and so $r_{33} = 0$.

Remark 3.5. Unlike the SVD, the QR factorization does not order the vectors in a "most important" and "least important" fashion. However, the upper-triangular matrix $\mathbf{R}$ does imply the linear (in)dependence of the column vectors of $\mathbf{U}$ and hence the column vectors of $\mathbf{Q}$ that form an orthonormal basis for colspace($\mathbf{U}$).

In a spirit similar to the full SVD of a matrix in equation (3.1.2), we can extend the orthonormal basis $\{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_{N_y}\}$ to an orthonormal basis for all of $\mathbb{R}^{N_x}$. Define the following matrices:

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{Q}' \end{bmatrix}_{N_x \times N_x}, \qquad \tilde{\mathbf{R}} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}_{N_x \times N_y},$$

where $\mathbf{Q}' \in \mathbb{R}^{N_x \times (N_x - N_y)}$ holds the appended basis vectors. The **full QR factorization** of a matrix $\mathbf{U} \in \mathbb{R}^{N_x \times N_y}$ is

$$\mathbf{U} = \tilde{\mathbf{Q}}\tilde{\mathbf{R}}, \tag{3.1.5}$$

where $\tilde{\mathbf{Q}}$ is a unitary matrix.

As with Section 3.1.1.1, we suffice to only list two popular algorithms used to compute the QR factorization presented in [172]. Although more efficient algorithms might be used in practice, these provide a good sense of the computational complexities that we expect to observe. The simulations contained within this dissertation use MATLAB's qr function to compute the (reduced) QR factorization of a matrix. MATLAB's qr function uses LAPACK to compute the QR factorization, and the routines used by LAPACK can be found in [5].

Algorithms for computing the QR factorization [172]:

- Modified Gram-Schmidt, $\sim 2N_x N_y^2$ flops.

- Householder triangularization, $\sim 2N_xN_y^2 - \frac{2}{3}N_y^3$ flops.

As we will see in Section 3.1.2.2, the low-rank solutions involved in this chapter will require computing the QR factorization of size $N_x \times r$ matrices. In which case, we expect the computational complexity to be roughly $\mathcal{O}(N_x r^2)$ flops, where $r \ll N_x$.

3.1.1.3 CP decomposition

We have seen one instance of how a matrix can be expressed as a finite sum of outer products of vectors, e.g., the SVD in equation (3.1.1). Componentwise, equation (3.1.1) can be expressed as

$$U(i,j) = \sum_{k=1}^{r} \sigma_k v_{x,k}(i) v_{y,k}(j),$$

where $U(i,j)$ denotes the (i,j) entry of matrix $\mathbf{U}$. For reasons that will become clear in a moment, we now denote entries of an array with tuples of the indices rather than with subscripts.

A matrix is simply a two-index array, or rather, an order-2 tensor. Just like a matrix can be expressed as, or approximated by, a finite sum of outer products of vectors, so too can order-d tensors [106, 109, 112].

Definition 3.1 (Rank-1 tensor)**.** Consider an order-d tensor $\boldsymbol{\mathscr{X}} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ with multi-index $i = (i_1, i_2, ..., i_d)$ for which $i_k = 1, 2, ..., N_k$ for $k = 1, 2, ..., d$. We call $\boldsymbol{\mathscr{X}}$ a **rank-1 tensor** if for some vectors $\mathbf{a}^{(k)} \in \mathbb{R}^{N_k}$, $k = 1, 2, ..., d$,

$$\boldsymbol{\mathscr{X}} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \ldots \circ \mathbf{a}^{(d)}, \tag{3.1.6}$$

where $\circ$ denotes the outer product. The componentwise form of equation (3.1.6) is

$$\boldsymbol{\mathscr{X}}(i_1, i_2, ..., i_d) = a^{(1)}(i_1) a^{(2)}(i_2) \ldots a^{(d)}(i_d).$$

Definition 3.2 (Tensor rank)**.** The **rank** of a tensor $\boldsymbol{\mathscr{X}}$, denoted rank($\boldsymbol{\mathscr{X}}$), is the smallest number of rank-1 tensors whose sum generates $\boldsymbol{\mathscr{X}}$.

Consider an order-d tensor $\boldsymbol{\mathscr{X}} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ with multi-index $i = (i_1, i_2, ..., i_d)$ for which $i_k = 1, 2, ..., N_k$ for $k = 1, 2, ..., d$. We want to express the tensor $\boldsymbol{\mathscr{X}}$ as

$$\boldsymbol{\mathscr{X}} \approx \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \ldots \circ \mathbf{a}_r^{(d)}, \tag{3.1.7}$$

where $\circ$ denotes the outer product, and for each $k = 1, 2, ..., d$ we have that $\mathbf{a}_r^{(k)} \in \mathbb{R}^{N_k}$ for $r = 1, 2, ..., R$. Equation (3.1.7) is known as the **CANDECOMP/PARAFAC (CP) decomposition**; there are many names for this tensor decomposition summarized in [109]. The CP decomposition factorizes a tensor into a sum of rank-1 tensors. Given the vectors in equation (3.1.7), we can recover the tensor $\boldsymbol{\mathscr{X}}$ to some tolerance. Another way to write equation (3.1.7) is to normalize the vectors in the following way,

$$\boldsymbol{\mathscr{X}} \approx \sum_{r=1}^{R} \lambda_r \frac{\mathbf{a}_r^{(1)}}{\left\|\mathbf{a}_r^{(1)}\right\|} \circ \frac{\mathbf{a}_r^{(2)}}{\left\|\mathbf{a}_r^{(2)}\right\|} \circ \ldots \circ \frac{\mathbf{a}_r^{(d)}}{\left\|\mathbf{a}_r^{(d)}\right\|}, \tag{3.1.8}$$

where $\lambda_r = \left\|\mathbf{a}_r^{(1)}\right\| \left\|\mathbf{a}_r^{(2)}\right\| \ldots \left\|\mathbf{a}_r^{(d)}\right\|$. Componentwise, equations (3.1.7) and (3.1.8) can be respectively expressed as

$$\boldsymbol{\mathscr{X}}(i_1, i_2, ..., i_d) \approx \sum_{r=1}^{R} a_r^{(1)}(i_1) a_r^{(2)}(i_2) \ldots a_r^{(d)}(i_d).$$

$$\boldsymbol{\mathscr{X}}(i_1, i_2, ..., i_d) \approx \sum_{r=1}^{R} \lambda_r \frac{a_r^{(1)}(i_1)}{\left\|\mathbf{a}_r^{(1)}\right\|} \frac{a_r^{(2)}(i_2)}{\left\|\mathbf{a}_r^{(2)}\right\|} \ldots \frac{a_r^{(d)}(i_d)}{\left\|\mathbf{a}_r^{(d)}\right\|}.$$

It is important to note that with respect to each dimension, the vectors do not need to be orthogonal. That is, for each $k = 1, 2, ..., d$, the vectors $\{\mathbf{a}_1^{(k)}, \mathbf{a}_2^{(k)}, ..., \mathbf{a}_R^{(k)}\}$ do not need to be orthogonal, although, in certain applications it is useful to have them be orthogonal.

As mentioned earlier, the main attraction of tensor decompositions is their ability to significantly reduce storage requirements and computational complexities. A full-blown tensor $\mathscr{X}$ has $\mathcal{O}(N^d)$ degrees of freedom and storing every entry is clearly unreasonable for larger d. Yet, the CP decomposition (3.1.7) only requires storing R vectors of size $N \times 1$ for each dimension. So, the storage complexity of the CP decomposition is $\mathcal{O}(dRN)$. This is much less than N^d, especially for higher-order tensors.

Takeaway 3.3. The CP decomposition (3.1.7) reduces the storage complexity from $\mathcal{O}(N^d)$ to $\mathcal{O}(dRN)$.

There is a vast literature on the CP decomposition, and we refer the reader to the excellent paper by Kolda and Bader [109] for a rich source of references. For the sake of this dissertation, here are a few important remarks.

Remark 3.6. The CP decomposition is not unique.

Remark 3.7. The **rank** of a tensor can also be defined as the smallest number of rank-1 tensors for which we attain an exact CP decomposition, where "exact" indicates equality in equations (3.1.7) and (3.1.8).

Remark 3.8. The best rank-k approximation of an order-2 tensor is given by the leading k factors of the SVD [61]. The same analogy is not true for tensors of order three or higher. In fact, a tensor may end up being approximated arbitrarily well by a low-rank CP decomposition. We suffice to state that in such cases, a different notion of rank, known as **border rank**, is used. We omit further discussion and refer the reader to [106, 109, 112] for more details.

The CP decomposition is attractive because of its reduction in storage complexity, as well as its straightforward construction. Moreover, it remains a popular tool in the data science and machine learning communities due to its flexibility and interpretability [3, 137, 146]. It is also a popular tools in psychometrics and biological applications, and several references are provided in [109]. Yet, the rank degeneracy

mentioned in Remark 3.8 poses a slight complication for large-scale simulations, and other tensor decompositions with similar storage complexities are sometimes preferred in higher-dimensional problems (e.g., Tucker [174], tensor train [111, 135], hierarchical Tucker [85, 112]).

3.1.2 Low-rank tensor approaches for time-dependent PDEs

Several low-rank tensor methods have been developed for high-dimensional time-dependent PDEs, particularly for kinetic models [45, 52, 62, 64, 65, 85, 105, 111]. Such methods have been designed to take advantage of the low-rank structure inherent in applicable PDEs. Some methods increase the rank of the numerical solution after each time-step and thus use a truncation procedure to remove redundant basis vectors [85, 86]. Other methods assume a low-rank basis for the solution that evolves dynamically [64, 65]. In this section, we review step-and-truncate methods and the dynamical low-rank (DLR) framework.

3.1.2.1 Step-and-truncate methods

As we evolve the solutions to time-dependent PDEs, we want to ensure low-rank structures are preserved. Step-and-truncate procedures are one way of maintaining low-rank structure [86, 149]. Simply put, these methods truncate the solution after each time-step, or rather, project the updated solution onto a lower-dimensional subspace. Two recent algorithms are the step-truncation algorithms presented in [149], and the basis removal procedures presented in [84, 85, 86]. The results in this dissertation use the basis removal procedure from [86], discussed below.

An illustrative example

As a motivating example, say $\mathbf{U}^{n+1} = \mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T$ is the solution to the two-dimensional heat equation. Updating the solution (e.g., by forward Euler

method) can increase the rank of the numerical solution going from time t^n to t^{n+1}. Assuming $\mathbf{U}^n$ has (low-)rank r^n, one time-step of forward Euler yields

$$\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T = \mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T + \Delta t\left(\mathbf{D}^x\mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T + \mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{D}^y\mathbf{V}^{y,n})^T\right), \tag{3.1.9}$$

where $\mathbf{D}^x$ and $\mathbf{D}^y$ approximate the second-partial derivatives. The updated solution has grown from rank r^n to rank $3r^n$.

$$\mathbf{U}^{n+1} = \begin{bmatrix}\mathbf{V}^{x,n} & \mathbf{D}^x\mathbf{V}^{x,n} & \mathbf{V}^{x,n}\end{bmatrix}_{N_x\times 3r^n}\begin{bmatrix}\mathbf{S}^n & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \Delta t\mathbf{S}^n & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \Delta t\mathbf{S}^n\end{bmatrix}_{3r^n\times 3r^n}\begin{bmatrix}\mathbf{V}^{y,n} & \mathbf{V}^{y,n} & \mathbf{D}^y\mathbf{V}^{y,n}\end{bmatrix}^T_{N_y\times 3r^n}. \tag{3.1.10}$$

Even if the solution has low-rank structure, the rank will continue to grow unless a step-and-truncate method is performed, e.g., basis removal procedure. As discussed in Section 3.1.1.1, the truncated SVD provides the optimal low-rank approximation. Following the basis removal procedure in [86], let $\mathbf{V}^{x,n+1}$, $\mathbf{S}^{n+1}$ and $\mathbf{V}^{y,n+1}$ be the augmented matrices in equation (3.1.10). Using the reduced QR factorization and SVD,

$$\underbrace{\mathbf{V}^{x,n+1}}_{QR}\mathbf{S}^{n+1}(\underbrace{\mathbf{V}^{y,n+1}}_{QR})^T = \mathbf{Q}_x\underbrace{\mathbf{R}_x\mathbf{S}^{n+1}\mathbf{R}_y^T}_{SVD}\mathbf{Q}_y^T = (\mathbf{Q}_x\mathbf{U})\mathbf{\Sigma}(\mathbf{Q}_y\mathbf{V})^T.$$

Only keeping the r^{n+1} singular values larger than some tolerance $\epsilon > 0$, redefine the updated low-rank solution

$$\mathbf{V}^{x,n+1} \coloneqq \mathbf{Q}_x\mathbf{U}_{:,1:r^{n+1}}, \quad \mathbf{S}^{n+1} \coloneqq \mathbf{\Sigma}_{1:r^{n+1},1:r^{n+1}}, \quad \mathbf{V}^{y,n+1} \coloneqq \mathbf{Q}_y\mathbf{V}_{:,1:r^{n+1}}.$$

Note that the columns of $\mathbf{Q}_x$ and $\mathbf{U}$ being orthonormal implies that the columns of $\mathbf{Q}_x\mathbf{U}$ are orthonormal; and similarly for the columns of $\mathbf{Q}_y\mathbf{V}$.

Referring back to the computational complexities for the SVD and QR factorization mentioned in Sections 3.1.1.1 and 3.1.1.2, notice that the flop count is dramatically reduced for low-rank solutions. For notational simplicity we use N for spatial mesh

in each dimension, and r for the rank. According to the computational complexities of the algorithms stated in Sections 3.1.1.1 and 3.1.1.2: the computational complexity to compute the QR decompositions of $\mathbf{V}^{x,n+1}\mathbf{V}^{y,n+1}$ is $\mathcal{O}(Nr^2)$ flops, and the computational complexity to compute the SVD of $\mathbf{R}_x\mathbf{S}^{n+1}\mathbf{R}_y^T \in \mathbb{R}^{r\times r}$ is $\mathcal{O}(r^3)$ flops. The computational cost of the matrix multiplications for the redefined updated bases is $\mathcal{O}(Nr^2)$. If $r \ll N$, we avoid the curse of dimensionality since the overall computational complexity is dominated by $\mathcal{O}(Nr^2)$ and grows linearly in N; the full-rank solution would have $\mathcal{O}(N^3)$ flops for the SVD, QR factorizations and matrix multiplications.

A simpler case

In the previous example, the matrices $\mathbf{V}^{x,n+1}$ and $\mathbf{V}^{y,n+1}$ were not assumed to be orthonormal, hence why the QR factorization was needed. In the case where these matrices are orthonormal, the QR factorization is not needed. Say we are simply given the updated solution $\mathbf{U}^{n+1} = \mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T$ but that it is not yet truncated. We want to compute a low-rank approximation by removing unnecessary basis vectors. As discussed in Section 3.1.1.1, the truncated SVD provides the optimal low-rank approximation. Computing the SVD of $\mathbf{S}^{n+1}$,

$$\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T = \mathbf{V}^{x,n+1}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{V}^{y,n+1})^T. \tag{3.1.11}$$

Only keeping the r^{n+1} singular values larger than some tolerance $\epsilon > 0$, redefine the updated low-rank solution

$$\mathbf{V}^{x,n+1} \coloneqq \mathbf{V}^{x,n+1}\mathbf{U}_{:,1:r^{n+1}}, \quad \mathbf{S}^{n+1} \coloneqq \mathbf{\Sigma}_{1:r^{n+1},1:r^{n+1}}, \quad \mathbf{V}^{y,n+1} \coloneqq \mathbf{V}^{y,n+1}\mathbf{V}_{:,1:r^{n+1}}.$$

3.1.2.2 Dynamical low-rank (DLR) methods

Dynamical low-rank (DLR) methods are quickly becoming a popular way to evolve high-dimensional time-dependent PDEs [32, 45, 63, 65, 125]. Since this dissertation is only concerned with order-2 tensors, we just consider the DLR framework for the two-dimensional case. However, these methods naturally extend to high-order tensors and related tensor decompositions [128].

Broadly speaking, the DLR framework factors the time-dependent tensor into a tensor product of basis functions from each dimension, together with a transfer matrix that stores the coefficients. Take for example the singular value decomposition (3.1.1). The singular vectors (i.e., orthonormal basis vectors) can be thought of as time dependent basis functions per dimension, and the singular values can be thought of as time dependent coefficients. Assuming the solution/tensor has low-rank structure, the bases can be evolved in a dynamical way. In this subsection, we review the three equations that form the foundation of the DLR framework. There are various robust integrators developed for the DLR framework for different individual needs, and we leave their discussion to the literature.

We present the DLR formulation applied to equation (3.0.2), although the DLR formulation applied to other equations follows similarly. The DLR scheme is built upon an assumption on the low-rank approximation for the solution,

$$u(x,y,t) = \sum_{i=1}^{r}\sum_{j=1}^{r} V_i^x(x,t) S_{ij}(t) V_j^y(y,t), \tag{3.1.12}$$

where $\{V_i^x(x,t) : i = 1,2,...,r\}$ and $\{V_j^y(y,t) : i = 1,2,...,r\}$ are orthonormal time-dependent bases in the $x-$ and $y-$dimensions, respectively. If we were to discretize equation (3.1.12) in space, the matrix analogue might look like the SVD. They are not the same since $\mathbf{S} \in \mathbb{R}^{r\times r}$ need not be diagonal, although it is invertible.

Continuing with the continuous low-rank approximation (3.1.12), define the

auxiliary bases

$$K_J(x,t) = \sum_{i=1}^{r} V_i^x(x,t) S_{iJ}(t), \qquad J = 1, 2, ..., r, \tag{3.1.13a}$$

$$L_I(y,t) = \sum_{j=1}^{r} S_{Ij}(t) V_j^y(y,t), \qquad I = 1, 2, ..., r. \tag{3.1.13b}$$

Projecting equation (3.0.2), the DLR approach solves the following three equations:

$$\frac{\partial K_J}{\partial t}(x,t) = \left\langle RHS, V_J^y \right\rangle_y, \tag{3.1.14a}$$

$$\frac{\partial L_I}{\partial t}(y,t) = \left\langle RHS, V_I^x \right\rangle_x, \tag{3.1.14b}$$

$$\frac{\partial S_{IJ}}{\partial t}(t) = \left\langle \left\langle RHS, V_I^x \right\rangle_x, V_J^y \right\rangle_y, \tag{3.1.14c}$$

where RHS is the righthand side of the equation $u_t = RHS$, and $\langle \cdot, \cdot \rangle_x$ and $\langle \cdot, \cdot \rangle_y$ denote the L^2 inner products in x and y, respectively. After some straightforward algebra, equations (3.1.14) simplify to the following three equations:

$$\frac{\partial K_J}{\partial t}(x,t) = d_1^2 \frac{\partial^2 K_J}{\partial x^2} + \sum_{j=1}^{r} c_{jJ}^y K_j, \tag{3.1.15a}$$

$$\frac{\partial L_I}{\partial t}(y,t) = \sum_{i=1}^{r} c_{iI}^x L_i + d_2^2 \frac{\partial^2 L_I}{\partial y^2}, \tag{3.1.15b}$$

$$\frac{\partial S_{IJ}}{\partial t}(t) = \sum_{i=1}^{r} c_{iI}^x S_{iJ} + \sum_{j=1}^{r} c_{jJ} S_{Ij}, \tag{3.1.15c}$$

where

$$c_{jJ}^y = \left\langle d_2^2 \frac{\partial^2 V_j^y}{\partial y^2}, V_J^y \right\rangle_y, \tag{3.1.16a}$$

$$c_{iI}^x = \left\langle d_1^2 \frac{\partial^2 V_i^x}{\partial x^2}, V_I^x \right\rangle_x. \tag{3.1.16b}$$

The order and way in which one solves these three equations (3.1.14) varies

depending on the DLR method. Traditionally, the K equation is solved, followed by the S equation, followed by the L equation [65, 125]. However, evolving K inherently evolves V^x and S, and evolving L inherently evolves V^y and S. Performing the K, S and L steps in this order evolves S over three time-steps. So, the S step is solved backwards in time so that S is evolved by a single time-step in total. This of course becomes complicated and raises concerns when dealing with diffusion. This led to the *unconventional integrator* that solves the K and L steps simultaneously, and then solves for S [33]. The K and L steps only keep the updated bases and toss the updated S, leaving the evolution of S to only come from the S step.

We leave further discussion of DLR methods to the literature [32, 33, 45, 63, 65, 105, 125]. One important note to make is that these DLR methods traditionally *project and then discretize*, that is, they project the continuous model and then discretize to solve for K, L and S. After discretizing and solving for the updated auxiliary bases $\mathbf{K}^{n+1} \in \mathbb{R}^{N_x \times r}$ and $\mathbf{L}^{n+1} \in \mathbb{R}^{N_y \times r}$, one can recover the updated bases $\mathbf{V}^x \in \mathbb{R}^{N_x \times r}$ and $\mathbf{V}^y \in \mathbb{R}^{N_y \times r}$ using the reduced QR factorization (3.1.4),

$$\mathbf{K}^{n+1} = \mathbf{QR} =: \mathbf{V}^{x,n+1}\mathbf{S}_K^{n+1},$$

$$\mathbf{L}^{n+1} = \mathbf{QR} =: \left(\mathbf{S}_L^{n+1}(\mathbf{V}^{x,n+1})^T\right)^T.$$

Although $\mathbf{S}_K^{n+1}$ and $\mathbf{S}_L^{n+1}$ are updated approximations of $\mathbf{S}(t = t^{n+1})$, they were evolved while only considering a projection in a single dimension. Hence, $\mathbf{S}^{n+1}$ should also be evolved using equation (3.1.15c) since the differential equation is projected in both directions. The updated solution is then

$$\mathbf{U}^{n+1} = \mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T.$$

3.1.3 von Neumann stability analysis

The von Neumann analysis is a tool for determining the stability of linear numerical methods for linear problems, e.g., equation (3.0.2). Our proposed method

constructs a low-rank scheme based on traditional implicit time integrators for solving the diffusion equation. As such, we review the von Neumann stability analysis of the time integrators of interest to assess their stability properties and to predict the behavior of their numerical solutions. For linear problems with periodic boundary conditions, assume

$$u^n(x) = \big(\alpha(k)\big)^n e^{ikx}, \qquad \text{for all } k, \tag{3.1.17}$$

where k is the wave number, n is the number of time-steps, and $\alpha(k)$ is the amplification factor. Here, we are only concerned with investigating the numerical stability of various time discretizations, so we consider the semi-discrete solution $u^n(x)$.

Theorem 3.2 (von Neumann Stability Analysis [119])**.** A linear numerical scheme is stable if and only if $|\alpha(k)| \leq 1$, for all k.

The implicit low-rank time integrator presented in this chapter is investigated under four different time discretizations: backward Euler, Crank-Nicolson, second-order backward differentiation formula, and second-order diagonally implicit Runge-Kutta. As the numerical tests will reveal, the amplification factor reveals a lot about the expected behavior of the error. Even if the von Neumann analysis indicates (un)conditional stability, negative values of α for large wave numbers $|k| \gg 0$ might result in *ringing behavior and oscillations* since the sign of solution (3.1.17) will alternate with each time-step. This ringing behavior is nonideal and could affect the observed error and convergence. So, we present the von Neumann analysis of those four time-stepping methods here. For the sake of simplifying the algebra, we shall assume in this section an isotropic (constant) diffusion tensor $\mathbf{D} = D\mathbf{I}$; in this case equation (3.0.1) is the heat equation.

3.1.3.1 Backward Euler (bE)

The backward Euler method for solving equation (3.0.1) is

$$\frac{u^{n+1} - u^n}{\Delta t} = \mathcal{L}(u^{n+1}, t^{n+1}; x), \tag{3.1.18}$$

where $\mathcal{L}(u,t;x) = D\nabla^2 u$. Plugging in the form (3.1.17),

$$\frac{\alpha^{n+1}e^{ikx} - \alpha^n e^{ikx}}{\Delta t} + k^2 D\alpha^{n+1}e^{ikx} = 0.$$

Solving for the amplification factor yields

$$\alpha(k) = \frac{1}{1 + k^2 D\Delta t}. \tag{3.1.19}$$

Clearly $|\alpha(k)| \leq 1$ for all k, and so method (3.1.18) is unconditionally stable. Under a convenient change of coordinates $\alpha(k) = A(\xi = \sqrt{k^2 D\Delta t})$, we plot the amplification factor below.

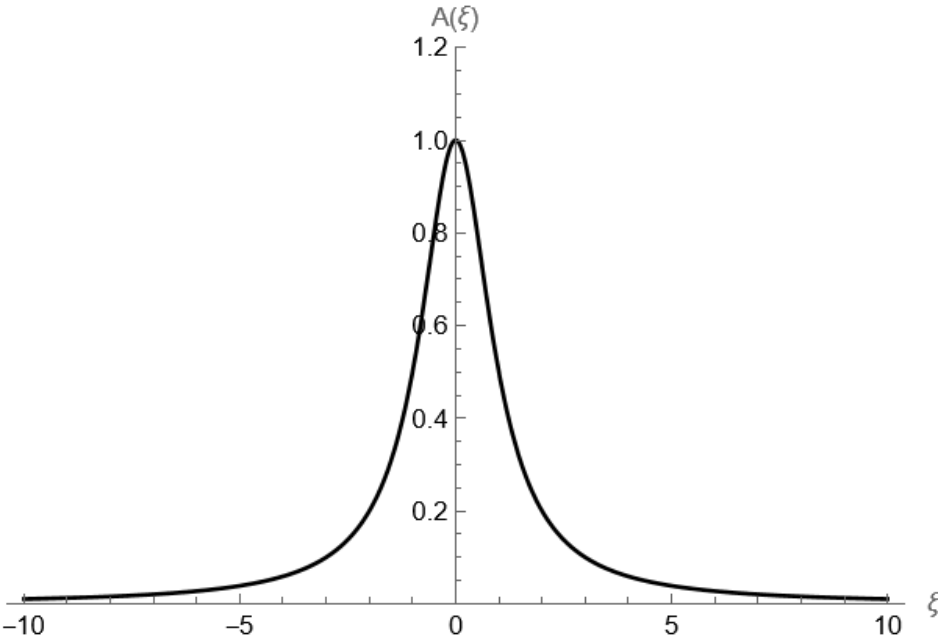


Figure 3.1: Plotting the amplification factor for backward Euler method applied to the heat equation.

Notice that the amplification factor is always positive and decays rapidly to zero. This positivity implies that backward Euler applied to equation (3.0.1) is immune to oscillations.

3.1.3.2 Crank-Nicolson (CN)

The Crank-Nicolson method for solving equation (3.0.1) is

$$\frac{u^{n+1}-u^n}{\Delta t} = \frac{1}{2}\mathcal{L}(u^{n+1}, t^{n+1}; x) + \frac{1}{2}\mathcal{L}(u^n, t^n; x), \tag{3.1.20}$$

where $\mathcal{L}(u,t;x) = D\nabla^2 u$. Plugging in the form (3.1.17),

$$\frac{\alpha^{n+1}e^{ikx} - \alpha^n e^{ikx}}{\Delta t} + \frac{k^2 D}{2}(\alpha^{n+1}e^{ikx} + \alpha^n e^{ikx}) = 0.$$

Solving for the amplification factor yields

$$\alpha(k) = \left(1 - \frac{k^2 D\Delta t}{2}\right) \Big/ \left(1 + \frac{k^2 D\Delta t}{2}\right). \tag{3.1.21}$$

Clearly $|\alpha(k)| \leq 1$ for all k, and so method (3.1.20) is unconditionally stable. Under a convenient change of coordinates $\alpha(k) = A(\xi = \sqrt{k^2 D\Delta t})$, we plot the amplification factor below.

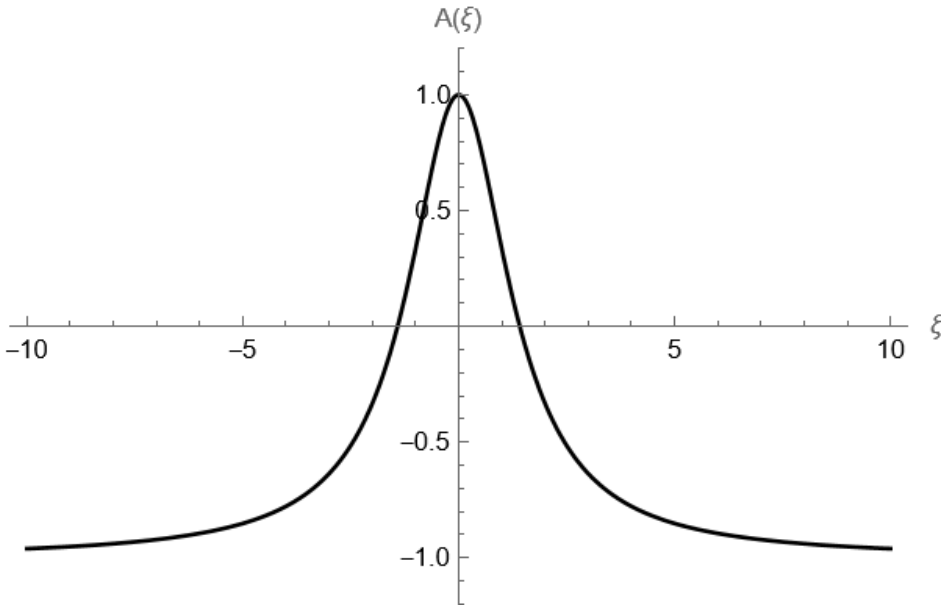


Figure 3.2: Plotting the amplification factor for Crank-Nicolson method applied to the heat equation.

Notice that the amplification factor is negative for $k > \sqrt{\frac{2}{D\Delta t}}$ (or $\xi > \sqrt{2}$), and $\alpha \to -1$ as $|k| \to \infty$. This negativity implies that Crank-Nicolson applied to equation (3.0.1) might suffer from oscillations if enough modes are excited. To investigate this further we consider two time scales: $\Delta t \sim D\Delta x^2/2$ and $\Delta t \gg D\Delta x^2/2$.

If $\Delta t \sim D\Delta x^2/2$, then $\alpha(k) = 0$ when $k = \sqrt[+]{\frac{2}{D\Delta t}} \sim \frac{2}{D\Delta x}$. In other words, when the time-stepping size is very small, the amplification factor will be positive for *several* wave numbers that roughly fall within $-\frac{2}{D\Delta x} < k < \frac{2}{D\Delta x}$. Assuming the solution does not excite more than $\frac{2}{D\Delta x}$ modes –which is a reasonable assumption– then Crank-Nicolson will not suffer from oscillations.

However, if $\Delta t \gg D\Delta x^2/2$, then $\alpha(k) = 0$ when $k = \sqrt[+]{\frac{2}{D\Delta t}} \ll \frac{2}{D\Delta x}$. In other words, when the time-stepping size is relatively large (e.g., $\mathcal{O}(\Delta x)$), the amplification factor will be negative for *several* wave numbers $\sqrt[+]{\frac{2}{D\Delta t}} < |k| < \frac{2}{D\Delta x}$. For solutions that excite modes in this region, Crank-Nicolson will suffer from ringing behavior and oscillations, hence affecting the observed error and convergence.

3.1.3.3 Backward differentiation formula (BDF2)

The second-order backward differentiation formula (BDF2) for solving equation (3.0.1) is

$$\frac{1}{\Delta t}\left(u^{n+2} - \frac{4}{3}u^{n+1} + \frac{1}{3}u^{n}\right) = \frac{2}{3}\mathcal{L}(u^{n+2}, t^{n+2}; x), \tag{3.1.22}$$

where $\mathcal{L}(u, t; x) = D\nabla^2 u$. Plugging in the form (3.1.17),

$$\frac{1}{\Delta t}\left(\alpha^{n+2}e^{ikx} - \frac{4}{3}\alpha^{n+1}e^{ikx} + \frac{1}{3}\alpha^{n}e^{ikx}\right) + \frac{2k^2D}{3}\alpha^{n+2}e^{ikx} = 0.$$

Solving for the amplification factor yields

$$\alpha_{\pm}(k) = \frac{2 \pm \sqrt{1 - 2k^2D\Delta t}}{3 + 2k^2D\Delta t}. \tag{3.1.23}$$

Even for the complex variable case, $|\alpha_{\pm}(k)| \leq 1$ for all k, and so method (3.1.22) is unconditionally stable. Under a convenient change of coordinates $\alpha_{\pm}(k) = A_{\pm}(\xi =$

$\sqrt{k^2 D\Delta t}$), we plot the amplification factors below. Since the real parts of the amplification factors are always positive and $|\alpha_{\pm}(k)| \leq 1$ for all k, we do not have to worry about negative values. Hence, we do not have to worry about ringing behavior and oscillations. But to better visualize the behavior of the amplification factors, we plot their absolute values.

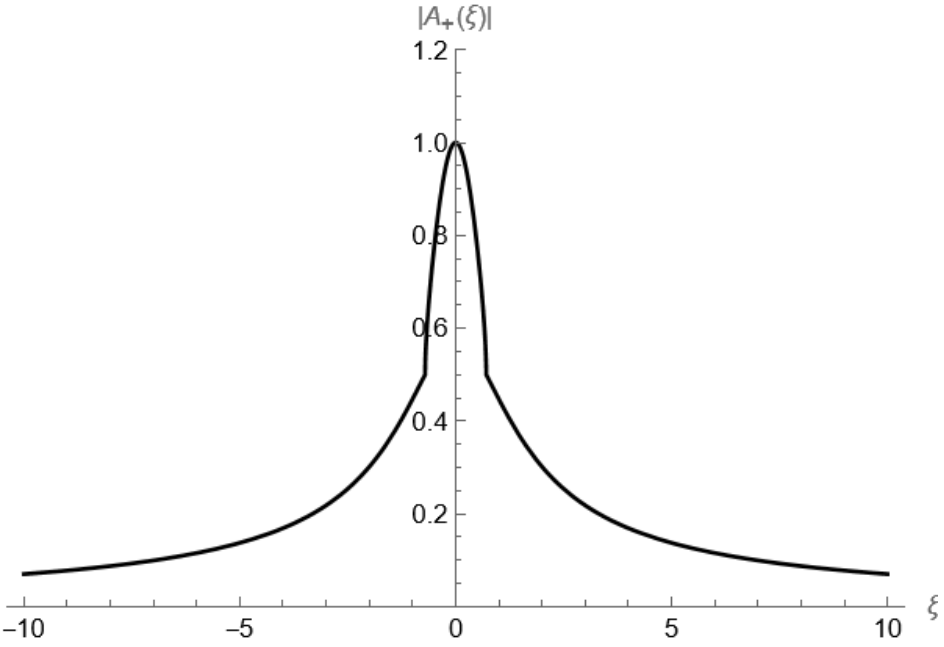


Figure 3.3: Plotting the absolute value of the amplification factor $A_+(\xi)$ for BDF2 applied to the heat equation.

Notice that the absolute value of the amplification factor decays rapidly to zero, and one can clearly see the same is true for the real part of the amplification factor. Although we plotted the absolute value of the amplification factors to account for the imaginary components, the real components of the amplification factors are positive. This positivity implies that BDF2 applied to equation (3.0.1) is immune to oscillations.

3.1.3.4 Diagonally implicit Runge-Kutta (DIRK2)

We consider two second-order diagonally implicit Runge-Kutta (DIRK) methods –one by Pareschi and Russo that is L-stable [138] and another that is stiffly-accurate [55]– whose Butcher tables are respectively given by

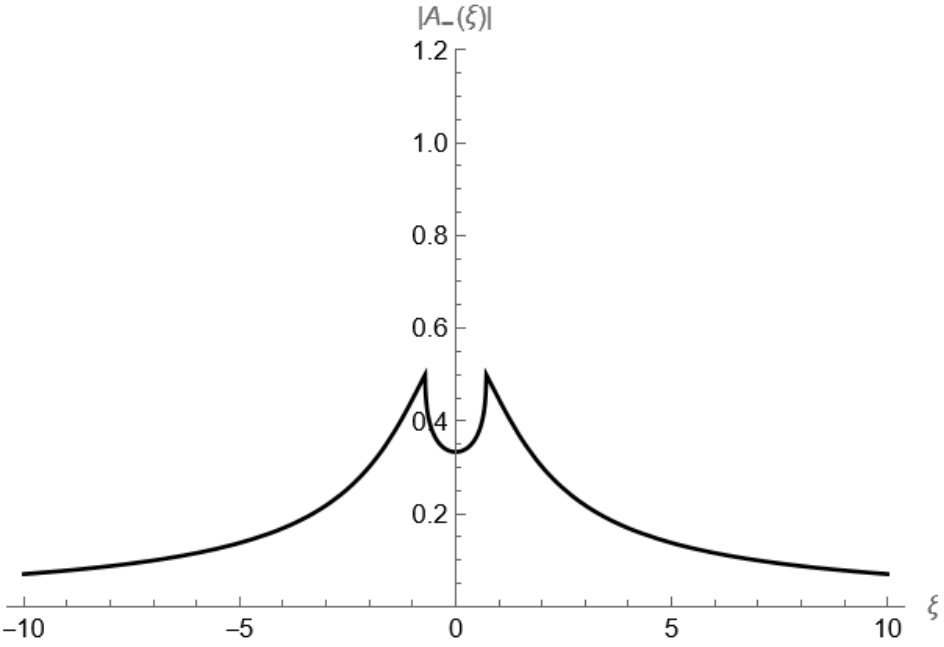


Figure 3.4: Plotting the absolute value of the amplification factor $A_{-}(\xi)$ for BDF2 applied to the heat equation.

L-stable DIRK2 [138]

Let $\gamma = 1 - 1/\sqrt{2}$

γ	γ	0
1-γ	$1-2\gamma$	γ
	$1/2$	$1/2$

Stiffly-accurate DIRK2 [55]

Let $\gamma = 1 - 1/\sqrt{2}$

γ	γ	0
1	$1-\gamma$	γ
	$1-\gamma$	γ

For a review of implicit Runge-Kutta methods see Section 2.1.2. It turns out that both DIRK2 methods applied to equation (3.0.1) have the same amplification factor, so we only present the amplification factor for the L-stable DIRK2 method. The

L-stable DIRK2 method for solving equation (3.0.1) is

$$u^{n+1} = u^n + \Delta t \left(\frac{1}{2} K_1 + \frac{1}{2} K_2 \right), \tag{3.1.24}$$

$$K_1 = \mathcal{L}(u^n + \Delta t(\gamma K_1), t^n + \gamma \Delta t; x), \tag{3.1.25}$$

$$K_2 = \mathcal{L}(u^n + \Delta t \left((1-2\gamma) K_1 + \gamma K_2 \right), t^n + (1-\gamma)\Delta t; x). \tag{3.1.26}$$

where $\mathcal{L}(u, t; x) = D\nabla^2 u$. We will need to solve for K_1 and K_2 implicitly, which in one-dimension requires two boundary conditions: $K(x = 0) = K(x = 2\pi)$ and $K'(x = 0) = K'(x = 2\pi)$, where we let K denote K_1 or K_2 depending on the stage. One can *easily* verify that under these boundary conditions, the solution to the ordinary differential equation

$$\left(D\Delta t\gamma \frac{d^2}{dx^2} - \mathbb{1} \right) K = -e^{ikx} \quad \Longleftrightarrow \quad K = \left(\mathbb{1} - D\Delta t\gamma \frac{d^2}{dx^2} \right)^{-1} e^{ikx} \tag{3.1.27}$$

is

$$K(x) = \frac{1}{1 + k^2 D\Delta t\gamma} e^{ikx}. \tag{3.1.28}$$

After a bit of tedious algebra, the equations for K_1 and K_2 are

$$K_1 = D \left(\mathbb{1} - D\Delta t\gamma \nabla^2 \right)^{-1} \nabla^2 u^n, \tag{3.1.29}$$

$$K_2 = K_1 + D\Delta t(1-2\gamma) \left(\mathbb{1} - D\Delta t\gamma \nabla^2 \right)^{-1} \nabla^2 K_1. \tag{3.1.30}$$

Plugging in the form (3.1.17) and by equation (3.1.28),

$$\begin{aligned} K_1 &= -\alpha^n k^2 D \left(\mathbb{1} - D\Delta t\gamma \nabla^2 \right)^{-1} e^{ikx} \\ &= \frac{-\alpha^n k^2 D}{1 + k^2 D\Delta t\gamma} e^{ikx}, \end{aligned} \tag{3.1.31a}$$

$$\begin{aligned} K_2 &= \frac{-\alpha^n k^2 D}{1 + k^2 D\Delta t\gamma} e^{ikx} + \frac{\alpha^n k^4 D^2 \Delta t(1-2\gamma)}{1 + k^2 D\Delta t\gamma} \left(\mathbb{1} - D\Delta t\gamma \nabla^2 \right)^{-1} e^{ikx} \\ &= \frac{-\alpha^n k^2 D}{1 + k^2 D\Delta t\gamma} e^{ikx} + \frac{\alpha^n k^4 D^2 \Delta t(1-2\gamma)}{(1 + k^2 D\Delta t\gamma)^2} e^{ikx} \end{aligned} \tag{3.1.31b}$$

Plugging in the form (3.1.17) and using equation (3.1.31),

$$\alpha^{n+1}e^{ikx} = \alpha^n e^{ikx} - \frac{\alpha^n k^2 D\Delta t}{1 + k^2 D\Delta t\gamma}e^{ikx} + \frac{\alpha^n (k^2 D\Delta t)^2(1-2\gamma)}{2(1 + k^2 D\Delta t\gamma)^2}e^{ikx}. \tag{3.1.32}$$

Solving for the amplification factor yields

$$\alpha(k) = 1 - \frac{k^2 D\Delta t}{1 + k^2 D\Delta t\gamma} + \frac{(k^2 D\Delta t)^2(1-2\gamma)}{2(1 + k^2 D\Delta t\gamma)^2}. \tag{3.1.33}$$

Although not obvious without plotting the solution, we find that $|\alpha(k)| \leq 1$ for all k, and so method (3.1.24) is unconditionally stable. Under a convenient change of coordinates $\alpha(k) = A(\xi = \sqrt{k^2 D\Delta t})$, we plot the amplification factor below.

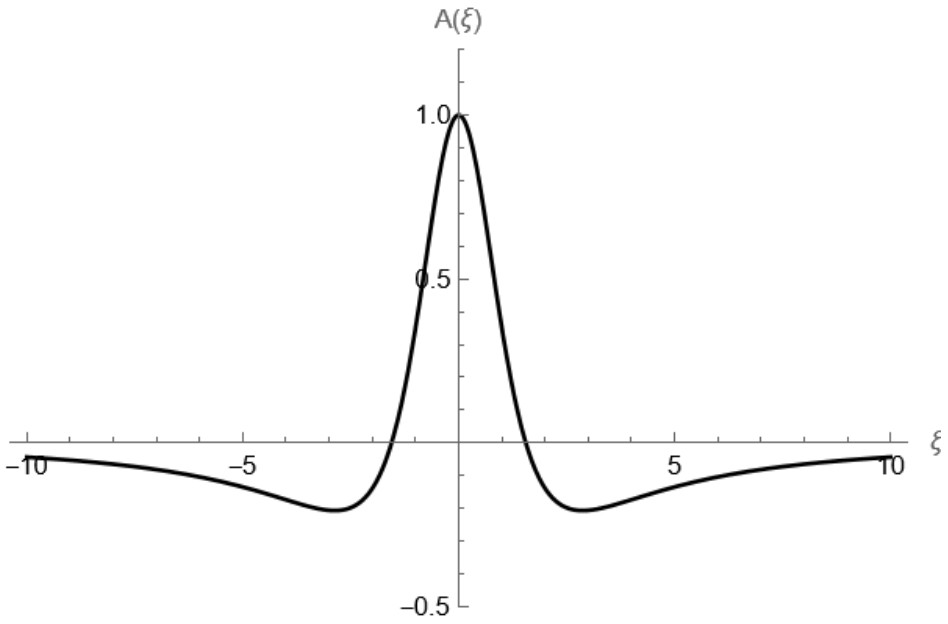


Figure 3.5: Plotting the amplification factor for DIRK2 applied to the heat equation.

Notice that the amplification factor has negative values, but the negative values decay to zero as $|k| \to \infty$. Moreover, the negative values are relatively small (> -2.2). As with Crank-Nicolson method, we consider two time scales: $\Delta t \sim D\Delta x^2/2$ and $\Delta t \gg D\Delta x^2/2$.

If $\Delta t \sim D\Delta x^2/2$, then $\alpha(k) = 0$ when $k = \sqrt[+]{\frac{1+\sqrt{2}}{D\Delta t}} \sim \frac{2(1+\sqrt{2})}{\Delta x}$. In other words,

when the time-stepping size is very small, the amplification factor will be positive for *several* wave numbers $-\frac{2(1+\sqrt{2})}{\Delta x} < k < \frac{2(1+\sqrt{2})}{\Delta x}$. Assuming the solution does not excite more than $\frac{2(1+\sqrt{2})}{\Delta x}$ modes –which is a reasonable assumption– then method (3.1.24) will not suffer from oscillations.

If $\Delta t \gg D\Delta x^2/2$, then $\alpha(k) = 0$ when $k = \sqrt[+]{\frac{1+\sqrt{2}}{D\Delta t}} \ll \frac{2(1+\sqrt{2})}{\Delta x}$. But according to Figure 3.5, we might expect the most negative values of α to fall within $|k| < \frac{2(1+\sqrt{2})}{\Delta x}$. Beyond which point, although the amplification factor will be negative for higher wave numbers, their contributions will be quickly decaying to zero. So, when the time-stepping size is relatively large (e.g., $\mathcal{O}(\Delta x)$), we can still expect method (3.1.24) to be immune to oscillations.

3.2 The implicit low-rank scheme

The proposed scheme solves the diffusion equation using traditional implicit time integrators by updating the bases in each dimension. We use several implicit time-discretizations suitable for stiff terms. We first present a first-order scheme with backward Euler, followed by a second-order scheme with stiffly-accurate second-order DIRK. Other second-order schemes using Crank-Nicolson and BDF2 are included in the appendices. At the continuous level, we assume the solution takes the same form as in the DLR framework (3.1.12),

$$u(x, y, t) = \sum_{i=1}^{r} \sum_{j=1}^{r} V_i^x(x, t) S_{ij}(t) V_j^y(y, t).$$

We remark that unlike DLR methods in which the equation is *projected and then discretized*, our proposed method *discretizes and then projects* the equation.

Discretizing the solution.

We discretize the spatial domain $[0, 1] \times [0, 1]$ using N_x and N_y evenly spaced

grid points in each dimension. The separated one-dimensional meshes are

$$0 = x_1 < x_2 < ... < x_{N_x-1} < x_{N_x} = 1,$$

$$0 = y_1 < y_2 < ... < y_{N_y-1} < y_{N_y} = 1,$$

where $\Delta x_i \coloneqq x_i - x_{i-1} = \Delta x$ for $i = 2, 3, ..., N_x$, and $\Delta y_j \coloneqq y_j - y_{j-1} = \Delta y$ for $j = 2, 3, ..., N_y$. At the semi-discrete level, the solution takes the form

$$\mathbf{U}(t) = \mathbf{V}^x(t)\mathbf{S}(t)(\mathbf{V}^y(t))^T, \tag{3.2.1}$$

where $\mathbf{U}(t) \in \mathbb{R}^{N_x \times N_y}$, the orthonormal columns of $\mathbf{V}^x(t) \in \mathbb{R}^{N_x \times r}$ are the basis vectors for the $x-$dimension, the orthonormal columns of $\mathbf{V}^y(t) \in \mathbb{R}^{N_y \times r}$ are the basis vectors for the $y-$dimension, and $\mathbf{S}(t) \in \mathbb{R}^{r \times r}$. This is not the same as the SVD since the entries along the diagonal of $S(t)$ are not necessarily ordered. The rank of the solution (3.2.1) is also a function of time, $r(t)$. Orthonormality of the column vectors is defined with the discrete unweighted ℓ^2 inner product (i.e., dot product),

$$\langle \mathbf{v}^x_{i1}, \mathbf{v}^x_{i2} \rangle \coloneqq \mathbf{v}^x_{i1} \cdot \mathbf{v}^x_{i2} = \delta_{i1,i2}.$$

The semi-discrete equation to solve is

$$\frac{d}{dt}\left(\mathbf{V}^x\mathbf{S}(\mathbf{V}^y)^T\right) = \mathbf{D}^x\mathbf{V}^x\mathbf{S}(\mathbf{V}^y)^T + \mathbf{V}^x\mathbf{S}(\mathbf{D}^y\mathbf{V}^y)^T, \tag{3.2.2}$$

where we use second-order centered differences to discretize the diffusive terms,

$$\mathbf{D}^x = \frac{d_1^2}{\Delta x^2}\text{tridiag}(1, -2, 1), \qquad \mathbf{D}^y = \frac{d_2^2}{\Delta y^2}\text{tridiag}(1, -2, 1).$$

The results herein assume homogeneous Dirichlet boundary conditions, so we only solve equation (3.2.2) on the interior nodes. Other boundary conditions would require a slight modification of $\mathbf{D}^x$ and $\mathbf{D}^y$, as well as the addition of a source term if

the boundary conditions are nonhomogeneous.

In the following sections, we discretize the time-interval $[0, T_f]$ using $N_t + 1$ evenly distributed time-steps,

$$0 = t^0 < t^1 < ... < t^{N_t - 1} < t^{N_t} = T_f,$$

where $\Delta t^n := t^n - t^{n-1} = \Delta t$ for $n = 1, 2, ..., N_t$.

3.2.1 A first-order scheme using backward Euler

The first-order backward Euler method (3.1.18) is a one-stage Runge-Kutta method,

$$U^{n+1} = U^n + \Delta t \mathcal{L}(U^{n+1}; t^{n+1}), \tag{3.2.3}$$

where $\mathcal{L}(U; t) = (d_1^2 \partial_x^2 + d_2^2 \partial_y^2) U$. Using backward Euler to discretize equation (3.2.2), we update the low-rank solution using the unconventional DLR framework [33]; our treatment will differ from the unconventional DLR method once we consider higher-order time-integrators. The K and L steps will *discretize, freeze and project* the differential equation in each dimension to compute the updated bases. The S step will use the updated bases from the K and L step to *project* the differential equation in both dimensions, and then *compress* the solution.

K and L steps

Discretizing equation (3.2.2) using backward Euler,

$$\begin{aligned} \mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T - \Delta t \mathbf{D}^x \mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T \\ - \Delta t \mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{D}^y \mathbf{V}^{y,n+1})^T = \mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T. \end{aligned} \tag{3.2.4}$$

Freezing the solution in the $y-$dimension, we can update the $x-$basis $\mathbf{V}^x$; and vice versa. Looking at equation (3.2.4), freezing the solution in one dimension is challenging because doing so requires knowledge of the frozen basis at times t^n and t^{n+1}.

By first approximating the one-dimensional bases at the future time with *approximate bases*, $\mathbf{V}^{x,\star}$ and $\mathbf{V}^{y,\star}$, we can then solve for the updated one-dimensional bases, $\mathbf{V}^{x,n+1}$ and $\mathbf{V}^{y,n+1}$. The approximate bases are defined using known information. The approximate bases for the first-order scheme are defined by

$$\mathbf{V}^{x,\star} \coloneqq \mathbf{V}^{x,n} \in \mathbb{R}^{N_x \times r^n}, \tag{3.2.5a}$$

$$\mathbf{V}^{y,\star} \coloneqq \mathbf{V}^{y,n} \in \mathbb{R}^{N_y \times r^n}. \tag{3.2.5b}$$

The approximate bases used to freeze the solution introduce a temporal error since they are approximating the bases at time t^{n+1}. Using only the current bases at time t^n introduces a $\mathcal{O}(\Delta t)$ error. This is okay since we are computing a first-order approximation. However, higher-order time integrators will require much richer approximate bases.

After substituting $\mathbf{V}^{y,n+1}$ with $\mathbf{V}^{y,\star}$, *projecting* equation (3.2.4) onto the column space of $\mathbf{V}^{y,\star}$ yields the Sylvester equation

$$(\mathbf{I} - \Delta t \mathbf{D}^x)\mathbf{K}^{n+1} + \mathbf{K}^{n+1}\left(-\Delta t (\mathbf{D}^y \mathbf{V}^{y,\star})^T \mathbf{V}^{y,\star}\right) = \mathbf{K}^n, \tag{3.2.6}$$

where $\mathbf{K}^n = \mathbf{V}^{x,n}\mathbf{S}^n$. Similarly, substituting $\mathbf{V}^{x,n+1}$ with $\mathbf{V}^{x,\star}$ and projecting equation (3.2.4) onto the column space of $\mathbf{V}^{x,\star}$ yields the Sylvester equation

$$(\mathbf{I} - \Delta t \mathbf{D}^y)\mathbf{L}^{n+1} + \mathbf{L}^{n+1}\left(-\Delta t (\mathbf{D}^x \mathbf{V}^{x,\star})^T \mathbf{V}^{x,\star}\right) = \mathbf{L}^n, \tag{3.2.7}$$

where $\mathbf{L}^n = \left(\mathbf{S}^n (\mathbf{V}^{y,n})^T\right)^T = \mathbf{V}^{y,n}(\mathbf{S}^n)^T$.

After solving the Sylvester equations (3.2.6), (3.2.7) for $\mathbf{K}^{n+1}$ and $\mathbf{L}^{n+1}$, one can easily compute the updated orthonormal bases. Recall that the column spaces of $\mathbf{K}^{n+1}$ and $\mathbf{L}^{n+1}$ are the same as the column spaces of $\mathbf{V}^{x,n+1}$ and $\mathbf{V}^{y,n+1}$, respectively. Computing the reduced QR factorizations $\mathbf{K}^{n+1} = \mathbf{Q}_x\mathbf{R}_x$ and $\mathbf{L}^{n+1} = \mathbf{Q}_y\mathbf{R}_y$, the

updated bases are defined by

$$\mathbf{V}^{x,n+1} \coloneqq \mathbf{Q}_x, \qquad \mathbf{V}^{y,n+1} \coloneqq \mathbf{Q}_y.$$

Remark 3.9. As discussed later in Section 3.2.3.1, diagonalizing a Sylvester equation $\mathbf{AX} + \mathbf{XB} = \mathbf{C}$ leads to greater efficiency. By "diagonalizing a Sylvester equation" we mean diagonalizing the differential operator such that $\mathbf{A}$ is diagonal. Since the differential operator in equation (3.0.2) is so nice, this diagonalization is straightforward. One can compute the eigenvalue decomposition of the discrete Laplacian and then easily diagonalize $\mathbf{I} - \Delta t\mathbf{D}^x$ and $\mathbf{I} - \Delta t\mathbf{D}^y$. Without loss of generality, let $\mathbf{W}^x\mathbf{Z}^x(\mathbf{W}^x)^T$ be the diagonalization of $\mathbf{I} - \Delta t\mathbf{D}^x$. Defining $\tilde{\mathbf{K}}^{n+1} \coloneqq (\mathbf{W}^x)^T\mathbf{K}^{n+1}$, equation (3.2.6) reduces to

$$\mathbf{Z}^x\tilde{\mathbf{K}}^{n+1} + \tilde{\mathbf{K}}^{n+1}\left(-\Delta t(\mathbf{D}^y\mathbf{V}_1^{y,\star})^T\mathbf{V}_1^{y,\star}\right) = (\mathbf{W}^x)^T\mathbf{K}^n. \tag{3.2.8}$$

Solving equation (3.2.8) for $\tilde{\mathbf{K}}^{n+1}$ leads to $\mathbf{K}^{n+1} = \mathbf{W}^x\tilde{\mathbf{K}}^{n+1}$. Similarly, we can diagonalize equation (3.2.7). It is important to note that diagonalizing the Sylvester equations is only advantageous if the differential operator is time-independent. Otherwise, an eigenvalue decomposition will need to be computed for each time-step rather than once and for all.

S step

Projecting equation (3.2.4) in both dimensions onto the column spaces of the updated bases $\mathbf{V}^{x,n+1}$ and $\mathbf{V}^{y,n+1}$,

$$\begin{aligned}\mathbf{S}^{n+1} - \Delta t(\mathbf{V}^{x,n+1})^T\mathbf{D}^x\mathbf{V}^{x,n+1}\mathbf{S}^{n+1} - \Delta t\mathbf{S}^{n+1}(\mathbf{D}^y\mathbf{V}^{y,n+1})^T\mathbf{V}^{y,n+1} \\ = (\mathbf{V}^{x,n+1})^T\mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T\mathbf{V}^{y,n+1}.\end{aligned} \tag{3.2.9}$$

Since $-\Delta t(\mathbf{V}^{x,n+1})^T\mathbf{D}^x\mathbf{V}^{x,n+1}$ and $-\Delta t(\mathbf{D}^y\mathbf{V}^{y,n+1})^T\mathbf{V}^{y,n+1}$ are symmetric $r^n \times r^n$ matrices, we transform equation (3.2.9) into a more efficient form. Computing the

eigenvalue decompositions of the real symmetric matrices

$$-\Delta t(\mathbf{V}^{x,n+1})^T\mathbf{D}^x\mathbf{V}^{x,n+1} = \mathbf{Q}^x\mathbf{\Lambda}^x(\mathbf{Q}^x)^T, \qquad -\Delta t(\mathbf{D}^y\mathbf{V}^{y,n+1})^T\mathbf{V}^{y,n+1} = \mathbf{Q}^y\mathbf{\Lambda}^y(\mathbf{Q}^y)^T$$

requires $\mathcal{O}(r^3)$ flops since the matrices are symmetric but not tridiagonal [76]. Letting

$$\tilde{\mathbf{S}}^{n+1} = (\mathbf{Q}^x)^T\mathbf{S}^{n+1}\mathbf{Q}^y, \tag{3.2.10a}$$

$$\tilde{\mathbf{B}} = (\mathbf{V}^{x,n+1}\mathbf{Q}^x)^T\mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T\mathbf{V}^{y,n+1}\mathbf{Q}^y, \tag{3.2.10b}$$

equation (3.2.9) becomes the Sylvester equation

$$(\mathbf{I} + \mathbf{\Lambda}^x)\tilde{\mathbf{S}}^{n+1} + \tilde{\mathbf{S}}^{n+1}\mathbf{\Lambda}^y = \tilde{\mathbf{B}}. \tag{3.2.11}$$

Solving equation (3.2.11) has a relatively small computational complexity. The updated $\mathbf{S}^{n+1}$ is obtained by

$$\mathbf{S}^{n+1} = \mathbf{Q}^x\tilde{\mathbf{S}}^{n+1}(\mathbf{Q}^y)^T.$$

Compressing the updated solution $\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T$ is done via the SVD. Following the step-and-truncate procedure in Section 3.1.2.1, let $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the SVD of $\mathbf{S}^{n+1}$ and r^{n+1} be the number of singular values larger than some small tolerance $\epsilon > 0$. Redefine the bases to be

$$\mathbf{V}^{x,n+1} \coloneqq \mathbf{V}^{x,n+1}\mathbf{U}_{:,1:r^{n+1}}, \quad \mathbf{S}^{n+1} \coloneqq \mathbf{\Sigma}_{1:r^{n+1},1:r^{n+1}}, \quad \mathbf{V}^{y,n+1} \coloneqq \mathbf{V}^{y,n+1}\mathbf{V}_{:,1:r^{n+1}}. \tag{3.2.12}$$

The presented first-order scheme used the backward Euler discretization. The K and L steps discretized the equation using the first-order implicit integrator, froze the solution in one dimension at a time, and projected the equation onto the spaces spanned by approximate bases. The S step then projected the equation onto the space spanned by both pre-compressed updated bases to update the transfer matrix, and compressed the solution by a step-and-truncate procedure. The non-diagonalized

first-order scheme with backward Euler is outlined in Algorithm 3.1. We present the algorithm for homogeneous Dirichlet boundary conditions, but other boundary conditions would only require slight modifications.

To initialize the scheme, we compute the full SVD of $\mathbf{U}^0$, only keeping the first r^0 singular values and their corresponding singular vectors. In this initialization we *do not* truncate based on a tolerance $\epsilon > 0$. Numerical tests showed that keeping a larger/richer initial basis is important for capturing the immediate dynamics of diffusion problems. Even if the initial condition is theoretically low-rank, the immediate diffusion dynamics could cause the rank to instantaneously increase. This requires a richer initial basis that can appropriately capture the solution at the future time $t = t^1$. As such, we keep the first $r^0 = \text{ceil}(\max(N_x, N_y)/N_0)$ singular vectors/values of the full SVD of the initial condition, for some N_0. Although not rigorously justified, we found $N_0 = 3$ to produce a sufficiently rich enough basis. This high-rank initial condition does not dramatically affect the overall computational complexity since it is only for a single time-step, whereas every subsequent time-step truncates the solution by tolerance $\epsilon > 0$. We typically set $\epsilon \in [1.0E-12, 1.0E-08]$ except for problems where the solution decays very rapidly and might require a smaller ϵ.

As per Remark 3.9, the appropriate diagonalizations must also be computed when using the diagonalized variant of the first-order scheme. Assuming a time-independent differential operator, this should only need to be performed once in the initialization.

Algorithm 3.1. First-order scheme with backward Euler

Inputs: $\mathbf{U}^n = \mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T$; rank r^n.

Outputs: $\mathbf{U}^{n+1} = \mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T$; rank r^{n+1}.

***K* and *L* Steps**

Let subscript $_{int}$ denote the interior nodes.

1a. Let $\mathbf{V}^{x,n+1} = \mathbf{0} \in \mathbb{R}^{N_x \times r^n}$ and $\mathbf{V}^{y,n+1} = \mathbf{0} \in \mathbb{R}^{N_y \times r^n}$.

1b. Define $\mathbf{V}^{x,\star}$ and $\mathbf{V}^{y,\star}$ according to equation (3.2.5).

1c. Compute $\mathbf{K}_{int}^{n+1}$ and $\mathbf{L}_{int}^{n+1}$ from equations (3.2.6) and (3.2.7).

1d. Compute reduced QR factorization $\mathbf{K}_{int}^{n+1} = \mathbf{Q}_x\mathbf{R}_x$, set $\mathbf{V}^{x,n+1}(2 : N_x - 1, :) = \mathbf{Q}_x$.

1e. Compute reduced QR factorization $\mathbf{L}_{int}^{n+1} = \mathbf{Q}_y\mathbf{R}_y$, set $\mathbf{V}^{y,n+1}(2 : N_y - 1, :) = \mathbf{Q}_y$.

S **Step**

2a. Compute $\tilde{\mathbf{S}}^{n+1}$ from equation (3.2.11).

2b. Compute $\mathbf{S}^{n+1}$ from $\tilde{\mathbf{S}}^{n+1}$.

2c. Redefine $\mathbf{V}^{x,n+1}$, $\mathbf{S}^{n+1}$, $\mathbf{V}^{y,n+1}$ and r^{n+1} according to equation (3.2.12).

3.2.2 A second-order scheme using DIRK2

The scheme formulation using second-order time-discretizations is a simple extension of the first-order scheme. Since DIRK2 has two stages, there will be two $K - L - S$ phases; herein lies the difference between our proposed algorithm and the unconventional DLR method. To accommodate higher-order multistage DIRK integrators, the updated bases from the first $K - L - S$ phase will be used to define the approximate bases for the second $K-L-S$ phase. We present the second-order scheme using the stiffly-accurate DIRK2 [55]. The second-order schemes using CN and BDF2 follow similarly and are included in Appendices B and C, respectively. The second-order diagonally implicit Runge-Kutta (DIRK2) method is a two-stage Runge-Kutta method,

$$U^{(1)} = U^n + \gamma\Delta t\mathcal{L}(U^{(1)}; t^{(1)}), \tag{3.2.13a}$$

$$U^{n+1} = U^n + (1-\gamma)\Delta t\mathcal{L}(U^{(1)}; t^{(1)}) + \gamma\Delta t\mathcal{L}(U^{n+1}; t^{n+1}), \tag{3.2.13b}$$

where $\mathcal{L}(U; t) = (d_1^2\partial_x^2 + d_2^2\partial_y^2)U$, $t^{(1)} = t^n + \gamma\Delta t$ and $\gamma = 1 - 1/\sqrt{2}$. Using stiffly-accurate DIRK2 to discretize equation (3.2.2), we can update the bases of the low-rank solution in a two-stage $K - L - S$ fashion similar to the first-order scheme.

$K-L-S$ phase 1

Observe that the first stage of DIRK2 is simply the backward Euler integrator over a time-step of $\Delta t^{(1)} = \gamma \Delta t$ from time t^n to $t^{(1)}$. Following Algorithm 3.1, we obtain the low-rank solution $\mathbf{U}^{(1)} = \mathbf{V}^{x,(1)}\mathbf{S}^{(1)}(\mathbf{V}^{y,(1)})^T$ of rank $r^{(1)}$.

$K-L-S$ phase 2: K and L steps

Discretizing equation (3.2.2) using the second stage formula of DIRK2,

$$\begin{aligned}\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T - \gamma\Delta t\mathbf{D}^x\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T \\ - \gamma\Delta t\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{D}^y\mathbf{V}^{y,n+1})^T = RHS,\end{aligned} \tag{3.2.14}$$

$$\begin{aligned}RHS = \mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T + (1-\gamma)\Delta t\mathbf{D}^x\mathbf{V}^{x,(1)}\mathbf{S}^{(1)}(\mathbf{V}^{y,(1)})^T \\ + (1-\gamma)\Delta t\mathbf{V}^{x,(1)}\mathbf{S}^{(1)}(\mathbf{D}^y\mathbf{V}^{y,(1)})^T.\end{aligned} \tag{3.2.15}$$

Looking at equation (3.2.4), *freezing* the solution one dimension at a time requires knowledge of the frozen bases at times t^n, $t^{(1)}$ and t^{n+1}. We know the bases at times t^n and $t^{(1)}$, but not at time t^{n+1}. The approximate bases 3.2.5 used in the first-order scheme will not suffice since the $\mathcal{O}(\Delta t)$ error will destroy the desired second-order accuracy. However, we can augment the bases from the previous Runge-Kutta stages to form richer bases that will (hopefully) not destroy the accuracy. Consider the augmented bases

$$\left[\mathbf{V}^{x,n} \;\middle|\; \mathbf{V}^{x,(1)}\right] \in \mathbb{R}^{N_x \times (r^n + r^{(1)})}, \quad \left[\mathbf{V}^{y,n} \;\middle|\; \mathbf{V}^{y,(1)}\right] \in \mathbb{R}^{N_y \times (r^n + r^{(1)})}.$$

These augmented bases span richer spaces that approximate the solution at time t^{n+1} using information from the previous stages at times t^n and $t^{(1)}$. Assuming the solution does not change much over a single time-step, the augmented bases might have several redundancies. Computing the reduced SVDs of the augmented bases, let $\mathbf{U}^x$ be the left singular vectors of $\left[\mathbf{V}^{x,n}|\mathbf{V}^{x,(1)}\right]$, and $\mathbf{U}^y$ be the left singular vectors of $\left[\mathbf{V}^{y,n}|\mathbf{V}^{y,(1)}\right]$. Further let r^x and r^y be the respective number of singular values

greater than the same tolerance ϵ. The approximate bases for the second stage of the second-order scheme are defined by

$$\mathbf{V}^{x,\star} := \mathbf{U}^x(:, 1:r), \tag{3.2.16a}$$

$$\mathbf{V}^{y,\star} := \mathbf{U}^y(:, 1:r), \tag{3.2.16b}$$

where $r = \max(r^x, r^y)$. We let the rank be the maximum of r^x and r^y since we need the approximate bases to have the same number of (column) vectors.

After substituting $\mathbf{V}^{y,n+1}$ with $\mathbf{V}^{y,\star}$, *projecting* equation (3.2.14) onto the column space of $\mathbf{V}^{y,\star}$ yields the Sylvester equation

$$(\mathbf{I} - \gamma\Delta t\mathbf{D}^x)\mathbf{K}^{n+1} + \mathbf{K}^{n+1}\left(-\gamma\Delta t(\mathbf{D}^y\mathbf{V}^{y,\star})^T\mathbf{V}^{y,\star}\right) = (RHS)\mathbf{V}^{y,\star}, \tag{3.2.17}$$

where $\mathbf{K}^n = \mathbf{V}^{x,n}\mathbf{S}^n$. Similarly, substituting $\mathbf{V}^{x,n+1}$ with $\mathbf{V}^{x,\star}$ and projecting equation (3.2.14) onto the column space of $\mathbf{V}^{x,\star}$ yields the Sylvester equation

$$(\mathbf{I} - \gamma\Delta t\mathbf{D}^y)\mathbf{L}^{n+1} + \mathbf{L}^{n+1}\left(-\gamma\Delta t(\mathbf{D}^x\mathbf{V}^{x,\star})^T\mathbf{V}^{x,\star}\right) = (RHS)^T\mathbf{V}^{x,\star}, \tag{3.2.18}$$

where $\mathbf{L}^n = \left(\mathbf{S}^n(\mathbf{V}^{y,n})^T\right)^T = \mathbf{V}^{y,n}(\mathbf{S}^n)^T$.

After solving the Sylvester equations (3.2.17), (3.2.18) for $\mathbf{K}^{n+1}$ and $\mathbf{L}^{n+1}$, one can easily compute the updated orthonormal bases like in the first-order scheme. Computing the reduced QR factorizations $\mathbf{K}^{n+1} = \mathbf{Q}_x\mathbf{R}_x$ and $\mathbf{L}^{n+1} = \mathbf{Q}_y\mathbf{R}_y$, the updated bases are defined by

$$\mathbf{V}^{x,n+1} := \mathbf{Q}_x, \qquad \mathbf{V}^{y,n+1} := \mathbf{Q}_y.$$

Remark 3.10. If the diagonalized variant of the second-order scheme is desired, then the matrices $\mathbf{I} - \gamma\Delta t\mathbf{D}^x$ and $\mathbf{I} - \gamma\Delta t\mathbf{D}^y$ will need to be diagonalized. Instead of solving

equation (3.2.8), the second stage of the second-order scheme will solve

$$\mathbf{Z}^x\tilde{\mathbf{K}}^{n+1} + \tilde{\mathbf{K}}^{n+1}\left(-\gamma\Delta t(\mathbf{D}^y\mathbf{V}_1^{y,\star})^T\mathbf{V}_1^{y,\star}\right) = (\mathbf{W}^x)^T\left((RHS)\mathbf{V}^{y,\star}\right). \tag{3.2.19}$$

The diagonalized variant of equation (3.2.18) follows similarly. It is important to follow the order of operations on the righthand side of equation (3.2.19). Doing so will cost $\mathcal{O}(N^2r)$ flops for the matrix multiplication; otherwise, the cost will be $\mathcal{O}(N^3)$.

$K-L-S$ phase 2: S step

Projecting equation (3.2.14) in both dimensions onto the column spaces of the updated bases $\mathbf{V}^{x,n+1}$ and $\mathbf{V}^{y,n+1}$,

$$\begin{aligned}\mathbf{S}^{n+1} - \gamma\Delta t(\mathbf{V}^{x,n+1})^T\mathbf{D}^x\mathbf{V}^{x,n+1}\mathbf{S}^{n+1} - \gamma\Delta t\mathbf{S}^{n+1}(\mathbf{D}^y\mathbf{V}^{y,n+1})^T\mathbf{V}^{y,n+1} \\ = (\mathbf{V}^{x,n+1})^T(RHS)\mathbf{V}^{y,n+1}.\end{aligned} \tag{3.2.20}$$

Computing the eigenvalue decompositions of the real symmetric matrices

$$-\gamma\Delta t(\mathbf{V}^{x,n+1})^T\mathbf{D}^x\mathbf{V}^{x,n+1} = \mathbf{Q}^x\mathbf{\Lambda}^x(\mathbf{Q}^x)^T, \qquad -\gamma\Delta t(\mathbf{D}^y\mathbf{V}^{y,n+1})^T\mathbf{V}^{y,n+1} = \mathbf{Q}^y\mathbf{\Lambda}^y(\mathbf{Q}^y)^T$$

requires $\mathcal{O}(r^3)$ flops since the matrices are symmetric but not tridiagonal [76]. Letting

$$\tilde{\mathbf{S}}^{n+1} = (\mathbf{Q}^x)^T\mathbf{S}^{n+1}\mathbf{Q}^y, \tag{3.2.21a}$$

$$\tilde{\mathbf{B}} = (\mathbf{V}^{x,n+1}\mathbf{Q}^x)^T(RHS)\mathbf{V}^{y,n+1}\mathbf{Q}^y, \tag{3.2.21b}$$

equation (3.2.20) becomes the Sylvester equation

$$(\mathbf{I} + \mathbf{\Lambda}^x)\tilde{\mathbf{S}}^{n+1} + \tilde{\mathbf{S}}^{n+1}\mathbf{\Lambda}^y = \tilde{\mathbf{B}}. \tag{3.2.22}$$

Solving equation (3.2.22) has a relatively small computational complexity. The updated $\mathbf{S}^{n+1}$ is obtained by

$$\mathbf{S}^{n+1} = \mathbf{Q}^x\tilde{\mathbf{S}}^{n+1}(\mathbf{Q}^y)^T.$$

Compressing the updated solution $\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T$ is done just like in the first-order scheme. Let $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the SVD of $\mathbf{S}^{n+1}$ and r^{n+1} be the number of singular values larger than some small tolerance $\epsilon > 0$. Redefine the bases to be

$$\mathbf{V}^{x,n+1} \coloneqq \mathbf{V}^{x,n+1}\mathbf{U}_{:,1:r^{n+1}}, \quad \mathbf{S}^{n+1} \coloneqq \mathbf{\Sigma}_{1:r^{n+1},1:r^{n+1}}, \quad \mathbf{V}^{y,n+1} \coloneqq \mathbf{V}^{y,n+1}\mathbf{V}_{:,1:r^{n+1}}. \tag{3.2.23}$$

We outline the second-order scheme with stiffly-accurate DIRK2 below in Algorithm 3.2. We present the algorithm for homogeneous Dirichlet boundary conditions, but other boundary conditions would only require slight modifications. The initialization procedure is the same as for Algorithm 3.1.

Algorithm 3.2. Second-order scheme with stiffly-accurate DIRK2

Inputs: $\mathbf{U}^n = \mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T$; rank r^n.

Outputs: $\mathbf{U}^{n+1} = \mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T$; rank r^{n+1}.

$K-L-S$ **Phase 1**

Compute the rank $r^{(1)}$ solution $\mathbf{U}^{(1)} = \mathbf{V}^{x,(1)}\mathbf{S}^{(1)}(\mathbf{V}^{y,(1)})^T$ using Algorithm 3.1 over time-step $\gamma\Delta t$.

$K-L-S$ **Phase 2:** K **and** L **Steps**

Let subscript $_{int}$ denote the interior nodes.

1a. Let $\mathbf{V}^{x,n+1} = \mathbf{0} \in \mathbb{R}^{N_x \times r^n}$ and $\mathbf{V}^{y,n+1} = \mathbf{0} \in \mathbb{R}^{N_y \times r^n}$.

1b. Construct the augmented bases $\left[\mathbf{V}^{x,n} \mid \mathbf{V}^{x,(1)}\right]$ and $\left[\mathbf{V}^{y,n} \mid \mathbf{V}^{y,(1)}\right]$, and define $\mathbf{V}^{x,\star}$ and $\mathbf{V}^{y,\star}$ according to equation (3.2.16).

1c. Compute $\mathbf{K}_{int}^{n+1}$ and $\mathbf{L}_{int}^{n+1}$ from equations (3.2.17) and (3.2.18).

1d. Compute reduced QR factorization $\mathbf{K}_{int}^{n+1} = \mathbf{Q}_x\mathbf{R}_x$, set $\mathbf{V}^{x,n+1}(2 : N_x - 1, :) = \mathbf{Q}_x$.

1e. Compute reduced QR factorization $\mathbf{L}_{int}^{n+1} = \mathbf{Q}_y\mathbf{R}_y$, set $\mathbf{V}^{y,n+1}(2 : N_y - 1, :) = \mathbf{Q}_y$.

$K-L-S$ **Phase 2:** S **Step**

2a. Compute $\tilde{\mathbf{S}}^{n+1}$ from equation (3.2.22).

2b. Compute $\mathbf{S}^{n+1}$ from $\tilde{\mathbf{S}}^{n+1}$.

2c. Redefine $\mathbf{V}^{x,n+1}$, $\mathbf{S}^{n+1}$, $\mathbf{V}^{y,n+1}$ and r^{n+1} according to equation (3.2.23).

3.2.3 Computational complexity

We claim that the proposed implicit low-rank integrators have significant computational savings when solutions have low-rank structure. The computational complexities of the first- and second-order schemes are the same order of magnitude. Since the second-order scheme has two stages instead of one, its computational complexity is roughly twice that of the first-order scheme. The computational complexity for each time-step is dominated by the Sylvester equation. As such, we first outline the expected cost of solving the Sylvester equation in Algorithms 3.1 and 3.2. Then, we summarize the computational complexity of the proposed scheme.

3.2.3.1 Computational complexity of solving the Sylvester equation

We discuss the computational complexity required to set up and solve the Sylvester equation

$$\mathbf{AX} + \mathbf{XB} = \mathbf{C},$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$, $\mathbf{B} \in \mathbb{R}^{r\times r}$, $\mathbf{C} \in \mathbb{R}^{N\times r}$ and solution $\mathbf{X} \in \mathbb{R}^{N\times r}$. The Bartels-Stewart algorithm [14] is a popular way to numerically solve Sylvester equations for the two-dimensional case, with a computational complexity of $\sim 10(N^3 + r^3) + \frac{5}{2}(N^2 r + N r^2)$ flops; the cubic terms come from computing the Schur decompositions of $\mathbf{A}$ and $\mathbf{B}$. Another algorithm is the Hessenberg-Schur variant of the Bartels-Stewart algorithm [75], with a computational complexity of $\sim \frac{5}{3}N^3 + 10r^3 + 5N^2 r + \frac{5}{2}N r^2$ flops; the first cubic term come from using Householder reflectors on $\mathbf{A}$ instead of computing its Schur decomposition. If A has diagonal structure, then the computational cost is reduced to $\mathcal{O}(N^2 r)$. The Hessenberg-Schur variant is implemented in the SLICOT library [16, 176]

and is used by MATLAB's control system toolbox. The SLICOT library is built on LAPACK and BLAS collection [5, 18].

There are two variants of the proposed scheme: (1) solving non-diagonalized Sylvester equations, e.g., equation (3.2.6), and (2) solving diagonalized Sylvester equations, e.g., equation (3.2.8). Table 3.1 shows the dominant computational cost to compute $\mathbf{C}$ and solve the Sylvester equation. We show both the full-rank ($N \times N$) and low-rank ($N \times r$) systems to demonstrate the computational savings when solutions are low-rank. In addition, the non-diagonalized and diagonalized variants of the proposed scheme are compared.

	Cost to set up $\mathbf{C}$		Cost to solve the Sylvester equation	
	Full-rank	Low-rank	Full-rank	Low-rank
Non-diagonalized variant	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2r)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N^3)$
Diagonalized variant	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2r)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2r)$

Table 3.1: The dominant computational cost to set up and solve the Sylvester equations applicable to the proposed implicit low-rank integrator. The reference Sylvester equations are equations (3.2.6) and (3.2.8).

As seen in Table 3.1, setting up the Sylvester equations for low-rank solutions observes quadratic computational complexity compared to cubic computational complexity for full-rank solutions; this is simply from the matrix multiplication. Moreover, the computational complexity to set up and solve the Sylvester equation in the low-rank setting with the diagonalized variant is dominated by $\mathcal{O}(N^2r)$.

Futher note that the computational complexity for solving the Sylvester equation with the non-diagonalized variant is $\mathcal{O}(N^3)$ for both full-rank and low-rank solutions. Even though the full-rank and low-rank solutions both require cubic complexity, the low-rank solution is still advantageous. Recall the computational complexity of the Hessenberg-Schur variant of the Bartels-Stewart algorithm. The computational complexity to solve a Sylvester equation for a full-rank $N \times N$ solution is $\sim \frac{115}{6}N^3$ flops, or rather, roughly $\sim 19N^3$ flops. Whereas, the computational complexity to solve a Sylvester equation for a low-rank $N \times r$ solution is dominated by $\sim \frac{5}{3}N^3$ flops.

Despite both having a computational complexity of $\mathcal{O}(N^3)$ flops, the low-rank solution is roughly ten times cheaper than the full-rank solution.

3.2.3.2 Computational complexity of the proposed scheme

We provide a very rough estimate of the computational complexity for Algorithm 3.1; the computational complexity for Algorithm 3.2 is similar. Any computation performed outside the time-stepping loop is not considered (e.g., computing the full SVD of the initial condition $\mathbf{U}^0$ for the initialization, and computing the error after the final time-step). Since it is difficult to determine the exact computational complexities of the algorithms used by LAPACK and BLAS for the MATLAB functions svd, qr, and sylvester, we use the computational complexities outlined in this chapter as rough estimates.

For simplicity we assume $N = N_x = N_y$ and $r = r^n \ll N$. Computing the matrices for the Sylvester equations is dominated by $\mathcal{O}(N^2 r)$ flops. Solving the non-diagonalized Sylvester equations (3.2.6) and (3.2.7) is dominated by $\mathcal{O}(N^3)$ flops; solving the diagonalized Sylvester equation (3.2.8) is dominated by $\mathcal{O}(N^2 r)$ flops. All other reduced QR factorizations, reduced SVDs and matrix multiplications are dominated by $\mathcal{O}(Nr^2)$ flops. All together, the total computational complexity of Algorithm 3.1 (for a single time-step) is roughly dominated by $\mathcal{O}(N^3)$ flops if using the non-diagonalized variant of the algorithm. Otherwise, the total computational complexity of Algorithm 3.1 (for a single time-step) is roughly dominated by $\mathcal{O}(N^2 r)$ flops if using the diagonalized variant.

Remark 3.11. (Initializing the scheme). The diagonalized variant of the proposed scheme requires the diagonalization of $\mathbf{I} - \Delta t \mathbf{D}^x$ and $\mathbf{I} - \Delta t \mathbf{D}^y$. The computational cost of this diagonalization is $\mathcal{O}(N^3)$, but it only needs to be performed once if the diffusion operators $\mathbf{D}^x$, $\mathbf{D}^y$ are time-independent and Δt remains fixed. Furthermore, initializing the proposed scheme requires computing the full SVD of the initial condition, which also has a computational cost of $\mathcal{O}(N^3)$. Given that most practical simulations set at most $N \sim 1000$ and run to large final times, meaning many time-steps, the initialization

cost is very tolerable. A decent portion of the CPU runtime for small final times is attributed to the intialization procedure, but for large final times the CPU runtime is dominated by the time-stepping.

3.3 Numerical tests

In this section, we compare the performance of the first- and second-order schemes. The CPU runtime and spatial and temporal convergence are presented for various initial conditions. We solve equation (3.0.2),

$$u_t = d_1^2 u_{xx} + d_2^2 u_{yy}, \qquad (x, y) \in (0, 1)^2, \qquad t > 0, \tag{3.3.1}$$

for constants $d_1, d_2 > 0$. Homogeneous Dirichlet boundary conditions are assumed, and the final time is $T_f = 1$ unless otherwise stated. The three initial conditions we consider are

$$\begin{aligned} u_0(x, y) = 0.5\exp\left[-400\left((x - 0.3)^2 + (y - 0.35)^2\right)\right] \\ + 0.8\exp\left[-400\left((x - 0.65)^2 + (y - 0.5)^2\right)\right], \end{aligned} \tag{3.3.2}$$

$$u_0(x, y) = \sum_{m=1}^{3} \sin(m\pi x) \sin(m\pi y), \tag{3.3.3}$$

$$u_0(x, y) = \frac{\sin(\pi x) \sin(\pi y)}{1 + (x - 0.5)^2 + (y - 0.5)^2}, \tag{3.3.4}$$

where initial conditions (3.3.2)-(3.3.4) are order-2 tensors of theoretical rank one, rank two and infinite rank, respectively. Initial condition (3.3.2) is used for diffusion coefficients $d_1 = d_2 = 1/2$; initial conditions (3.3.3) and (3.3.4) are used for diffusion coefficients $d_1 = 1/2$ and $d_2 = 1/3$. To enforce homogeneous Dirichlet boundary conditions, the scaling of the Gaussians/Maxwellians in initial condition (3.3.2) is large enough so that the function is less than machine precision on the boundary.

The reasons for using initial condition (3.3.2) are: it activates several Fourier modes, and it represents distribution functions commonly seen in some kinetic models.

The reason for using initial condition (3.3.3) is it has similar structure to the solution since it is just the first three Fourier modes. The reasons for using initial condition (3.3.4) are: it has theoretical infinite rank, and the numerator offers some structure similar to the first Fourier mode.

Recall that we *do not* truncate the initial condition based on a tolerance $\epsilon > 0$. We found that a larger/richer initial basis was necessary to observe spatial convergence. This is because although the initial condition might be low-rank, the initial dynamics of diffusion could make the rank increase instantaneously; for some initial conditions, this jump in rank could be quite large. In order to capture the correct initial dynamics, the initial time-step needs a richer approximate basis than the theoretical low-rank basis of the initial condition. For instance, initial condition (3.3.2) is a rank-2 tensor, but those two basis functions will not provide a rich enough (approximate) basis for the higher-rank solution at the next time-step. As such, we keep several singular vectors from the full SVD of the initial condition; the singular values that are theoretically zero are on the order of $1.0E-15$ when computed in MATLAB. The initial rank is set as $r^0 = \text{ceil}(N/N_0)$ for integer $N_0 = 3$ unless otherwise stated.

The general Fourier series solution of equation (3.0.2) is

$$u(x,y,t) = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} \alpha_{nm} \exp\left(-(d_1^2 n^2 + d_2^2 m^2)\pi^2 t\right) \sin(n\pi x) \sin(m\pi y), \tag{3.3.5}$$

where $\alpha_{nm} = 4\int_0^1 \int_0^1 u_0(x,y) \sin(n\pi x) \sin(m\pi y) dx dy$. The solution corresponding to initial condition (3.3.3) falls out naturally by orthogonality. We compute the Fourier coefficients for the solutions corresponding to initial conditions (3.3.2) and (3.3.4) in Mathematica® and only keep the coefficients for which $\alpha_{nm} \exp\left(-(d_1^2 n^2 + d_2^2 m^2)\pi^2 t\right)$ is less than machine precision.

There are several sources of error. Spatial error is due to the second-order centered-difference approximation of the second-derivatives and the tolerance used to truncate the singular values in the basis removal procedure. Temporal error is due to

the time-discretization and the approximate bases. We compute the discrete ℓ^1 error,

$$\|\mathbf{U} - \mathbf{U}_{exact}\|_1 = \Delta x \Delta y \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \left|\mathbf{U}(i,j) - \mathbf{U}_{exact}(i,j)\right|. \tag{3.3.6}$$

To help better track the computational complexity, all numerical results included in this section assume $N_x = N_y = N$.

3.3.1 CPU runtime

In this subsection, we present the CPU runtimes of the first- and second-order schemes. We compare the runtimes for both variations of the algorithms using the diagonalized and non-diagonalized Sylvester equations. To help identify how much of the computational cost is due to each step, we break down the runtimes for the initialization and time-stepping phases. The initialization phase for the algorithm with diagonalization is further broken down into the time spent computing the full SVD of the initial condition and the eigenvalue decompositions. The time-stepping phase is broken down into the time spent on the *first* time-step and the average runtime spent per *subsequent* time-step. For both phases we then show the change in runtime after each refinement,

$$\log_2(\text{ratio of the runtimes})$$

to see the quadratic $\mathcal{O}(N^2 r)$ or cubic $\mathcal{O}(N^3)$ computational complexity. We do not include the cost of computing the ℓ^1 error. The CPU runtimes were computed in MATLAB R2019b on a Dell Inspiron 5570 laptop with the 8th Generation Intel® Core™ i7-8550U Processor and 16GB RAM.

Example 3.1. (A benchmark method)

We first present numerical results for a familiar benchmark method, the (classical) alternating-direction implicit (ADI) method [59, 140]. The results using the full-rank ADI method will serve as a comparison for the results using the proposed low-rank scheme. The ADI method used herein uses centered-differences in space and

Crank-Nicolson in time. The method is formally second-order in both space and time. We note that the second-order accuracy in time also comes from a second-order splitting error. For the sake of this dissertation, we omit the details of ADI method and leave their discussion to [59, 140].

Table 3.2: CPU runtime of the second-order ADI method. Final time $T_f = 1$ and time-stepping size $\Delta t = 0.05$.

N	Initial condition (3.3.2)			Initial condition (3.3.3)			Initial condition (3.3.4)		
	Total CPU runtime		ℓ^1 error	Total CPU runtime		ℓ^1 error	Total CPU runtime		ℓ^1 error
40	1.72E-01s	0.00	2.19E-05	6.25E-02s	0.00	1.80E-05	1.09E-01s	0.00	1.73E-05
80	7.81E-02s	-1.14	3.12E-05	1.25E-01s	1.00	3.38E-05	6.25E-02s	-0.81	3.26E-05
160	2.03E-01s	1.38	3.35E-05	2.19E-01s	0.81	3.78E-05	2.66E-01s	2.09	3.65E-05
320	1.56E+00s	2.94	3.41E-05	1.63E+00s	2.89	3.88E-05	1.56E+00s	2.56	3.73E-05
640	8.66E+00s	2.47	3.42E-05	8.67E+00s	2.42	3.90E-05	8.55E+00s	2.45	3.77E-05
1280	3.77E+01s	2.12	3.42E-05	3.69E+01s	2.09	3.91E-05	3.73E+01s	2.12	3.78E-05
2560	1.71E+02s	2.18	3.42E-05	1.68E+02s	2.19	3.91E-05	1.70E+02s	2.19	3.78E-05

The CPU runtimes for all three initial conditions (3.3.2)-(3.3.4) are shown in Table 3.2 with final time $T_f = 1$ and fixed time-stepping size $\Delta t = 0.05$. The results indicate quadratic computational complexity, if not slightly above. We remind the reader that the ADI method solves for the full-rank solution.

Example 3.2. (First-order scheme with backward Euler)

Since all three initial conditions had similar quantitative results, we only show the CPU runtime for initial condition (3.3.2). We compare the CPU runtimes using the diagonalized and non-diagonalized variants of Algorithm 3.1 in Table 3.3. The time-stepping size is fixed at $\Delta t = 0.05$ so that the computational complexity can be observed under mesh refinement. Since the solution is at final time $T_f = 1$, there are a total of 20 time-steps. As seen in the table, computing the full SVD of the initial condition makes up a considerable portion of the total CPU runtime. However, the initialization is done once and will become less relevant as more time-steps are taken. The initialization step also shows $\mathcal{O}(N^3)$ complexity. Although the SVD suggests $\mathcal{O}(N^4)$ complexity, this is likely from overloading the memory cache with arrays larger than a certain size. In our simulations the computational complexity of the full SVD

dropped back down to $\mathcal{O}(N^3)$ after another refinement level, but we did not include it in the table.

Table 3.3: CPU runtime of the first-order scheme with backward Euler. We use the rank-2 initial condition (3.3.2), initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 0.05$.

Diagonalized scheme

N	Initialization				Time-stepping				Total CPU runtime		ℓ^1 error
	SVD		Diagonalization		First time-step		Per time-step				
40	3.25E-02s	0.00	1.14E-03s	0.00	7.62E-03s	0.00	1.33E-03s	0.00	6.78E-02s	0.00	6.68E-05
80	6.88E-03s	-2.24	1.73E-03s	0.61	5.55E-03s	-0.46	9.69E-04s	-0.45	3.35E-02s	-1.02	6.66E-05
160	1.79E-02s	1.38	5.09E-03s	1.56	1.23E-02s	1.15	9.63E-04s	-0.01	5.45E-02s	0.70	6.63E-05
320	1.53E-02s	-0.22	1.06E-02s	1.06	2.92E-02s	1.24	2.05E-03s	1.09	9.62E-02s	0.82	6.60E-05
640	4.84E-02s	1.66	3.15E-02s	1.57	1.74E-01s	2.57	5.78E-03s	1.49	3.69E-01s	1.94	6.52E-05
1280	6.47E-01s	3.74	1.72E-01s	2.45	6.41E-01s	1.89	1.92E-02s	1.73	1.84E+00s	2.32	6.41E-05
2560	8.10E+00s	3.65	8.87E-01s	2.37	3.72E+00s	2.54	7.20E-02s	1.91	1.41E+01s	2.94	6.28E-05

Non-diagonalized scheme

N	Initialization		Time-stepping				Total CPU runtime		ℓ^1 error
	SVD		First time-step		Per time-step				
40	2.70E-02s	0.00	7.59E-03s	0.00	1.31E-03s	0.00	6.08E-02s	0.00	6.68E-05
80	6.52E-03s	-2.05	3.39E-03s	-1.16	1.99E-03s	0.61	4.98E-02s	-0.29	6.66E-05
160	1.45E-02s	1.15	9.39E-03s	1.47	2.97E-03s	0.58	8.33E-02s	0.74	6.62E-05
320	1.33E-02s	-0.12	3.37E-02s	1.84	6.59E-03s	1.15	1.79E-01s	1.10	6.55E-05
640	5.22E-02s	1.97	2.14E-01s	2.67	3.04E-02s	2.21	8.74E-01s	2.29	6.43E-05
1280	6.00E-01s	3.52	8.72E-01s	2.03	1.63E-01s	2.42	4.73E+00s	2.44	6.32E-05
2560	8.02E+00s	3.74	4.55E+00s	2.38	9.21E-01s	2.50	3.10E+01s	2.71	6.19E-05

The observed computational complexity of the first time-step somewhat matches the expected $\mathcal{O}(N^2 r^0) = \mathcal{O}(N^3/3)$ for the diagonalized variant, and $\mathcal{O}(N^3)$ for the non-diagonalized variant. More importantly, the observed computational complexity of the diagonalized variant for all other time-steps is $\mathcal{O}(N^2)$ since $r \ll N$; the rank evolution is shown in Section 3.3.3. The average CPU runtime per subsequent time-step is roughly an order of magnitude smaller for the diagonalized variant than that of the non-diagonalized variant.

The observed total computational complexity for both variants is $\mathcal{O}(N^3)$. The diagonalized variant observes cubic complexity since the initialization and first time-step make up a majority of the CPU runtime. However, this will decrease towards $\mathcal{O}(N^2)$ as the number of time-steps increases and the total runtime is dominated by

the time-stepping. Comparing the two variants, the diagonalized variant is roughly twice as fast as the non-diagonalized variant when taking 20 time-steps.

The non-diagonalized variant demonstrates cubic computational complexity, whereas the benchmark ADI method enjoys quadratic computational complexity. However, the results in Tables 3.2 and 3.3 suggest that a very fine mesh is required for the ADI method to outperform the non-diagonalized variant in terms of total CPU runtime. Even for the finest mesh that we tested, 2560×2560, the ADI method took just over five times as long to run as the non-diagonalized variant did.

Although a higher computational complexity, the non-diagonalized variant will be of greater importance when used to solve more complicated models. The $\mathcal{O}(N^3)$ complexity and overall runtime of the non-diagonalized variant are still tolerable, especially when we start considering high-dimensional problems such as those in kinetic dynamics. We remark that higher-dimensional analogues to these DLR-type methods have been developed using various tensor decompositions [33, 34, 108, 124, 126, 127]. Storing these high-order tensor decompositions (e.g., Tucker, hierarchical Tucker, tensor train, tree tensor networks) either avoids or minimizes the curse of dimensionality.

Example 3.3. (Second-order scheme with stiffly-accurate DIRK2)

Since all three initial conditions had similar quantitative results, we only show the CPU runtime for initial condition (3.3.2). We compare the CPU runtimes using the diagonalized and non-diagonalized variants of Algorithm 3.2 in Table 3.4. The time-stepping size is fixed at $\Delta t = 0.05$ so that the computational complexity can be observed under mesh refinement. Since the solution is at final time $T_f = 1$, there are a total of 20 time-steps. As seen in the table, computing the full SVD of the initial condition has the same CPU runtime as in Table 3.3 for the first-order scheme. However, the initialization makes up a smaller proportion of the overall runtime than in the first-order case since Algorithm 3.2 is essentially Algorithm 3.1 performed twice.

The observed computational complexities for both the diagonalized and non-diagonalized variants of Algorithm 3.2 are similar to those of Algorithm 3.1, as compared with the results in Table 3.3. The only slight differences being the ℓ^1 error and

Table 3.4: CPU runtime of the second-order scheme with stiffly-accurate DIRK2. We use the rank-2 initial condition (3.3.2), initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 0.05$.

Diagonalized scheme

N	Initialization				Time-stepping				Total CPU runtime		ℓ^1 error
	SVD		Diagonalization		First time-step		Per time-step				
40	3.35E-02s	0.00	1.44E-03s	0.00	2.60E-02s	0.00	3.21E-03s	0.00	1.25E-01s	0.00	9.55E-07
80	7.50E-03s	-2.16	3.70E-03s	1.37	8.00E-03s	-1.70	3.36E-03s	0.07	8.65E-02s	-0.53	1.14E-06
160	1.44E-02s	0.95	2.86E-03s	-0.37	2.76E-02s	1.79	4.58E-03s	0.44	1.37E-01s	0.66	1.19E-06
320	1.39E-02s	-0.05	9.89E-03s	1.79	7.54E-02s	1.45	7.17E-03s	0.65	2.43E-01s	0.83	1.19E-06
640	4.59E-02s	1.72	4.22E-02s	2.09	4.17E-01s	2.47	2.08E-02s	1.54	9.21E-01s	1.92	1.19E-06
1280	7.93E-01s	4.11	1.62E-01s	1.94	1.55E+00s	1.90	7.05E-02s	1.76	3.92E+00s	2.09	1.17E-06
2560	8.05E+00s	3.34	9.44E-01s	2.54	9.33E+00s	2.59	2.65E-01s	1.91	2.36E+01s	2.59	1.16E-06

Non-diagonalized scheme

N	Initialization		Time-stepping				Total CPU runtime		ℓ^1 error
	SVD		First time-step		Per time-step				
40	6.89E-02s	0.00	2.40E-02s	0.00	4.39E-03s	0.00	1.81E-01s	0.00	9.55E-07
80	7.20E-03s	-3.26	1.20E-02s	-1.00	5.05E-03s	0.20	1.20E-01s	-0.59	1.14E-06
160	1.61E-02s	1.16	2.64E-02s	1.13	7.74E-03s	0.62	1.97E-01s	0.72	1.18E-06
320	1.32E-02s	-0.28	7.91E-02s	1.58	1.71E-02s	1.15	4.35E-01s	1.14	1.19E-06
640	5.27E-02s	2.00	4.14E-01s	2.39	7.14E-02s	2.06	1.90E+00s	2.12	1.18E-06
1280	6.57E-01s	3.64	1.96E+00s	2.24	3.64E-01s	2.35	9.90E+00s	2.38	1.16E-06
2560	8.06E+00s	3.62	1.07E+01s	2.45	1.99E+00s	2.45	5.86E+01s	2.57	1.14E-06

the computational complexity of the total CPU runtime for the diagonalized variant. The computational complexity of the diagonalized variant is closer to quadratic than that in Table 3.3 since the CPU runtime from the initialization and first time-step now make up a smaller proportion of the overall runtime.

The non-diagonalized variant demonstrates cubic computational complexity, whereas the benchmark ADI method enjoys quadratic computational complexity. However, the results in Tables 3.2 and 3.4 suggest that a very fine mesh is required for the ADI method to outperform the non-diagonalized variant in terms of total CPU runtime. Even for the finest mesh that we tested, 2560×2560, the ADI method took roughly three times as long to run as the non-diagonalized variant did.

Example 3.4. (Second-order scheme with Crank-Nicolson)

Since all three initial conditions had similar quantitative results, we only show

the CPU runtime for initial condition (3.3.2). We compare the CPU runtimes using the diagonalized and non-diagonalized variants of the second-order scheme with Crank-Nicolson in Table 3.5. The time-stepping size is fixed at $\Delta t = 0.05$ so that the computational complexity can be observed under mesh refinement. Since the solution is at final time $T_f = 1$, there are a total of 20 time-steps. The results in Table 3.5 are similar to the results in Table 3.4. As such, we refer the reader to Example 3.8 for interpretation of the results.

Table 3.5: CPU runtime of the second-order scheme with Crank-Nicolson. We use the rank-2 initial condition (3.3.2), initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 0.05$.

Diagonalized scheme											
N	Initialization				Time-stepping				Total CPU runtime		ℓ^1 error
	SVD		Diagonalization		First time-step		Per time-step				
40	6.18E-02s	0.00	3.67E-03s	0.00	2.01E-02s	0.00	3.97E-03s	0.00	1.65E-01s	0.00	4.13E-04
80	5.47E-03s	-3.50	3.05E-03s	-0.27	7.02E-03s	-1.52	5.80E-03s	0.55	1.32E-01s	-0.33	4.93E-04
160	1.42E-02s	1.38	2.95E-03s	-0.05	2.73E-02s	1.96	9.50E-03s	0.71	2.34E-01s	0.83	5.21E-04
320	1.56E-02s	0.14	1.07E-02s	1.86	7.14E-02s	1.39	1.35E-02s	0.50	3.67E-01s	0.65	5.27E-04
640	4.02E-02s	1.36	4.22E-02s	1.99	3.54E-01s	2.31	2.71E-02s	1.01	9.79E-01s	1.42	5.28E-04
1280	6.10E-01s	3.92	2.07E-01s	2.29	1.61E+00s	2.18	8.32E-02s	1.62	4.09E+00s	2.06	5.29E-04
2560	8.09E+00s	3.73	9.35E-01s	2.18	9.52E+00s	2.57	3.19E-01s	1.94	2.49E+01s	2.61	5.29E-04
Non-diagonalized scheme											
N	Initialization				Time-stepping				Total CPU runtime		ℓ^1 error
	SVD				First time-step		Per time-step				
40	3.41E-02s		0.00		2.37E-02s	0.00	5.70E-03s	0.00	1.72E-01s	0.00	4.13E-04
80	6.14E-03s		-2.48		1.29E-02s	-0.88	7.53E-03s	0.40	1.70E-01s	-0.02	4.93E-04
160	1.71E-02s		1.48		3.22E-02s	1.32	1.27E-02s	0.75	3.03E-01s	0.84	5.21E-04
320	1.71E-02s		-0.00		8.72E-02s	1.44	2.09E-02s	0.72	5.21E-01s	0.78	5.27E-04
640	5.75E-02s		1.75		4.14E-01s	2.25	7.94E-02s	1.93	2.06E+00s	1.98	5.28E-04
1280	6.35E-01s		3.47		1.99E+00s	2.27	3.73E-01s	2.23	1.01E+01s	2.29	5.29E-04
2560	8.03E+00s		3.66		1.11E+01s	2.48	2.05E+00s	2.46	6.02E+01s	2.57	5.29E-04

Example 3.5. (Second-order scheme with BDF2)

Since all three initial conditions had similar quantitative results, we only show the CPU runtime for initial condition (3.3.2). We compare the CPU runtimes using the diagonalized and non-diagonalized variants of the second-order scheme with BDF2 in Table 3.6. The time-stepping size is fixed at $\Delta t = 0.05$ so that the computational complexity can be observed under mesh refinement. Since the solution is at final time

$T_f = 1$, there are a total of 20 time-steps. The results in Table 3.6 are similar to the results in Tables 3.4 and 3.5. As such, we refer the reader to Example 3.8 for interpretation of the results.

Table 3.6: CPU runtime of the second-order scheme with BDF2. We use the rank-2 initial condition (3.3.2), initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 0.05$.

Diagonalized scheme											
N	Initialization				Time-stepping				Total CPU runtime		ℓ^1 error
	SVD		Diagonalization		First time-step		Per time-step				
40	2.95E-02s	0.00	3.54E-03s	0.00	2.23E-02s	0.00	3.83E-03s	0.00	1.36E-01s	0.00	1.09E-05
80	8.71E-03s	-1.76	4.54E-03s	0.36	1.43E-02s	-0.64	3.61E-03s	-0.09	1.03E-01s	-0.40	1.10E-05
160	1.62E-02s	0.90	3.60E-03s	-0.33	3.95E-02s	1.46	4.75E-03s	0.40	1.59E-01s	0.62	1.11E-05
320	1.65E-02s	0.03	1.50E-02s	2.06	8.09E-02s	1.04	1.12E-02s	1.24	3.48E-01s	1.13	1.11E-05
640	4.63E-02s	1.49	4.98E-02s	1.73	3.92E-01s	2.28	3.04E-02s	1.44	1.13E+00s	1.70	1.11E-05
1280	6.44E-01s	3.80	2.11E-01s	2.08	1.82E+00s	2.21	1.14E-01s	1.91	5.07E+00s	2.17	1.11E-05
2560	8.06E+00s	3.65	9.97E-01s	2.24	9.82E+00s	2.44	5.20E-01s	2.18	2.98E+01s	2.55	1.11E-05
Non-diagonalized scheme											
N	Initialization				Time-stepping				Total CPU runtime		ℓ^1 error
	SVD				First time-step		Per time-step				
40	3.57E-02s		0.00		2.59E-02s	0.00	5.55E-03s	0.00	1.78E-01s	0.00	1.09E-05
80	8.21E-03s		-2.12		1.66E-02s	-0.64	4.35E-03s	-0.35	1.16E-01s	-0.62	1.10E-05
160	1.57E-02s		0.94		4.26E-02s	1.36	9.22E-03s	1.08	2.52E-01s	1.12	1.11E-05
320	1.82E-02s		0.21		8.72E-02s	1.03	2.02E-02s	1.13	5.29E-01s	1.07	1.11E-05
640	4.43E-02s		1.28		4.25E-01s	2.29	7.67E-02s	1.93	2.08E+00s	1.97	1.11E-05
1280	6.28E-01s		3.83		1.93E+00s	2.18	3.95E-01s	2.36	1.08E+01s	2.38	1.11E-05
2560	8.37E+00s		3.74		1.32E+01s	2.77	2.48E+00s	2.65	7.36E+01s	2.76	1.11E-05

3.3.2 Convergence analysis

In this subsection, we demonstrate the spatial and temporal convergence of the first- and second-order schemes. To observe the temporal convergence, we fix the spatial mesh and vary the time-stepping size

$$\Delta t = CFL\Delta x, \tag{3.3.7}$$

where $CFL > 0$ is the CFL number (up to some scaling). Unless otherwise stated, the results in this subsection use tolerance $\epsilon = 1.0E-10$ and initial rank $r^0 = \text{ceil}(N/3)$. Second-order centered differences are used to approximate the spatial derivatives.

Example 3.6. (A benchmark method)

To serve as a benchmark comparison, we first show the convergence results for the classical ADI method [59, 140] that solves for the full-rank solution. The method is formally second-order in both space and time. As seen in Table 3.7, we observe the expected second-order convergence under spatial mesh refinement using time-stepping size $\Delta t = 3\Delta x$ up to final time $T_f = 1$. Figure 3.6 shows the expected second-order temporal convergence, for which we use mesh 320×320 and CFL numbers varying from 0.1 to 20. Notice that the rank two initial condition (3.3.2) leads to ringing behavior for larger time-stepping sizes due to the Crank-Nicolson method, as discussed in Section 3.1.3. This is because initial condition (3.3.2) excites several modes. The proposed low-rank scheme also experiences this ringing behavior when Crank-Nicolson method is used for the time integration.

Table 3.7: Convergence study with spatial mesh refinement of the second-order ADI method. Final time $T_f = 1$ and time-stepping size $\Delta t = 3\Delta x$.

N	Initial condition (3.3.2)			Initial condition (3.3.3)			Initial condition (3.3.4)		
	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime
40	9.88E-05	0.00	1.09E-01s	6.50E-05	0.00	7.81E-02s	6.25E-05	0.00	6.25E-02s
80	6.78E-06	3.87	7.81E-02s	1.64E-05	1.98	1.56E-01s	1.58E-05	1.98	1.09E-01s
160	7.07E-08	6.58	6.88E-01s	4.15E-06	1.98	5.94E-01s	3.95E-06	2.00	7.50E-01s
320	1.74E-08	2.02	8.19E+00s	1.04E-06	2.00	7.98E+00s	9.87E-07	2.00	7.98E+00s
640	4.37E-09	2.00	9.28E+01s	2.61E-07	2.00	9.01E+01s	2.46E-07	2.00	9.17E+01s

Example 3.7. (First-order scheme with backward Euler)

As seen in Table 3.3, the ℓ^1 error is the nearly indistinguishable for the diagonalized and non-diagonalized variants. Since both variants produce the same error, we test for convergence only using the diagonalized variant. Table 3.8 shows the convergence under spatial mesh refinement using time-stepping size $\Delta t = 3\Delta x$. As expected with $\Delta t = 3\Delta x$, the temporal error from backward Euler dominates the error as indicated by the first-order convergence. The temporal error from backward Euler dominates the error even for small time-stepping sizes as seen in Figure 3.7. Figure 3.7 shows the expected first-order temporal convergence, for which we use fixed mesh 320×320 and

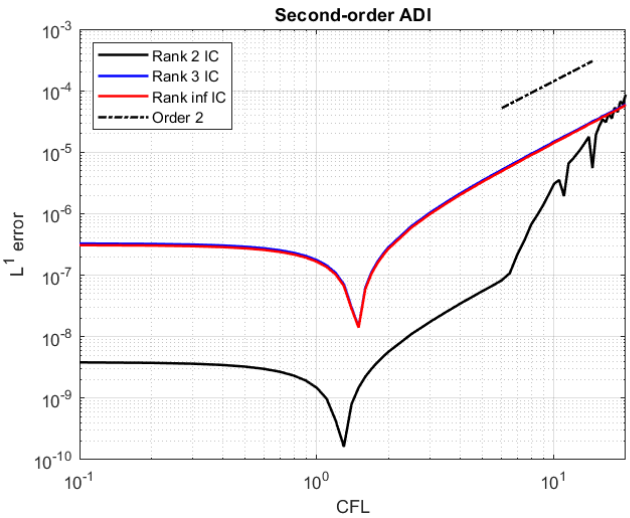


Figure 3.6: Error plot of second-order ADI method with mesh 320×320. The errors are shown for initial conditions (3.3.2)-(3.3.4).

CFL numbers varying from 0.1 to 20. The ℓ^1 errors are larger than those of the full-rank ADI method due to the time-stepping error from the first-order versus second-order time integrators.

Table 3.8: Convergence study with spatial mesh refinement of the first-order scheme with backward Euler (with diagonalization). Final time $T_f = 1$, initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 3\Delta x$.

N	Initial condition (3.3.2)			Initial condition (3.3.3)			Initial condition (3.3.4)		
	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime
40	1.03E-04	0.00	7.58E-02s	5.65E-03	0.00	5.80E-02s	5.32E-03	0.00	7.11E-02s
80	4.82E-05	1.11	3.94E-02s	2.79E-03	1.04	2.61E-02s	2.62E-03	1.04	7.80E-02s
160	2.32E-05	1.07	8.26E-02s	1.38E-03	1.02	8.18E-02s	1.30E-03	1.02	6.14E-02s
320	1.13E-05	1.04	2.07E-01s	6.87E-04	1.01	1.92E-01s	6.47E-04	1.01	2.23E-01s
640	5.61E-06	1.02	1.27E+00s	3.43E-04	1.01	1.18E+00s	3.23E-04	1.01	1.38E+00s

Example 3.8. (Second-order scheme with stiffly-accurate DIRK2)

As seen in Table 3.4, the ℓ^1 error is the nearly indistinguishable for the diagonalized and non-diagonalized variants. Since both variants produce the same error, we test for convergence only using the diagonalized variant. Table 3.9 shows the convergence under spatial mesh refinement using time-stepping size $\Delta t = 3\Delta x$. As expected with

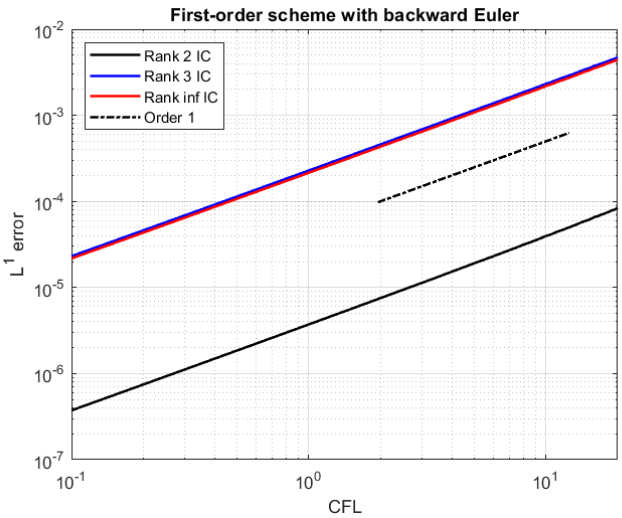


Figure 3.7: Error plot of first-order scheme using backward Euler with mesh 320×320 and tolerance $\epsilon = 1.0E-10$. The errors are shown for initial conditions (3.3.2)-(3.3.4).

Table 3.9: Convergence study with spatial mesh refinement of the second-order scheme with stiffly-accurate DIRK2 (with diagonalization). Final time $T_f = 1$, initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 3\Delta x$.

N	Initial condition (3.3.2)			Initial condition (3.3.3)			Initial condition (3.3.4)		
	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime
40	2.42E-06	0.00	1.11E-01s	9.74E-05	0.00	1.04E-01s	9.16E-05	0.00	1.12E-01s
80	6.02E-07	2.04	9.04E-02s	2.42E-05	2.04	4.81E-02s	2.28E-05	2.04	5.03E-02s
160	1.51E-07	2.02	2.00E-01s	6.08E-06	2.01	1.45E-01s	5.72E-06	2.01	1.41E-01s
320	3.76E-08	2.01	6.27E-01s	1.52E-06	2.01	4.88E-01s	1.43E-06	2.01	5.45E-01s
640	9.41E-09	2.00	3.74E+00s	3.80E-07	2.00	3.53E+00s	3.57E-07	2.00	3.78E+00s

$\Delta t = 3\Delta x$, the we observe second-order convergence. Figure 3.8 shows the expected second-order temporal convergence, for which we use fixed mesh 320×320 and CFL numbers varying from 0.1 to 20. The temporal error from DIRK2 starts to dominate the error around $CFL = 2$, as seen in Figure 3.8. The ℓ^1 errors are comparable to those of the full-rank ADI method.

Example 3.9. (Second-order scheme with Crank-Nicolson)

As seen in Table 3.5, the ℓ^1 error is the nearly indistinguishable for the diagonalized and non-diagonalized variants. Since both variants produce the same error, we test for convergence only using the diagonalized variant. Table 3.10 shows the

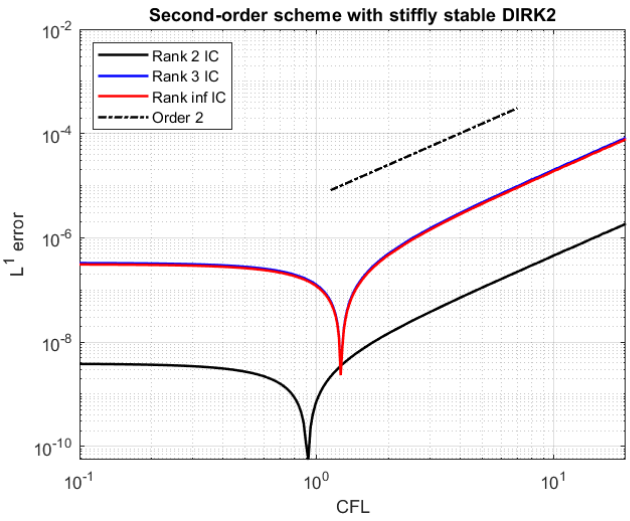


Figure 3.8: Error plot of second-order scheme using stiffly-accurate DIRK2 with mesh 320×320 and tolerance $\epsilon = 1.0E - 10$. The errors are shown for initial conditions (3.3.2)-(3.3.4).

convergence under spatial mesh refinement using time-stepping size $\Delta t = 3\Delta x$. As expected with $\Delta t = 3\Delta x$, the we observe second-order convergence. Figure 3.9 shows the expected second-order temporal convergence, for which we use fixed mesh 320×320 and CFL numbers varying from 0.1 to 20. The temporal error from Crank-Nicolson starts to dominate the error around $CFL = 1$, as seen in Figure 3.9. The ℓ^1 errors are comparable to those of the full-rank ADI method.

The Crank-Nicolson method experiences ringing behavior for larger time-stepping sizes and initial conditions/solutions that excite several modes, as discussed in Section 3.1.3. We show later in Section 3.3.3 that the rank of the solution to initial condition (3.3.2) is larger than the other two initial conditions. This higher rank is due to the fact that Gaussian/Maxwellian distributions excite several modes, hence why only the results from initial condition (3.3.2) experience divergent behavior in the error for larger time-stepping sizes. As seen in Figure 3.9, the amplification factor (see Section 3.1.3) becomes negative for $CFL > 5$ when the spatial mesh is fixed at 320×320. Since many functions in physical applications excite several modes, the second-order scheme with Crank-Nicolson is not the best choice, especially if larger time-stepping

sizes are desired.

Table 3.10: Convergence study with spatial mesh refinement of the second-order scheme with Crank-Nicolson (with diagonalization). Final time $T_f = 1$, initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 3\Delta x$.

N	Initial condition (3.3.2)			Initial condition (3.3.3)			Initial condition (3.3.4)		
	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime
40	1.30E-03	0.00	1.25E-01s	2.17E-04	0.00	1.07E-01s	2.09E-04	0.00	1.00E-01s
80	1.21E-04	3.49	1.58E-01s	5.48E-05	2.02	5.10E-02s	5.28E-05	2.02	8.08E-02s
160	7.63E-07	7.38	4.86E-01s	1.38E-05	2.00	1.32E-01s	1.31E-05	2.02	2.13E-01s
320	8.13E-08	3.25	1.48E+00s	3.46E-06	2.01	5.10E-01s	3.29E-06	2.01	7.20E-01s
640	2.04E-08	2.00	4.46E+00s	8.68E-07	2.00	3.63E+00s	8.20E-07	2.01	3.91E+00s

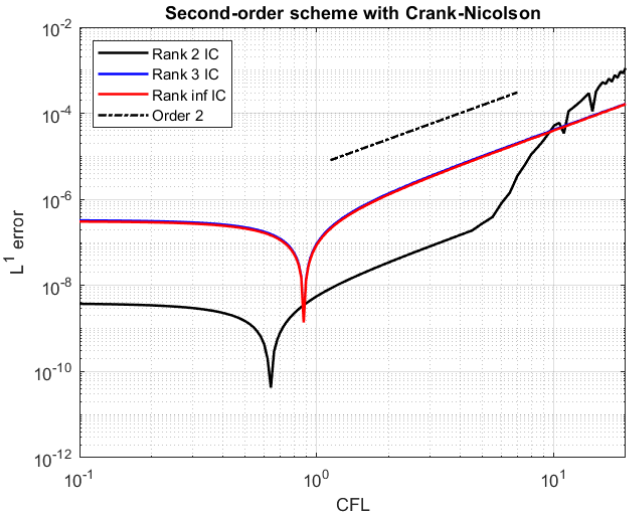


Figure 3.9: Error plot of second-order scheme using Crank-Nicolson with mesh 320×320 and tolerance $\epsilon = 1.0E-10$. The errors are shown for initial conditions (3.3.2)-(3.3.4).

Example 3.10. (Second-order scheme with BDF2)

As seen in Table 3.6, the ℓ^1 error is the nearly indistinguishable for the diagonalized and non-diagonalized variants. Since both variants produce the same error, we test for convergence only using the diagonalized variant. Table 3.11 shows the convergence under spatial mesh refinement using time-stepping size $\Delta t = 3\Delta x$. As expected with $\Delta t = 3\Delta x$, the we observe second-order convergence. Figure 3.10 shows the expected second-order temporal convergence, for which we use fixed mesh 320×320 and CFL

Table 3.11: Convergence study with spatial mesh refinement of the second-order scheme with BDF2 (with diagonalization). Final time $T_f = 1$, initial rank $r^0 = \text{ceil}(N/3)$, tolerance $\epsilon = 1.0E-10$, and time-stepping size $\Delta t = 3\Delta x$.

N	Initial condition (3.3.2)			Initial condition (3.3.3)			Initial condition (3.3.4)		
	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime	ℓ^1 error	ℓ^1 order	Runtime
40	2.60E-05	0.00	1.55E-01s	1.06E-03	0.00	8.33E-02s	9.96E-04	0.00	1.41E-01s
80	5.83E-06	2.20	1.30E-01s	2.47E-04	2.14	5.75E-02s	2.33E-04	2.14	1.13E-01s
160	1.41E-06	2.06	2.35E-01s	6.10E-05	2.04	1.48E-01s	5.74E-05	2.04	1.83E-01s
320	3.44E-07	2.04	7.34E-01s	1.50E-05	2.03	5.12E-01s	1.41E-05	2.03	5.75E-01s
640	8.54E-08	2.02	3.73E+00s	3.75E-06	2.01	3.41E+00s	3.52E-06	2.01	3.72E+00s

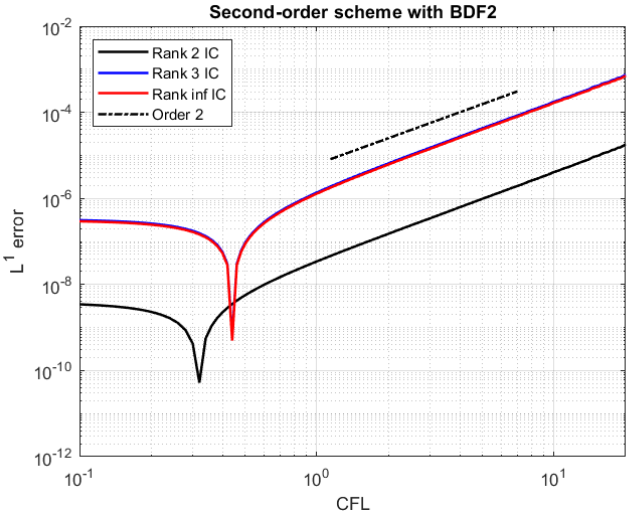


Figure 3.10: Error plot of second-order scheme using BDF2 with mesh 320×320 and tolerance $\epsilon = 1.0E-10$. The errors are shown for initial conditions (3.3.2)-(3.3.4).

numbers varying from 0.1 to 20. The temporal error from BDF2 starts to dominate the error around $CFL = 0.6$, as seen in Figure 3.10. The ℓ^1 errors are comparable to those of the full-rank ADI method.

3.3.3 Rank evolution

In this subsection, we present and analyze the rank evolution of the solution. We compare the results using the two time-stepping sizes used in this section: $\Delta t = 0.5$ and $\Delta t = 3\Delta x$. Since the results of the non-diagonalized and diagonalized variants of the proposed scheme produce nearly identical results, we only show results using the

diagonalized variant. Since the only difference between the two variants is the diagonalization of the real symmetric tridiagonal matrix $\mathbf{I} - \Delta t \mathbf{D}^x$ (and similarly for y), it is not surprising that the results are identical. However, this is most likely because the matrix being diagonalized has nice structure and is not *too large*. More complicated differential operators in the future work, such as convection-diffusion, might cause numerical inaccuracies in the diagonalization, in which case the non-diagonalized variant will be the safer choice.

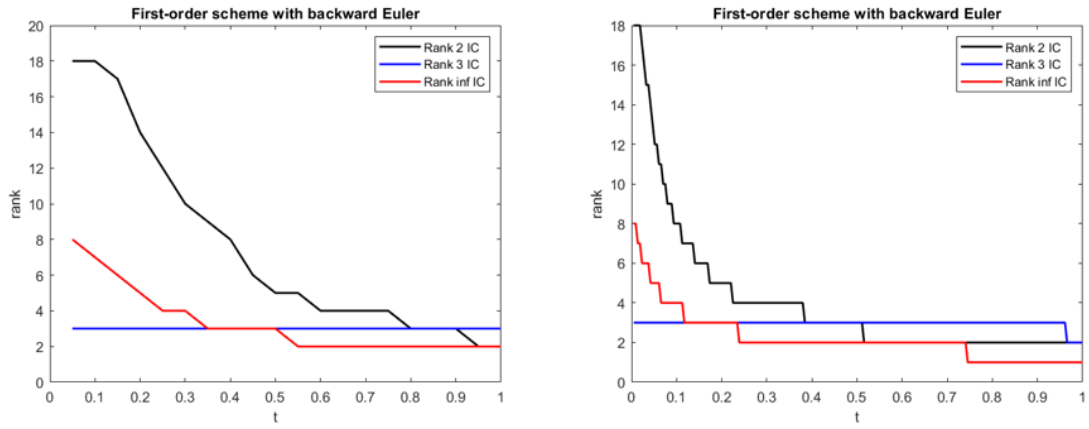


Figure 3.11: The rank evolution using backward Euler with mesh 640×640 and tolerance $\epsilon = 1.0E - 10$. The rank is shown for all three initial conditions (3.3.2)-(3.3.4). (left) Time-stepping size $\Delta t = 0.05$; (right) Time-stepping size $\Delta t = 3\Delta x$.

As seen in Figures 3.11-3.14, the solution corresponding to initial condition (3.3.2) has a higher rank than the solutions corresponding to initial conditions (3.3.3), (3.3.4) (for short times). This is because the initial condition (3.3.2) starts "further" from its equilibrium solution. Whereas, initial condition (3.3.3) is exactly the first three Fourier modes and thus starts close in structure to its equilibrium solution. And, initial condition (3.3.4) is "far" away from its equilibrium solution due to the structure of Runge's function, yet not too far since the numerator is the first Fourier mode.

Furthermore, as seen in Figures 3.11-3.14, the ranks for the solutions to all three initial conditions decay faster for $\Delta t = 3\Delta x$ than for $\Delta t = 0.05 > 3\Delta x$. This is due to the approximate bases not being rich enough to effectively capture the fast diffusion dynamics over time-steps that are too large. In other words, the larger the

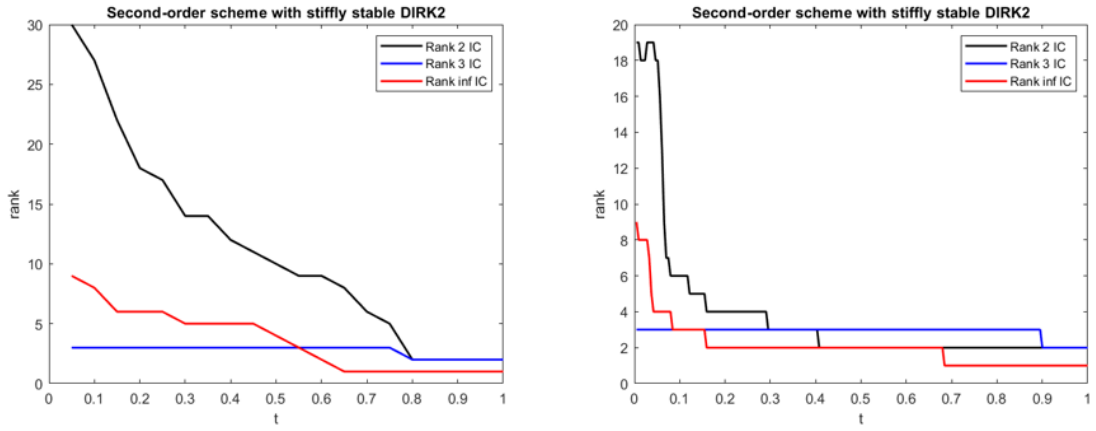


Figure 3.12: The rank evolution using stiffly-accurate DIRK2 with mesh 640×640 and tolerance $\epsilon = 1.0E-10$. The rank is shown for all three initial conditions (3.3.2)-(3.3.4). (left) Time-stepping size $\Delta t = 0.05$; (right) Time-stepping size $\Delta t = 3\Delta x$.

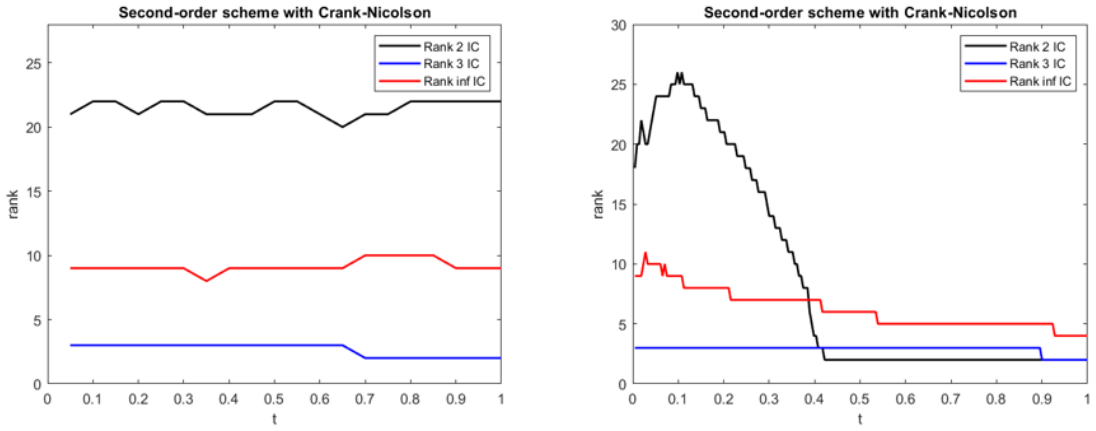


Figure 3.13: The rank evolution using Crank-Nicolson with mesh 640×640 and tolerance $\epsilon = 1.0E-10$. The rank is shown for all three initial conditions (3.3.2)-(3.3.4). (left) Time-stepping size $\Delta t = 0.05$; (right) Time-stepping size $\Delta t = 3\Delta x$.

time-step the more you are asking the approximate bases to predict. Hence, although the diffusion dynamics are captured for large time-stepping sizes to an extent, the rank decay will not be as fast.

Lastly, the rank appears stagnant in Figure 3.13 for the second-order scheme using Crank-Nicolson with a time-stepping size $\Delta t = 0.05$. This is due to the ringing behavior that Crank-Nicolson experiences for time-stepping sizes that are too large, and/or for solutions that excite several modes (see Section 3.1.3). Even for solutions

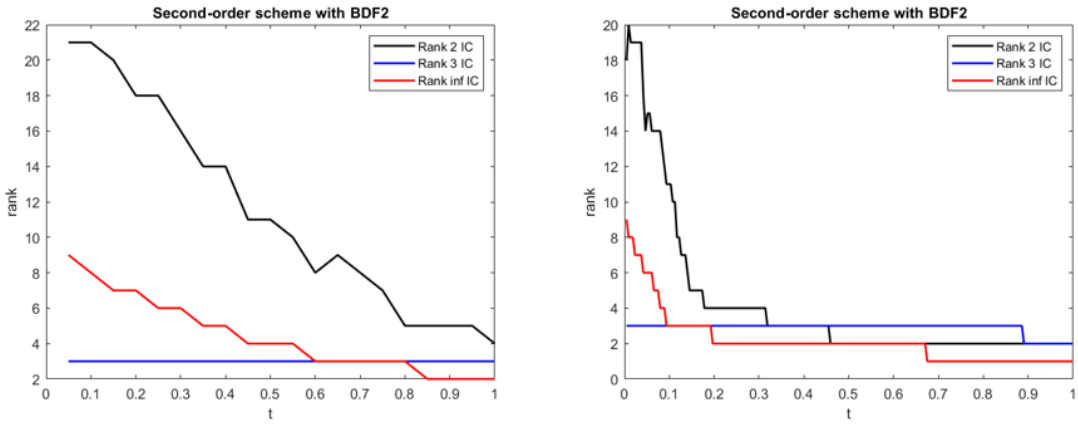


Figure 3.14: The rank evolution using BDF2 with mesh 640×640 and tolerance $\epsilon = 1.0E-10$. The rank is shown for all three initial conditions (3.3.2)-(3.3.4). (left) Time-stepping size $\Delta t = 0.05$; (right) Time-stepping size $\Delta t = 3\Delta x$.

that do not excite too many modes, $\Delta t = 0.05$ is so large that the amplification factor is negative for small wave numbers. Unlike the rank evolutions shown in Figure 3.13 for $\Delta t = 0.05$, the rank evolutions for $\Delta t = 3\Delta x$ decay as they should. This is because a time-stepping size of $\Delta t = 3\Delta x$ with a fixed mesh of 640×640 is small enough that the ringing behavior does not occur.

3.4 Conclusions and follow-up work

A novel implicit low-rank method was presented for solving the diffusion equation. Traditional integrators are utilized in our low-rank setting. Similar to the unconventional method [32], we first update the one-dimensional bases in each direction, and then use the updated bases to update the coefficients of the low-rank solution. Although we do not update the bases in parallel, the proposed scheme is designed to do so. First-order and second-order schemes are presented, and we show that the Crank-Nicolson time discretization produces oscillatory ringing behavior for solutions that activate several modes. Convergence in both space and time is shown, as well as the computational benefits of the low-rank solution.

Ongoing and future work includes constructing rich high-order bases without the need of the K and L steps, extending the proposed scheme to higher-order accuracy

using other spatial discretizations and stiffly-accurate DIRK methods, and formalizing a rigorous analysis of the proposed scheme. In particular, error estimates for the proprosed scheme in the low-rank setting are highly desired. We also want to apply the proposed scheme to more complicated models, e.g., convection-diffusion equations and the Leonard-Bernstein-Fokker-Planck equation.

Chapter 4

A LOW-RANK TENSOR SCHEME WITH STRUCTURE-PRESERVING QUALITIES FOR SOLVING THE 1D2V VLASOV-FOKKER-PLANCK EQUATION

In this chapter, we are concerned with numerically solving the Vlasov-Fokker-Planck equation

$$\frac{\partial f_\alpha}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f_\alpha + \frac{q_\alpha}{m_\alpha} \mathbf{E} \cdot \nabla_{\mathbf{v}} f_\alpha = \sum_{\beta}^{N_s} C_{\alpha\beta}(f_\alpha, f_\beta), \tag{4.0.1}$$

where N_s is the total number of plasma species, $f_\alpha(\mathbf{x}, \mathbf{v}, t)$ is the particle distribution function of species α, and $C_{\alpha\beta}$ is the Fokker-Planck collision operator for species α colliding with species β. Our treatment of equation (4.0.1) assumes one spatial dimension and two velocity dimensions in cylindrical coordinates (1D2V), as well as the linearized form of the Fokker-Planck collision operator (also known as the Leonard-Bernstein-Fokker-Planck operator). Although linearized, the Leonard-Bernstein-Fokker-Planck collision operator is formally nonlinear in the sense that $C_{\alpha\beta}$ depends on macroscopic parameters that are nonlinear functionals of the solution. Furthermore, we assume the collision operator only models Coulomb collisions between a single ion species α with itself ($C_{\alpha\alpha}$) and free electrons ($C_{\alpha e}$). We use a hybrid kinetic-ion and fluid-electron model. That is, we model the ions kinetically with equation (4.0.1) but assume the free electrons behave like a fluid. Coupling equation (4.0.1) with the fluid-electron energy equation yields a nonlinear system of equations with the goal of solving for f_α.

We present a low-rank tensor scheme for solving the Vlasov-Leonard-Bernstein-Fokker-Planck (VLBFP) equation. The low-rank structure of the solution is enforced

using the SVD to remove redundant basis vectors, see Section (3.1.1.1). We observe structure-preserving qualities by discretizing the collision operator with the robust structure-preserving Chang-Cooper (SPCC) method [139]. Although the SPCC method was originally published to accommodate Cartesian coordinates, we extend this method to cylindrical coordinates. Unlike the DLR-inspired method in Chapter 3 that evolves each basis independently, the low-rank tensor method presented here evolves the entire solution in one step. Thus, we update the solution with a different solver that can handle large systems of tensor product structure.

This chapter is organized as follows. Section 4.1 reviews the three technical components used in this scheme: the SPCC method, and the solver for solution updating. Sections 4.2 and 4.3 outline the proposed scheme. Section 4.4 presents numerical results verifying the convergence and structure-preservation qualities of the scheme. And lastly, Section 4.5 has concluding remarks and ongoing work.

4.1 Review of technical components

In this section we review the two key components that help build the proposed scheme. Although we extend the following methods to handle cylindrical coordinates, we start by presenting them in Cartesian coordinates. To stay consistent with the notation later in this chapter, we use $(v_x, v_y) \in \mathbb{R}^2$ to denote Cartesian coordinates in 2V velocity space. First, we review the SPCC discretization used to discretize the Leonard-Bernstein-Fokker-Planck collision operator. Second, we review a solver for large linear systems of tensor-product structure that we use to update the low-rank tensor solution.

4.1.1 A structure-preserving Chang-Cooper (SPCC) discretization in Cartesian coordinates

Structure preservation is highly desired when designing algorithms for solving kinetic models. Physical structures that scientists strive to capture include positivity preservation, equilibrium preservation, relative entropy dissipation, and asymptotic preservation, amongst others. In [139], Pareschi and Zanella recently developed a structure-preserving Chang-Cooper (SPCC) scheme for solving the Fokker-Planck equations of the form

$$\frac{\partial f}{\partial t} = \nabla_{\mathbf{v}} \cdot \left[\mathscr{B}[f](\mathbf{v},t) f(\mathbf{v},t) + \nabla_{\mathbf{v}} \left(D(\mathbf{v}) f(\mathbf{v},t) \right) \right], \tag{4.1.1}$$

where the right-hand side is the Fokker-Planck collision operator. Some authors refer to the quantity $-\left[\mathscr{B}[f](\mathbf{v},t) + \nabla_{\mathbf{v}} D(\mathbf{v})\right]$ as the friction coefficient. By generalizing and extending the popular Chang-Cooper scheme [36], the authors in [139] developed a scheme that they proved is positivity-preserving (under a CFL-type restriction), equilibrium-preserving and relative entropy dissipative. Equilibrium-preservation is enforced by imposing a zero-flux boundary condition and equating the discrete and continuous ratios of the solution on consecutive cells. Relative entropy dissipation is proved by rewriting the Fokker-Planck equation in the non-logarithmic Landau form. The SPCC scheme was demonstrated in Cartesian coordinates in velocity space only for the Fokker-Planck equation, that is, either spatially homogeneous or no Vlasov term. In this subsection, we summarize the one-dimensional case and state, but do not prove, the structure-preserving properties of the SPCC scheme. We refer the reader to [139] for the proofs and more details.

In the one-dimensional case, we can rewrite equation (4.1.1) as

$$\frac{\partial f}{\partial t} = \frac{\partial \mathcal{F}[f](v,t)}{\partial v}, \tag{4.1.2}$$

where the flux function decomposes the right-hand side into its friction and diffusion

terms,

$$\mathcal{F}[f](v,t) = \left(\mathscr{B}[f](v,t) + D'(v)\right) f(v,t) + D(v)\frac{\partial f(v,t)}{\partial v}. \tag{4.1.3}$$

Consider a uniform mesh in velocity v using N grid points,

$$v_{left} = v_1 < v_2 < ... < v_{N-1} < v_N = v_{right},$$

where $\Delta v = v_{i+1} - v_i$, for all i. Letting $v_{i\pm\frac{1}{2}} = v_i \pm \frac{\Delta v}{2}$, the conservative discretization of equation (4.1.2) is

$$\frac{df_i(t)}{dt} = \frac{\mathcal{F}_{i+\frac{1}{2}}(t) - \mathcal{F}_{i-\frac{1}{2}}(t)}{\Delta v}, \tag{4.1.4}$$

where $\mathcal{F}_{i\pm\frac{1}{2}}$ is the numerical flux. The key to the SPCC scheme, and what makes it robust and of Chang-Cooper type, is how we define the numerical flux. Broadly speaking, we let the friction term be expressed as a (non)linear weighted sum of neighboring point values. Let $\zeta[f](v,t) = \mathscr{B}[f](v,t) + D'(v)$ denote the friction coefficient and consider a general numerical flux function of the form

$$\mathcal{F}_{i+\frac{1}{2}} = \tilde{\zeta}_{i+\frac{1}{2}}\tilde{f}_{i+\frac{1}{2}} + D_{i+\frac{1}{2}}\frac{f_{i+1} - f_i}{\Delta v}, \tag{4.1.5a}$$

$$\tilde{f}_{i+\frac{1}{2}} = (1 - \delta_{i+\frac{1}{2}})f_{i+1} + \delta_{i+\frac{1}{2}}f_i. \tag{4.1.5b}$$

The goal is to define $\tilde{\zeta}_{i+\frac{1}{2}}$ and $\delta_{i+\frac{1}{2}}$ in such a way that structure is preserved while maintaining high-order accuracy. After a bit of algebra we can easily express the ratio f_{i+1}/f_i in terms of $\tilde{\zeta}_{i+\frac{1}{2}}$ and $\delta_{i+\frac{1}{2}}$. Imposing the zero-flux boundary condition at the continuous level, $\mathcal{F}[f](v,t) \equiv 0$, dividing equation (4.1.3) by $f(v,t)D(v)$, and integrating over $[v_i, v_{i+1}]$, we get an expression for $f(v_{i+1},t)/f(v_i,t)$. Imposing this analytical flux condition enforces equilibrium preservation. Equating the discrete and continuous ratios $f_{i+1}/f_i = f(v_{i+1},t)/f(v_i,t)$ yields the following expressions for $\tilde{\zeta}_{i+\frac{1}{2}}$ and $\delta_{i+\frac{1}{2}}$.

$$\tilde{\zeta}_{i+\frac{1}{2}} = \frac{D_{i+\frac{1}{2}}}{\Delta v}\int_{v_i}^{v_{i+1}} \frac{\mathscr{B}[f](v,t) + D'(v)}{D(v)}dv, \tag{4.1.6a}$$

$$\delta_{i+\frac{1}{2}} = \frac{1}{\lambda_{i+\frac{1}{2}}} + \frac{1}{1-\exp(\lambda_{i+\frac{1}{2}})}, \tag{4.1.6b}$$

$$\lambda_{i+\frac{1}{2}} = \frac{\Delta v \tilde{\zeta}_{i+\frac{1}{2}}}{D_{i+\frac{1}{2}}}, \tag{4.1.6c}$$

where the integral term can be approximated to arbitrarily high-order accuracy using any suitable quadrature. Equations (4.1.6) are referred to as **structure-preserving Chang-Cooper (SPCC) type schemes**. In the case that the integral term is approximated by the midpoint rule, the function $\mathscr{B}[f](v,t) = \mathscr{B}(w)$ is only a function of velocity, and the diffusion coefficient D is constant, equation (4.1.6) reduces to the **Chang-Cooper** method [26, 36, 131]. Next, we present the main properties from [139] but refer the reader to the paper for more details.

Theorem 4.1 (Mass conservation [139])**.** Let us consider the scheme (4.1.4), (4.1.5) for $i = 1, 2, ..., N$ with zero-flux boundary conditions $\mathcal{F}_{N+\frac{1}{2}} = \mathcal{F}_{-\frac{1}{2}} = 0$. Then

$$\sum_{i=0}^{N} \frac{d}{dt} f_i(t) = 0, \qquad \forall t > 0.$$

Theorem 4.2 (Positivity preservation for the explicit scheme [139])**.** Under the time-step restriction

$$\Delta t \leq \frac{\Delta v^2}{2(M\Delta v + D)}, \qquad M = \max_i |\tilde{\zeta}^n_{i+\frac{1}{2}}|, \qquad D = \max_i D_{i+\frac{1}{2}},$$

the explicit scheme

$$f_i^{n+1} = f_i^n + \Delta t \frac{\mathcal{F}^n_{i+\frac{1}{2}} - \mathcal{F}^n_{i-\frac{1}{2}}}{\Delta v}$$

with flux defined by (4.1.6) preserves non-negativity, i.e., $f_i^{n+1} \geq 0$ if $f_i^n \geq 0$, $i = 1, 2, ..., N$. Moreover, this non-negativity result can be extended to general strong stability-preserving (SSP) Runge-Kutta methods [77] since SSP methods are obtained by considering a convex combination of forward Euler methods.

Theorem 4.3 (Positivity preservation for the semi-implicit scheme [139])**.** Under the

time-step restriction

$$\Delta t < \frac{\Delta v}{2M}, \qquad M = \max_i |\tilde{\zeta}^n_{i+\frac{1}{2}}|,$$

the semi-implicit scheme

$$f_i^{n+1} = f_i^n + \Delta t \frac{\hat{\mathcal{F}}^{n+1}_{i+\frac{1}{2}} - \hat{\mathcal{F}}^{n+1}_{i-\frac{1}{2}}}{\Delta v},$$

where

$$\hat{\mathcal{F}}^{n+1}_{i+\frac{1}{2}} = \tilde{\zeta}^n_{i+\frac{1}{2}} \left[(1 - \delta^n_{i+\frac{1}{2}}) f^{n+1}_{i+1} + \delta^n_{i+\frac{1}{2}} f_i^{n+1} \right] + D_{i+\frac{1}{2}} \frac{f^{n+1}_{i+1} - f_i^{n+1}}{\Delta v},$$

preserves non-negativity, i.e., $f_i^{n+1} \geq 0$ if $f_i^n \geq 0$, $i = 1, 2, ..., N$. Moreover, this non-negativity result can be extended to higher-order semi-implicit schemes following [20].

Theorem 4.4 (Relative entropy dissipation [139])**.** Consider $\mathscr{B}[f](v,t) = v - u$, where $-1 < u < 1$ is a given constant. The numerical flux (4.1.5) defined by (4.1.6) satisfies the discrete entropy dissipation

$$\frac{d}{dt}\mathcal{H}_\Delta(f, f^\infty) = -\mathcal{I}_\Delta(f, f^\infty) \leq 0,$$

where

$$\mathcal{H}_\Delta(f, f^\infty) = \Delta v \sum_{i=0}^{N} f_i \log\left(\frac{f_i}{f_i^\infty}\right)$$

and $\mathcal{I}_\Delta$ is the positive discrete dissipation function

$$\mathcal{I}_\Delta(f, f^\infty) = \sum_{i=0}^{N} \left[\log\left(\frac{f_{i+1}}{f^\infty_{i+1}}\right) - \log\left(\frac{f_i}{f_i^\infty}\right) \right] \cdot \left(\frac{f_{i+1}}{f^\infty_{i+1}} - \frac{f_i}{f_i^\infty} \right) \hat{f}^\infty_{i+\frac{1}{2}} D_{i+\frac{1}{2}} \geq 0.$$

Remark 4.1. The entropy function $\mathcal{H}_\Delta$ defined in Theorem 4.4 is known as the Kullback relative entropy. There is a large literature on entropy inequalities and the long-time behavior of kinetic equations, and we refer the reader to [54] for more information.

4.1.2 A matrix exponential-based solver for large linear systems of tensor-product structure

Despite there being a vast literature on how to solve linear systems in the two-dimensional matrix case, such solvers for higher-order low-rank tensors can be more complicated (not to say the two-dimensional case is *easy*). Two methods for solving such systems are the low Kronecker-rank approximations in [79] and the preconditioned tensorized Krylov method in [113]. This subsection focuses on the prior method.

In [79], Grasedyck presents an implicit solver for large linear systems of tensor-product structure. Broadly speaking, the method approximates the solution within some error by a sum of vectors in tensor-product structure. Each vector involves the computation of a matrix exponential which although inconvenient, is tolerable in the low-rank setting since the method easily accommodates higher dimensions. The theoretical details of the method are more involved and left to the paper [79]. For the sake of this dissertation, we only present the approximate solution and its error estimates. We consider a linear system in d dimensions

$$Ax = b, \tag{4.1.7}$$

where A and b have a special structure

$$A = \sum_{i=1}^{d} \hat{A}_i, \quad \hat{A}_i = \underbrace{I \otimes ... \otimes I}_{i-1 \text{ times}} \otimes A_i \otimes \underbrace{I \otimes ... \otimes I}_{d-i \text{ times}}, \quad A_i \in \mathbb{R}^{N \times N}, \tag{4.1.8a}$$

$$b = \bigotimes_{i=1}^{d} b_i, \quad b_i^{(k)} \in \mathbb{R}^N, \quad b_{(j_1,...,j_d)} = \prod_{i=1}^{d} (b_i)_{j_i}, \quad \text{for } j \in \{1, ..., N\}^d, \tag{4.1.8b}$$

where $\otimes$ denotes the (Kronecker) tensor product. Each A_i is the discretization in the i^{th} dimension, and it is assumed the spectrum $\sigma(A)$ is contained in the left complex halfplane. Such operators arise in finite element and finite difference discretizations of partial differential equations, e.g., diffusion equations and convection-diffusion equations. Given $\epsilon > 0$, Grasedyck's algorithm can solve linear systems with structure

(4.1.8) for an approximate solution $\tilde{x}$ such that

$$\|x - \tilde{x}\|_2 \leq \epsilon \|x\|_2 .$$

Leaving its derivation to the paper, the approximate solution to (4.1.7), (4.1.8) is given in the following theorem; the proof can be found in [79].

Theorem 4.5 (Approximate solution to (4.1.7), (4.1.8) [79])**.** Let A be a matrix of tensor structure (4.1.8) with spectrum $\sigma(A)$ contained in the strip $\mathbf{\Omega} \coloneqq -[\lambda_{min}, \lambda_{max}] \oplus i[-\mu, \mu] \subseteq \mathbb{C}_-$. Let b be the tensor vector in (4.1.8). Let $K \in \mathbb{N}$ and (t_j, w_j), $j = -K, ..., K$ be the Stenger quadrature nodes and weights given in Appendix F. Then, the solution to $Ax = b$ can be approximated by

$$\tilde{x} \coloneqq -\sum_{j=-K}^{K} \frac{2w_j}{\lambda_{min}} \bigotimes_{i=1}^{d} \left(\exp\left(\frac{2t_j}{\lambda_{min}} A_i \right) b_i \right) \tag{4.1.9}$$

with error estimate

$$\|x - \tilde{x}\|_2 \leq \frac{C_{st}}{\pi \lambda_{min}} \exp\left(\frac{2\mu\lambda_{min}^{-1} + 1}{\pi} - \pi\sqrt{2K} \right) \oint_\Gamma \left\| (\lambda I - 2A/\lambda_{min})^{-1} \right\| d_\Gamma \lambda \|b\|_2 , \tag{4.1.10}$$

where C_{st} is a constant independent of A_i and K, and Γ denotes the boundary of a rectangle containing $\mathbf{\Omega}$ in the left complex halfplane. Here, λ_{min} and λ_{max} are the smallest and largest eigenvalues of A. These are easily computed since the eigenvalues of A are the sum of the eigenvalues of the A_i,

$$\sigma(A) = \sum_{i=1}^{d} \sigma(A_i) = \left\{ \sum_{i=1}^{d} \lambda_i : \lambda_i \in \sigma(A_i) \right\} .$$

The Stenger quadrature is used in the approximate solution. It is a quadrature originally intended to approximate improper integrals of scalar exponential functions that decay at infinity [160]. Grasedyck extends this quadrature to approximate matrix

exponentials. We provide the Stenger quadrature nodes and weights, as well as the error estimate in Appendix F.

For our purposes, all A_i are assumed diagonalizable, which allows us to use Algorithm 12 from [79] to compute the approximate solution. The author presents two other algorithms using a Taylor series expansion of the matrix exponential and $\mathcal{H}-$matrices [81], respectively. We outline Algorithm 12 from [79] below.

Algorithm 4.1. Algorithm 12 from [79]

Inputs: the diagonalizable matrices A_i; tensor vector $\otimes_{i=1}^{d} b_i$; Stenger quadrature nodes and weights (t_j, w_j), $j = -K, ..., K$.

Outputs: the approximate solution $\tilde{x}$.

1. Compute the eigenvalue decomposition $A_i = T_i D_i T_i^{-1}$, for $i = 1, ..., d$.
2. Set $\hat{b}_i := T_i^{-1} b_i$, for $i = 1, ..., d$.
3. For each $1 \leq i \leq d$ and $-K \leq j \leq K$, compute the vector $\tilde{x}_i^{(j)} := T_i \exp\left(\frac{2t_j}{\lambda_{min}} D_i\right) \hat{b}_i$, where $\lambda_{min} := \sum_{i=1}^{d} \lambda_{min}(A_i)$.
   Note: we used MATLAB's expm function to compute the matrix exponential and found it faster than the available packages in Julia as of 2022.
   The approximate solution is $\tilde{x} := -\sum_{j=-K}^{K} \frac{2w_j}{\lambda_{min}} \bigotimes_{i=1}^{d} \tilde{x}_i^{(j)}$.

Remark 4.2. The method described was for the case of a rank-1 righthand side. In fact, the righthand side is quite often a rank-m tensor (e.g., the rank-m solution at a previous time-step),

$$b = \sum_{k=1}^{m} b^{(k)},$$

where each $b^{(k)}$ has the tensor-product structure in (4.1.8). In this case, we simply

apply the inverse operator to each $b^{(k)}$ and the approximate solution is instead

$$\tilde{x} = -\sum_{k=1}^{m} \left(\sum_{j=-K}^{K} \frac{2w_j}{\lambda_{min}} \bigotimes_{i=1}^{2} \left(\exp\left(\frac{2t_j}{\lambda_{min}} A_i \right) b_i^{(k)} \right) \right). \tag{4.1.11}$$

The approximate solution can easily be computed (assuming all the A_i are diagonalizable) by applying Algorithm 4.1 to each $b^{(k)}$.

4.2 Discretizing the 1D2V Vlasov-Leonard-Bernstein-Fokker-Planck equation

The goal of this chapter is to solve a hybrid kinetic-ion and fluid-electron model for a single species ion distribution function $f_\alpha(\mathbf{x}, \mathbf{v}, t)$, where α denotes the ion species (e.g., hydrogen). The Vlasov-Fokker-Planck with the Leonard-Bernstein form of the Fokker-Planck collision operator is used for the kinetic model, and it is nonlinearly coupled with the fluid-electron energy equation. We assume one spatial dimension x and two velocity dimensions in cylindrical coordinates $(v_{||}, v_\perp)$ with azimuthal symmetry. The parallel velocity direction coincides with the spatial dimension, as seen in Figure 4.1.

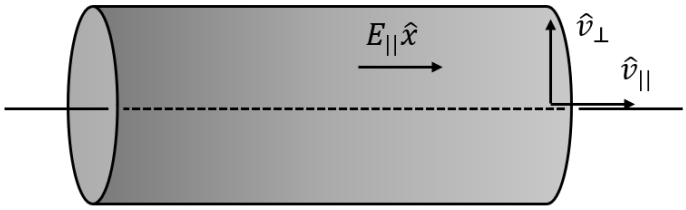


Figure 4.1: The phase space domain of the 1D2V model. The direction of the parallel velocity component coincides with the spatial direction.

The *nondimensional* model and its macroscopic parameters and kinetic fluxes are presented. The nondimensional Vlasov-Leonard-Bernstein-Fokker-Planck (VLBFP) model in 2V cylindrical coordinates is [162]

$$\frac{\partial f_\alpha}{\partial t} + v_{||} \frac{\partial f_\alpha}{\partial x} + \frac{q_\alpha}{m_\alpha} E_{||} \frac{\partial f_\alpha}{\partial v_{||}} = C_{\alpha\alpha} + C_{\alpha e}, \tag{4.2.1a}$$

$$C_{\alpha\alpha} = \nu_{\alpha\alpha}\nabla_{\mathbf{v}} \cdot \left(\frac{T_\alpha}{m_\alpha}\nabla_{\mathbf{v}} f_\alpha + (\mathbf{v} - \mathbf{u}_\alpha) f_\alpha \right), \tag{4.2.1b}$$

$$C_{\alpha e} = \nu_{\alpha e}\nabla_{\mathbf{v}} \cdot \left(\frac{T_e}{m_\alpha}\nabla_{\mathbf{v}} f_\alpha + (\mathbf{v} - \mathbf{u}_e) f_\alpha \right), \tag{4.2.1c}$$

where f_α is the distribution function for the single ion species α, and the charge, mass, temperature, bulk velocity and collision frequencies for the ions and electrons are respectively denoted by q, m, T, $\mathbf{u}$, and ν. The nondimensional fluid electron energy equation is [162]

$$\frac{3}{2}\frac{\partial p_e}{\partial t} + \frac{5}{2}\frac{\partial}{\partial x}\left(u_{e,||} p_e\right) - u_{e,||}\frac{\partial p_e}{\partial x} - \frac{\partial}{\partial x}\left(\kappa_{e,||}\frac{\partial T_e}{\partial x}\right) = W_{e\alpha}, \tag{4.2.2a}$$

$$W_{e\alpha} = -\left\langle \frac{m_\alpha |\mathbf{v}|^2}{2}, C_{\alpha e} \right\rangle, \tag{4.2.2b}$$

where $p_e = n_e T_e$ is the electron pressure, $\kappa_{e,||}$ is the electron Braginskii thermal conductivity, and the velocity space L^2 inner product is denoted

$$\langle F(\mathbf{v}), G(\mathbf{v})\rangle \doteq 2\pi \int_{-\infty}^{\infty}\int_{0}^{\infty} F(\mathbf{v})G(\mathbf{v}) v_\perp dv_\perp dv_{||}. \tag{4.2.3}$$

The ion mass, number density, temperature and charge are scaled to unity. To enforce conservation of the moment system at the spatio-temporal discrete level, we assume quasi-neutrality $n = n_\alpha = n_e$, ambipolarity $\mathbf{u} = \mathbf{u}_\alpha = \mathbf{u}_e$, and that the electric field is determined from Ohm's law $E_{||} = \frac{1}{q_e n_e}\frac{\partial p_e}{\partial x}$; for the applications of interest we assume $q_\alpha = -q_e$. We further assume drift only occurs in the parallel direction, that is, $u_\perp = 0$. The dimensional model and its nondimensionalization, including how we define the reference quantities, can be found in Appendix D. Appendix D uses tildes $\sim$ to denote the dimensionless variables, but the tildes are dropped in this chapter for notational ease. Under these assumptions, the nondimensional collision frequencies and thermal conductivity are given by the following expressions.

$$\nu_{\alpha\alpha} = \frac{n_\alpha}{T_\alpha^{3/2}} \tag{4.2.4a}$$

$$\frac{\nu_{\alpha e}}{\sqrt{2}} = \frac{\sqrt{m_e} n_e}{T_e^{3/2}} \tag{4.2.4b}$$

$$\kappa_{e,||} = \frac{3.2}{\sqrt{2}} \frac{T_e^{5/2}}{\sqrt{m_e}} \tag{4.2.4c}$$

4.2.1 The macroscopic system and kinetic fluxes

Taking the zeroth, first, and second order moments of the distribution function, define the physical quantities

$$n(x,t) \doteq \left\langle f_\alpha(x, v_\perp, v_{||}, t), 1 \right\rangle, \tag{4.2.5a}$$

$$(nu_{||})(x,t) \doteq \left\langle f_\alpha(x, v_\perp, v_{||}, t), v_{||} \right\rangle, \tag{4.2.5b}$$

$$(nU)_\alpha(x,t) \doteq \left\langle f_\alpha(x, v_\perp, v_{||}, t), \frac{v_\perp^2 + v_{||}^2}{2} \right\rangle, \tag{4.2.5c}$$

and the kinetic fluxes

$$S_\alpha(x,t) \doteq \left\langle v_{||} f_\alpha(x, v_\perp, v_{||}, t), v_{||} \right\rangle, \tag{4.2.6a}$$

$$Q_\alpha(x,t) \doteq \left\langle v_{||} f_\alpha(x, v_\perp, v_{||}, t), \frac{v_\perp^2 + v_{||}^2}{2} \right\rangle. \tag{4.2.6b}$$

As shown in Appendix E, if f_α is a Maxwellian distribution function, then the kinetic fluxes reduce to

$$S_\alpha(x,t) = \frac{nT_\alpha}{m_\alpha} + nu_{||}^2, \tag{4.2.7a}$$

$$Q_\alpha(x,t) = u_{||} \left((nU)_\alpha + \frac{nT_\alpha}{m_\alpha} \right). \tag{4.2.7b}$$

Here $m_\alpha n$, $m_\alpha n u_{||}$ and $m_\alpha (nU)_\alpha$ are the *local* mass, momentum and ion energy densities. Integrating over the spatial domain $\Omega_{\mathbf{x}}$, the *total* mass, momentum and energy are

$$\int_{\Omega_{\mathbf{x}}} m_\alpha n(x,t) dx, \tag{4.2.8a}$$

$$\int_{\Omega_{\mathbf{x}}} m_\alpha n u_{||}(x,t)dx, \tag{4.2.8b}$$

$$\int_{\Omega_{\mathbf{x}}} m_\alpha \left((nU)_\alpha + \frac{3}{2} n_e T_e \right)(x,t)dx, \tag{4.2.8c}$$

where the total energy consists of both the total ion and electron energies. Taking the first three moments of equations (4.2.1), (4.2.2) and integrating over the spatial domain $\Omega_{\mathbf{x}}$, the macroscopic system is

$$0 = \frac{d}{dt}\int_{\Omega_{\mathbf{x}}} n(x,t)dx + \left[n u_{||} \right]_{\partial\Omega_{\mathbf{x}}}, \tag{4.2.9a}$$

$$0 = \frac{d}{dt}\int_{\Omega_{\mathbf{x}}} (n u_{||})(x,t)dx + \left[S_\alpha - \frac{q_\alpha n T_e}{q_e m_\alpha} \right]_{\partial\Omega_{\mathbf{x}}}, \tag{4.2.9b}$$

$$0 = \frac{d}{dt}\int_{\Omega_{\mathbf{x}}} \left((nU)_\alpha + \frac{3}{2} n_e T_e \right)(x,t)dx + \left[Q_\alpha + \frac{5}{2} u_{||} n T_e - \kappa_{e,||} \frac{\partial T_e}{\partial x} \right]_{\Omega_{\partial\mathbf{x}}}. \tag{4.2.9c}$$

Equations (4.2.9) are derived more carefully in Appendix E. In practice, the spatial temperature gradient at infinity (i.e., $\partial\Omega_{\mathbf{x}}$) is zero. If the boundary terms cancel, the balance equations (4.2.9) reduce to the total mass, momentum and energy being time-invariant.

There is another important relationship for the presented model. Assuming f_α is an arbitrary distribution function, which for our purposes is a reasonable assumption, the ion temperature is given by

$$3 n_\alpha T_\alpha = m_\alpha \langle f_\alpha, |\mathbf{v} - \mathbf{u}|^2 \rangle. \tag{4.2.10}$$

Expanding $|\mathbf{v}|^2 = (v_{||} - u_{||})^2 + u_{||}^2 + v_{||}^2 + 2u_{||}(v_{||} - u_{||})$, it is straightforward to derive the relationship

$$2(nU)_\alpha = \frac{3 n_\alpha T_\alpha}{m_\alpha} + n u_{||}^2. \tag{4.2.11}$$

Equation (4.2.11) is derived in Appendix E. Note that the ion temperature T_α can be expressed in terms of n, $nu_{||}$ and $(nU)_\alpha$. Under the ambipolarity assumption and

equation (4.2.11), the second-order moment of the ion-electron Leonard-Bernstein-Fokker-Planck operator can be expressed more simply as

$$\left\langle \frac{|\mathbf{v}|^2}{2}, C_{\alpha e} \right\rangle_{\mathbf{v}} = \frac{3\nu_{\alpha e} n_\alpha}{m_\alpha} \left(T_e - T_\alpha\right). \tag{4.2.12}$$

Therefore, the fluid-electron energy equation (4.2.2) reduces to

$$\frac{3}{2}\frac{\partial p_e}{\partial t} + \frac{5}{2}\frac{\partial}{\partial x}\left(u_{||} p_e\right) - u_{||}\frac{\partial p_e}{\partial x} - \frac{\partial}{\partial x}\left(\kappa_{e,||}\frac{\partial T_e}{\partial x}\right) = 3\nu_{\alpha e} n_\alpha \left(T_\alpha - T_e\right). \tag{4.2.13}$$

4.2.2 Scheme formulation: discretizing in physical space-time

The spatial domain $[x_{left}, x_{right}]$ is uniformly discretized with N_x nodes,

$$x_{\text{left}} = x_1 < x_2 < ... < x_{N_x} = x_{\text{right}}.$$

The solution at each spatial node is a function of velocity and time, $f_\alpha(v_{||}, v_\perp, t; x = x_i)$. Further discretizing in time, consider N_t uniform time-steps over the time interval $[0, T_f]$,

$$0 = t^0 < t^1 < ... < t^{N_t} = T_f,$$

where each time-step is size Δt. Taking a method-of-lines approach, we will update the solution $f_\alpha(v_{||}, v_\perp, t; x = x_i)$ at each spatial node x_i. We discretize in time using the first-order implicit-explicit (IMEX) Runge-Kutta method, see Section 2.1.2.2. The collision operator and acceleration term are stiff and evolved implicitly; the Vlasov transport term is nonstiff and evolved explicitly. Discretizing equation (4.2.1) in space and time, for each spatial node x_i $(i = 1, 2, ..., N_x)$,

$$f_{\alpha,i}^{k+1} - \Delta t C_{\alpha\alpha,i}^{k+1} - \Delta t C_{\alpha e,i}^{k+1} + \frac{q_\alpha}{m_\alpha}\Delta t E_{||,i}^{k+1}\frac{\partial f_{\alpha,i}^{k+1}}{\partial v_{||}} = f_{\alpha,i}^k - v_{||}\frac{\Delta t}{\Delta x}\left(\hat{f}_{\alpha,i+\frac{1}{2}}^k - \hat{f}_{\alpha,i-\frac{1}{2}}^k\right), \tag{4.2.14}$$

where $f_{\alpha,i}^k$ approximates $f_\alpha(v_{||}, v_\perp; x = x_i, t = t^k)$, and $\hat{f}_{\alpha,i+\frac{1}{2}}^k$ are the numerical fluxes at the cell boundaries whose upwind direction is determined by the sign of $v_{||}$.

The goal is to solve equation (4.2.14) at each spatial node x_i. The solution is known to be low-rank in velocity space; this is a key assumption for the proposed scheme to be computationally advantageous. Following a step-and-truncate strategy similar to Chapter 3, the updated solution at each spatial node will then be truncated using the SVD. Although the scheme is described in greater detail later on, we start by broadly outlining the major steps of the proposed scheme.

Outline of the proposed scheme (for each x_i, $i = 1, 2, ..., N_x$)

0. Discretize the solution in velocity-space, $\mathbf{f}^k_{\alpha,i} \equiv f^k_{\alpha,i}$.

1. Discretize equation (4.2.14) in velocity-space and set up as a linear system.

**Note: the coupling with equation (4.2.2) occurs when discretizing the collision operator and electric field.*

2. (Add basis step). Solve equation (4.2.14) for $\mathbf{f}^{k+1,\star}_{\alpha,i}$, where the $\star$ denotes the full-rank/pre-truncated solution. Updating the solution will increase the rank/number of basis vectors.

3. (Remove basis step). Truncate $\mathbf{f}^{k+1,\star}_{\alpha,i}$ using the SVD to obtain the low-rank solution $\mathbf{f}^{k+1}_{\alpha,i}$. Truncating the updated solution will decrease the rank by removing redundant basis vectors.

4.2.3 Scheme formulation: discretizing in velocity space

The velocity domain in cylindrical coordinates $[v_{||,left}, v_{||,right}] \times [0, v_{\perp,right}]$ is uniformly discretized in each dimension with $N_{||}$ and $N_\perp$ nodes, respectively.

$$v_{||,left} = v_{||,1} < v_{||,2} < ... < v_{||,N_{||}} = v_{||,right}$$

$$\Delta v_\perp/2 = v_{\perp,1} < v_{\perp,2} < ... < v_{\perp,N_\perp} = v_{\perp,right} + \Delta v_\perp/2$$

The perpendicular discretization is shifted by half a cell length $\Delta v_\perp/2$ so that the origin is the left-most cell boundary and not a node. We found that the singularity at the origin due to the Jacobian in the differential equation caused artificial energy to enter the system if the origin was made a node.

The solution is discretized in velocity space in a low-rank format, e.g., the CP format (see Section 3.1.1.3) or the DLR format (see Section 3.1.2.2). Discretizing in velocity space, the solution at spatial node x_i ($i = 1, 2, ..., N_x$) and time t^k ($k = 0, 1, ..., N_t$) is

$$\mathbf{f}_{\alpha,i}^{k,\star} = \sum_{r=1}^{R_i^k} c_{i,r}^k \mathbf{U}_{i,r}^{(1),k} \otimes \mathbf{U}_{i,r}^{(2),k}, \tag{4.2.15}$$

where R_i^k is the rank at node x_i and time t^k, $\star$ denotes the full-rank/pre-truncated solution, $\otimes$ denotes the tensor product, $\{c_{i,r}^k \in \mathbb{R} : r = 1, 2, ..., R_i^k\}$ are the transfer coefficients, and $\{\mathbf{U}_{i,r}^{(1),k} \in \mathbb{R}^{N_{||}} : r = 1, 2, ..., R_i^k\}$ and $\{\mathbf{U}_{i,r}^{(2),k} \in \mathbb{R}^{N_\perp} : r = 1, 2, ..., R_i^k\}$ are the basis vectors in $v_{||}$ and $v_\perp$, respectively. Equation (4.2.15) can also be cast into the form

$$\mathbf{f}_{\alpha,i}^{k,\star} = \sum_{r=1}^{R_i^k} \left(\text{sgn}(c_{i,r}^k) \sqrt{|c_{i,r}^k|} \mathbf{U}_{i,r}^{(1),k} \right) \otimes \left(\sqrt{|c_{i,r}^k|} \mathbf{U}_{i,r}^{(2),k} \right). \tag{4.2.16}$$

Dropping unnecessary superscripts and subscripts for the sake of simplicity, the solution at each spatial node and time-step is

$$\mathbf{f}^\star = \sum_{r=1}^{R} c_r \mathbf{U}_r^{(1)} \otimes \mathbf{U}_r^{(2)}. \tag{4.2.17}$$

One can view this low-rank structure as the DLR framework with diagonal $\mathbf{S}$ (see Section 3.1.2.2); or, one can view it as the CP format where $\otimes$ is the outer product (see Section 3.1.1.3). But unlike the DLR methodology that updates one basis at a time, the proposed scheme updates everything at once. Recall that the low-rank format is not necessarily the same as the (reduced) SVD, although the SVD is a special case of the low-rank format. The basis vectors need not be orthonormal, as we will see after updating the solution. Furthermore, the vectors and transfer coefficients are not necessarily ordered from "most important to least important."

4.2.3.1 Computing the macroscopic quantities

Discretizing the collision operator and electric field on the lefthand side of equation (4.2.14) requires the macroscopic quantities n^{k+1}, $(nu_{||})^{k+1}$, $(nU)_{\alpha}^{k+1}$ and T_e^{k+1} since the lefthand side is evolved implicitly. To solve for $\left[n^{k+1}, (nu_{||})^{k+1}, (nU)^{k+1}, T_e^{k+1}\right]^T$ we need four equations for four unknowns. The system of equations consists of the first three moments of equation (4.2.14) and the semidiscrete form of the fluid-electron energy equation (4.2.13). (This is where the kinetic-ion and fluid-electron equations are coupled). Assuming proper boundary conditions, the first three moments of equation (4.2.14) yield (also see Appendix E)

$$0 = n_i^{k+1} - n_i^k + \frac{\Delta t}{\Delta x}\left(\widehat{nu_{||}}_{i+\frac{1}{2}}^k - \widehat{nu_{||}}_{i-\frac{1}{2}}^k\right), \tag{4.2.18a}$$

$$\begin{aligned} 0 = (nu_{||})_i^{k+1} - (nu_{||})_i^k + \frac{\Delta t}{\Delta x}\left(\hat{S}_{\alpha,i+\frac{1}{2}}^k - \hat{S}_{\alpha,i-\frac{1}{2}}^k\right) \\ - \frac{\Delta t}{2\Delta x}\frac{q_\alpha}{m_\alpha q_e}\left(n_{i+1}^{k+1}T_{e,i+1}^{k+1} - n_{i-1}^{k+1}T_{e,i-1}^{k+1}\right), \end{aligned} \tag{4.2.18b}$$

$$\begin{aligned} 0 = (nU)_{\alpha,i}^{k+1} - (nU)_{\alpha,i}^k + \frac{\Delta t}{\Delta x}\left(\hat{Q}_{\alpha,i+\frac{1}{2}}^k - \hat{Q}_{\alpha,i-\frac{1}{2}}^k\right) \\ - 3\Delta t\frac{\sqrt{2}\sqrt{m_e}}{m_\alpha^2}\frac{(n_i^{k+1})^2}{(T_{e,i}^{k+1})^{3/2}}\left(T_{e,i}^{k+1} - \frac{m_\alpha}{3}\left(2U_{\alpha,i}^{k+1} - (u_{||,i}^{k+1})^2\right)\right) \\ - \frac{\Delta t}{2\Delta x}\frac{q_\alpha}{m_\alpha q_e}u_{||,i}^{k+1}\left(n_{i+1}^{k+1}T_{e,i+1}^{k+1} - n_{i-1}^{k+1}T_{e,i-1}^{k+1}\right), \end{aligned} \tag{4.2.18c}$$

where the pressure term is discretized using a directionally unbiased centered differencing. The kinetic fluxes are computed using the midpoint rule to evaluate equations (4.2.5b), (4.2.6a), and (4.2.6b) with

$$\hat{f}_{\alpha,i+\frac{1}{2}}^k = \begin{cases} f_{\alpha,i}, & \text{if } v_{||} > 0, \\ f_{\alpha,i+1}, & \text{if } v_{||} \leq 0. \end{cases} \tag{4.2.19}$$

We assume Dirichlet boundary conditions in space for the local number density n, drift velocity $\mathbf{u}$, and ion temperature T_α. Using the macroscopic quantities of the initial condition at the boundary, we establish fixed Maxwellian distributions at the

ghost cells, $\mathbf{f}^M_{\alpha,1/2}$ and $\mathbf{f}^M_{\alpha,N_x+\frac{1}{2}}$. Note that the spatial boundary conditions are both Dirichlet and zero-flux, that is, the macroscopic values are fixed and the slope is zero.

For the fourth equation we discretize the fluid-electron energy equation (4.2.13) in space and time to get

$$\begin{aligned} 0 = n_i^{k+1}T_{e,i}^{k+1} - n_i^k T_{e,i}^k &+ \frac{5\Delta t}{3\Delta x}\left(u^k_{||,i+\frac{1}{2}}\widehat{nT}^k_{e,i+\frac{1}{2}} - u^k_{||,i-\frac{1}{2}}\widehat{nT}^k_{e,i-\frac{1}{2}}\right) \\ &- \frac{\Delta t}{3\Delta x}u^{k+1}_{||,i}\left(n^{k+1}_{i+1}T^{k+1}_{e,i+1} - n^{k+1}_{i-1}T^{k+1}_{e,i-1}\right) \\ &- \frac{2\Delta t}{3\Delta x^2}\left(\kappa^{k+1}_{e,||,i+\frac{1}{2}}(T^{k+1}_{e,i+1} - T^{k+1}_{e,i}) - \kappa^{k+1}_{e,||,i-\frac{1}{2}}(T^{k+1}_{e,i} - T^{k+1}_{e,i-1})\right) \\ &- 2\Delta t\frac{\sqrt{2}\sqrt{m_e}}{m_\alpha}\frac{(n_i^{k+1})^2}{(T^{k+1}_{e,i})^{3/2}}\left(\frac{m_\alpha}{3}\left(2U^{k+1}_{\alpha,i} - (u^{k+1}_{||,i})^2\right) - T^{k+1}_{e,i}\right), \end{aligned} \tag{4.2.20}$$

where the drift velocity and thermal conductivity at the cell boundaries are computed via averaging,

$$u_{||,i+\frac{1}{2}} \doteq \frac{u_{||,i} + u_{||,i+1}}{2}, \tag{4.2.21a}$$

$$\kappa_{e,||,i+\frac{1}{2}} \doteq \frac{\kappa_{||,i} + \kappa_{||,i+1}}{2}. \tag{4.2.21b}$$

The sign of $u_{||,i+\frac{1}{2}}$ determines the upwind direction for the numerical flux $\widehat{nT}^k_{e,i+\frac{1}{2}}$. Solving for $\mathbf{n}^{k+1} = \left[n_1^{k+1}, ..., n_{N_x}^{k+1}\right]^T$ is straightforward with equation (4.2.18a). A quasi-Newton solver is used to solve the system of equations (4.2.18b), (4.2.18c), (4.2.20) for the other three quantities

$$(\mathbf{nu}_{||})^{k+1} = \left[(nu_{||})_1^{k+1}, ..., (nu_{||})_{N_x}^{k+1}\right]^T, \qquad (\mathbf{nU})^{k+1}_\alpha = \left[(nU)^{k+1}_{\alpha,1}, ..., (nU)^{k+1}_{\alpha,N_x}\right]^T,$$

$$\mathbf{T}_e^{k+1} = \left[T^{k+1}_{e,1}, ..., T^{k+1}_{e,N_x}\right]^T.$$

Rather than compute the full Jacobian $\mathbf{J}$ of the system, we use a linear approximation of the Jacobian, $\mathbf{P} \approx \mathbf{J}$, that ignores negligably small terms based on the dimensional analysis in Appendix D. Letting $\mathbf{R}$ denote the residual and ℓ denote the iteration

counter in the quasi-Newton solver, we solve the linearized system

$$-\begin{bmatrix} \mathbf{R}^{(\ell)}_{nu_{||}} \\ \mathbf{R}^{(\ell)}_{(nU)_\alpha} \\ \mathbf{R}^{(\ell)}_{T_e} \end{bmatrix}_{3N_x\times 1} = \begin{bmatrix} \mathbf{P}^{(\ell)}_{nu_{||},nu_{||}} & \mathbf{P}^{(\ell)}_{nu_{||},(nU)_\alpha} & \mathbf{P}^{(\ell)}_{nu_{||},T_e} \\ \mathbf{P}^{(\ell)}_{(nU)_\alpha,nu_{||}} & \mathbf{P}^{(\ell)}_{(nU)_\alpha,(nU)_\alpha} & \mathbf{P}^{(\ell)}_{(nU)_\alpha,T_e} \\ \mathbf{P}^{(\ell)}_{T_e,nu_{||}} & \mathbf{P}^{(\ell)}_{Te,(nU)_\alpha} & \mathbf{P}^{(\ell)}_{T_e,T_e} \end{bmatrix}_{3N_x\times 3N_x} \begin{bmatrix} \delta(\mathbf{nu}_{||})^{(\ell+1)} \\ \delta(\mathbf{nU})^{(\ell+1)}_\alpha \\ \delta\mathbf{T}^{(\ell+1)}_e \end{bmatrix}_{3N_x\times 1} \tag{4.2.22}$$

for the updated macroscopic quantities until a specified tolerance is satisfied. In our numerical tests we terminate the solver if two tolerances are satisfied,

$$\left\|\mathbf{R}^{(\ell)}\right\|_\infty < \min\left\{5.0\times 10^{-12}, 5.0\times 10^{-10}\left\|\mathbf{R}^{k}\right\|_\infty\right\}. \tag{4.2.23}$$

Theoretically, the residual should be zero when using the full Jacobian. We enforce this by ensuring the infinity norm of the residual is just larger than $1.0E-13$ *and* roughly $1.0E-10$ times the norm of the residual at the previous time-step. We remark that the solver converged within a few iterations in our numerical tests. Letting

$$\mathbf{y}^k \doteq \begin{bmatrix} (\mathbf{nu}_{||})^k \\ (\mathbf{nU})^k_\alpha \\ \mathbf{T}^k_e \end{bmatrix}_{3N_x\times 1},$$

equation (4.2.22) can be written more simply as

$$-\mathbf{R}^{(\ell)} = \mathbf{P}^{(\ell)}\delta\mathbf{y}^{(\ell+1)}. \tag{4.2.24}$$

The exact expressions for the residual $\mathbf{R}$ and approximate Jacobian $\mathbf{P}$ can be found in Appendix G. Note that as seen in equation (4.2.22), the approximate Jacobian is a block matrix. And as seen in Appendix G, each block matrix is at most tridiagonal. This structure of $\mathbf{P}$ dramatically reduces the computational cost of the quasi-Newton solver. The quasi-Newton solver used to compute the updated macroscopic quantities is outlined in Algorithm 4.2.

Algorithm 4.2. A quasi-Newton solver for computing updated macroscopic quantities

---

Inputs: $\mathbf{y}^k$ and $\mathbf{n}^k$.

Outputs: $\mathbf{y}^{k+1}$ and $\mathbf{n}^{k+1}$.

1. Compute $\mathbf{n}^{k+1}$ using equation (4.2.18a).
2. Set $residualnorm = 1$, $tol = \min\left\{5.0E-12, 5.0E-10\left\|\mathbf{R}^k\right\|_\infty\right\}$, $\ell = 0$.
3. Set $\mathbf{y}^{(\ell)} = \mathbf{y}^k$.
4. Compute $\mathbf{R}^{(\ell)} = \mathbf{R}^k$ and $\mathbf{P}^{(\ell)} = \mathbf{P}^k$.

while $residualnorm > tol$

5a. Solve $-\mathbf{R}^{(\ell)} = \mathbf{P}^{(\ell)}\delta\mathbf{y}^{(\ell+1)}$ for $\delta\mathbf{y}^{(\ell+1)}$.

5b. Set $\mathbf{y}^{(\ell+1)} = \mathbf{y}^{(\ell)} + \delta\mathbf{y}^{(\ell+1)}$.

5c. Override $\mathbf{y}^{(\ell)} \coloneqq \mathbf{y}^{(\ell+1)}$.

5d. Compute and override $\mathbf{R}^{(\ell)}$ and $\mathbf{P}^{(\ell)}$.

5e. Compute and override $residualnorm = \left\|\mathbf{R}^{(\ell)}\right\|_\infty$.

5f. Override $\ell \coloneqq \ell + 1$.

6. Set $\mathbf{y}^{k+1} = \mathbf{y}^{(\ell)}$.

4.2.3.2 Discretizing the collision operator

With the updated macroscopic quantities computed, the implicitly treated Leonard-Bernstein-Fokker-Planck collision operator $C^{k+1}_{\alpha\alpha,i} + C^{k+1}_{\alpha e,i}$ can be discretized in velocity space. We discretize the collision operator using the SPCC method [139] for to its robustness and structure preservation (see Section 4.1.1). The SPCC method in [139] is presented in Cartesian coordinates but can easily be extended to cylindrical coordinates. Furthermore, the proven structure preservation of the SPCC method was only for the Fokker-Planck equation *without* a Vlasov component; and it was only for the full-rank/non-truncated solution. Proving the same structure preservation holds for our VLBFP model and low-rank framework is non-trivial. Yet, our numerical tests

observe the same structure-preserving qualities and suggest that the robustness of the SPCC discretization holds in our proposed scheme.

Without loss of generality, we only consider $C_{\alpha e}$ since discretizing $C_{\alpha\alpha}$ follows similarly. Moreover, we drop the subscript i and superscript $k+1$ for notational ease. It should be assumed that the discretizations in this subsection use values at spatial node x_i and time-level t^{k+1}, e.g.,

$$\begin{aligned} C_{\alpha e} &\longleftrightarrow C_{\alpha e,i}^{k+1} \\ f_\alpha^\star &\longleftrightarrow f_{\alpha,i}^{k+1,\star} \\ n, u_{||}, T_\alpha, T_e &\longleftrightarrow n_i^{k+1}, u_{||,i}^{k+1}, T_{\alpha,i}^{k+1}, T_{e,i}^{k+1} \end{aligned}$$

Expressing $C_{\alpha e}$ in the differential form used in [139],

$$\begin{aligned} \frac{C_{\alpha e}}{\nu_{\alpha e}} = \frac{1}{v_\perp}\frac{\partial}{\partial v_\perp}\left(v_\perp\left(\mathcal{B}_\perp[f_\alpha](v_\perp,t)f_\alpha + D_\perp[f_\alpha](v_\perp,t)\frac{\partial f_\alpha}{\partial v_\perp}\right)\right) \\ + \frac{\partial}{\partial v_{||}}\left(\mathcal{B}_{||}[f_\alpha](v_{||},t)f_\alpha + D_{||}[f_\alpha](v_{||},t)\frac{\partial f_\alpha}{\partial v_{||}}\right), \end{aligned} \tag{4.2.25}$$

where $\mathcal{B}_\perp[f_\alpha](v_\perp,t) = v_\perp$, $\mathcal{B}_{||}[f_\alpha](v_{||},t) = v_{||} - u_{||}$ and $D_\perp[f_\alpha](v_\perp,t) = D_{||}[f_\alpha](v_{||},t) = T_e/m_\alpha$. Note that although $D_\perp$ and $D_{||}$ are functionals dependent on f_α, their velocity derivatives are zero. Defining the fluxes

$$\mathcal{F}^{(\perp)}[f_\alpha](v_\perp,t) \doteq \mathcal{B}_\perp[f_\alpha](v_\perp,t)f_\alpha + D_\perp[f_\alpha](v_\perp,t)\frac{\partial f_\alpha}{\partial v_\perp}, \tag{4.2.26a}$$

$$\mathcal{F}^{(||)}[f_\alpha](v_{||},t) \doteq \mathcal{B}_{||}[f_\alpha](v_{||},t)f_\alpha + D_{||}[f_\alpha](v_{||},t)\frac{\partial f_\alpha}{\partial v_{||}}, \tag{4.2.26b}$$

the collision operator can be rewritten

$$\frac{C_{\alpha e}}{\nu_{\alpha e}} = \frac{1}{v_\perp}\frac{\partial}{\partial v_\perp}\Big(v_\perp\mathcal{F}^{(\perp)}[f_\alpha](v_\perp,t)\Big) + \frac{\partial}{\partial v_{||}}\Big(\mathcal{F}^{(||)}[f_\alpha](v_{||},t)\Big). \tag{4.2.27}$$

The Jacobian $v_\perp$ in the $v_\perp$-direction does not affect the equilibrium preservation of

the SPCC scheme since $v_\perp \mathcal{F}^{(\perp)} = 0 \Rightarrow \mathcal{F}^{(\perp)} = 0$ (see [36, 139] and Section 4.1.1 for how equilibrium preservation is enforced). Recall that the origin $v_\perp = 0$ is not a nodal point to avoid the singularity $1/v_\perp$.

We discretize each flux function separately. There is a *very important observation* to remark – each flux function is independent of the other direction, that is, $\mathcal{F}^{(\perp)} = \mathcal{F}^{(\perp)}[f_\alpha](v_\perp, t)$ and $\mathcal{F}^{(||)} = \mathcal{F}^{(||)}[f_\alpha](v_{||}, t)$. As such, e.g., the same discretization for $\mathcal{F}^{(\perp)}[f_\alpha](v_\perp, t)$ can be used for all $v_{||}$ values. Otherwise, each value of $v_{||}$ would require a different discretization for $\mathcal{F}^{(\perp)}[f_\alpha](v_\perp, t; v_{||})$.

SPCC discretization in the $v_{||}$-direction

The parallel flux term is discretized the same way as the SPCC scheme in Cartesian coordinates. The SPCC discretization in the $v_{||}$-direction is

$$\left[\nu_{\alpha e}\frac{\partial}{\partial v_{||}}\left(\mathcal{F}^{(||)}\right)\right]_{(v_{||,j_1}, v_{\perp,j_2})} \approx \nu_{\alpha e}\left(\frac{\mathcal{F}^{(||)}_{j_1+\frac{1}{2},j_2} - \mathcal{F}^{(||)}_{j_1-\frac{1}{2},j_2}}{\Delta v_{||}}\right), \tag{4.2.28}$$

where

$$\mathcal{F}^{(||)}_{j_1+\frac{1}{2},j_2} = A^{(||)}_{j_1+\frac{1}{2}}\left(\left(1-\delta^{(||)}_{j_1+\frac{1}{2}}\right)\mathbf{f}^\star_{\alpha,j_1+1,j_2} + \delta^{(||)}_{j_1+\frac{1}{2}}\mathbf{f}^\star_{\alpha,j_1,j_2}\right) + \frac{D_{||}}{\Delta v_{||}}\left(\mathbf{f}^\star_{\alpha,j_1+1,j_2} - \mathbf{f}^\star_{\alpha,j_1,j_2}\right), \tag{4.2.29a}$$

$$A^{(||)}_{j_1+\frac{1}{2}} = \frac{1}{\Delta v_{||}}\int_{v_{||,j_1}}^{v_{||,j_1+1}} \mathcal{B}_\perp(v_{||}, t^{k+1})dv_{||}, \tag{4.2.29b}$$

$$\lambda^{(||)}_{j_1+\frac{1}{2}} = \frac{\Delta v_{||}}{D_{||}}A^{(||)}_{j_1+\frac{1}{2}}, \tag{4.2.29c}$$

$$\delta^{(||)}_{j_1+\frac{1}{2}} = \frac{1}{\lambda^{(||)}_{j_1+\frac{1}{2}}} + \frac{1}{1-\exp(\lambda^{(||)}_{j_1+\frac{1}{2}})}. \tag{4.2.29d}$$

SPCC discretization in the $v_\perp$-direction

The perpendicular flux term is discretized similarly to the parallel flux term

with the addition of the Jacobian $v_\perp$. The SPCC discretization in the $v_{||}$-direction is

$$\left[\frac{\nu_{\alpha e}}{v_\perp}\frac{\partial}{\partial v_\perp}\left(v_\perp \mathcal{F}^{(\perp)}\right)\right]_{(v_{||,j_1},v_{\perp,j_2})} \approx \frac{\nu_{\alpha e}^{k+1}}{v_{\perp,j_2}}\left(\frac{v_{\perp,j_2+\frac{1}{2}}\mathcal{F}^{(\perp)}_{j_1,j_2+\frac{1}{2}} - v_{\perp,j_2-\frac{1}{2}}\mathcal{F}^{(\perp)}_{j_1,j_2-\frac{1}{2}}}{\Delta v_\perp}\right). \quad (4.2.30)$$

where

$$\mathcal{F}^{(\perp)}_{j_1,j_2+\frac{1}{2}} = A^{(\perp)}_{j_2+\frac{1}{2}}\left(\left(1-\delta^{(\perp)}_{j_2+\frac{1}{2}}\right)\mathbf{f}^\star_{\alpha,j_1,j_2+1} + \delta^{(\perp)}_{j_2+\frac{1}{2}}\mathbf{f}^\star_{\alpha,j_1,j_2}\right) + \frac{D_\perp}{\Delta v_\perp}\left(\mathbf{f}^\star_{\alpha,j_1,j_2+1} - \mathbf{f}^\star_{\alpha,j_1,j_2}\right), \quad (4.2.31a)$$

$$A^{(\perp)}_{j_2+\frac{1}{2}} = \frac{1}{\Delta v_\perp}\int_{v_{\perp,j_2}}^{v_{\perp,j_2+1}} \mathcal{B}_\perp(v_\perp, t^{k+1})dv_\perp, \quad (4.2.31b)$$

$$\lambda^{(\perp)}_{j_2+\frac{1}{2}} = \frac{\Delta v_\perp}{D_\perp}A^{(\perp)}_{j_2+\frac{1}{2}}, \quad (4.2.31c)$$

$$\delta^{(\perp)}_{j_2+\frac{1}{2}} = \frac{1}{\lambda^{(\perp)}_{j_2+\frac{1}{2}}} + \frac{1}{1-\exp(\lambda^{(\perp)}_{j_2+\frac{1}{2}})}. \quad (4.2.31d)$$

Since $v_\perp$ cannot take on negative values, the integral $A^{(\perp)}_{\frac{1}{2}}$ is over the interval $(0, \Delta v_\perp/2)$.

4.2.3.3 Discretizing the acceleration term

Using the updated macroscopic quantities computed, we discretize the acceleration term in (4.2.14) involving the electric field. The electric field at each spatial node x_i, $i = 1, 2, ..., N_x$ is approximated with centered differences on Ohm's law,

$$E^{k+1}_{||,i} = \frac{1}{q_e n^{k+1}_i}\frac{n^{k+1}_{i+1}T^{k+1}_{e,i+1} - n^{k+1}_{i-1}T^{k+1}_{e,i-1}}{2\Delta x}. \quad (4.2.32)$$

Using equation (4.2.32) to determine the proper upwind direction,

$$\left[\frac{q_\alpha}{m_\alpha}E^{k+1}_{||,i}\frac{\partial f^{k+1}_{\alpha,i}}{\partial v_{||}}\right]_{(v_{||,j_1},v_{\perp,j_2})} \approx \frac{q_\alpha}{m_\alpha}\mathcal{D}_E\left(\mathbf{f}^{k+1,\star}_{\alpha,i}\right), \quad (4.2.33)$$

where

$$\mathcal{D}_E\left(\mathbf{f}_{\alpha,i}^{k+1,\star}\right) = \max\left(E_{||,i}^{k+1},0\right)\frac{\mathbf{f}_{\alpha,i,j_1,j_2}^{k+1,\star} - \mathbf{f}_{\alpha,i,j_1-1,j_2}^{k+1,\star}}{\Delta v_{||}} + \min\left(E_{||,i}^{k+1},0\right)\frac{\mathbf{f}_{\alpha,i,j_1+1,j_2}^{k+1,\star} - \mathbf{f}_{\alpha,i,j_1,j_2}^{k+1,\star}}{\Delta v_{||}}. \tag{4.2.34}$$

4.3 Updating and truncating the discretized equation

Section 4.2 described how to discretize the stiff terms on the lefthand side of equation (4.2.14). With that done, equation (4.2.14) can be expressed as a linear system of tensor-product structure, and the updated full-rank/pre-truncated solution $\mathbf{f}_{\alpha,i}^{k+1,\star}$ can be computed at all spatial nodes x_i, $i = 1, 2, ..., N_x$. Then, a SVD basis removal procedure can be used to obtain the low-rank/truncated solution $\mathbf{f}_{\alpha,i}^{k+1}$.

4.3.1 Solving a linear system of tensor-product structure

Referring to the solver [79] outlined in Section (4.1.2), for each $i = 1, 2, ..., N_x$ we want to express equation (4.2.14) in the form

$$\left(\mathbf{A}_{||,i} \otimes \mathbf{I}_{N_\perp \times N_\perp} + \mathbf{I}_{N_{||} \times N_{||}} \otimes \mathbf{A}_{\perp,i}\right)\mathbf{f}_{\alpha,i}^{k+1,\star} = \mathbf{b}_i, \tag{4.3.1}$$

where $\mathbf{A}_{||,i}$ and $\mathbf{A}_{\perp,i}$ are the discretizations of the lefthand side of equation (4.2.14) in the parallel and perpendicular directions, respectively; and $\mathbf{b}_i$ is the righthand side of equation (4.2.14).

Let $\mathbf{A}_{\alpha e,i}^{(||)}$ and $\mathbf{A}_{\alpha\alpha,i}^{(||)}$ be the matrix representations of discretization (4.2.28) for the parallel terms of $C_{\alpha e,i}^{k+1}$ and $C_{\alpha\alpha,i}^{k+1}$, respectively. Similarly, let $\mathbf{A}_{\alpha e,i}^{(\perp)}$ and $\mathbf{A}_{\alpha\alpha,i}^{(\perp)}$ be the matrix representations of discretization (4.2.30) for the perpendicular terms of $C_{\alpha e,i}^{k+1}$ and $C_{\alpha\alpha,i}^{k+1}$, respectively. Let $\mathbf{A}_{E,i}^{(||)}$ be the matrix representation of discretization (4.2.33). All together,

$$\mathbf{A}_{||,i} \doteq \frac{1}{2}\mathbf{I}_{N_{||}\times N_{||}} - \Delta t\mathbf{A}_{\alpha\alpha,i}^{(||)} - \Delta t\mathbf{A}_{\alpha e,i}^{(||)} + \Delta t\mathbf{A}_{E,i}^{(||)}, \tag{4.3.2}$$

$$\mathbf{A}_{\perp,i} \doteq \frac{1}{2}\mathbf{I}_{N_\perp\times N_\perp} - \Delta t\mathbf{A}_{\alpha\alpha,i}^{(\perp)} - \Delta t\mathbf{A}_{\alpha e,i}^{(\perp)}. \tag{4.3.3}$$

And referring to solution (4.2.15) and the righthand side of equation (4.2.14),

$$\begin{aligned}\mathbf{b}_i \doteq & \sum_{r=1}^{R_i^k} c_{i,r}^k \mathbf{U}_{i,r}^{(1),k} \otimes \mathbf{U}_{i,r}^{(2),k} \\ & - \frac{\Delta t}{\Delta x}\left[\sum_{r=1}^{R_{i+1}^k} c_{i+1,r}^k \left(\min(\mathbf{v}_{||},\mathbf{0}) * \mathbf{U}_{i+1,r}^{(1),k}\right) \otimes \mathbf{U}_{i+1,r}^{(2),k} - \sum_{r=1}^{R_i^k} c_{i,r}^k \left(\min(\mathbf{v}_{||},\mathbf{0}) * \mathbf{U}_{i,r}^{(1),k}\right) \otimes \mathbf{U}_{i,r}^{(2),k}\right] \\ & - \frac{\Delta t}{\Delta x}\left[\sum_{r=1}^{R_i^k} c_{i,r}^k \left(\max(\mathbf{v}_{||},\mathbf{0}) * \mathbf{U}_{i,r}^{(1),k}\right) \otimes \mathbf{U}_{i,r}^{(2),k} - \sum_{r=1}^{R_{i-1}^k} c_{i-1,r}^k \left(\max(\mathbf{v}_{||},\mathbf{0}) * \mathbf{U}_{i-1,r}^{(1),k}\right) \otimes \mathbf{U}_{i-1,r}^{(2),k}\right],\end{aligned} \tag{4.3.4}$$

where $*$ denotes the Hadamard (elementwise) product. To utilize Grasedyck's solver [79] outlined in Algorithm 4.1, $\mathbf{b}_i$ needs to be expressed in the form (4.2.16). After which, Algorithm 4.1 can be used to compute the updated full-rank/pre-truncated solution, $\mathbf{f}_{\alpha,i}^{k+1,\star}$. Using the same notation as equation (4.1.9), the updated transfer coefficients are the coefficients

$$2w_j/\lambda_{min},$$

and the updated basis vectors in the parallel and perpendicular directions are respectively

$$\exp\left(\frac{2t_j}{\lambda_{min}}A_1\right) b_i^k \qquad \text{and} \qquad \exp\left(\frac{2t_j}{\lambda_{min}}A_2\right) b_i^k.$$

Referring to equations (4.1.9) and (4.3.4), the updated full-rank/pre-truncated solution

$$\mathbf{f}_{\alpha,i}^{k+1,\star} = \sum_{r=1}^{R_i^{k+1}} c_{i,r}^{k+1} \mathbf{U}_{i,r}^{(1),k+1} \otimes \mathbf{U}_{i,r}^{(2),k+1}$$

is rank $R_i^{k+1} = (3R_i^k + R_{i-1}^k + R_{i+1}^k)(2K+1)$, where $2K+1$ is the number of Stenger nodes/weights (which typically ranges from 31 to 301).

4.3.2 SVD truncation

The high-rank updated solution $\mathbf{f}_{\alpha,i}^{k+1,\star}$ necessitates the need to truncate and remove redundant basis vectors. Consider the updated solution stored as the matrix product

$$\mathbf{f}_{\alpha,i}^{k+1,\star} = \mathbf{U}_i^{(1),k+1}\mathbf{C}_i^{k+1}(\mathbf{U}_i^{(2),k+1})^T, \tag{4.3.5}$$

where $\mathbf{C}_i^{k+1}$ is diagonal, but the columns of $\mathbf{U}_i^{(1),k+1}$ and $\mathbf{U}_i^{(2),k+1}$ need not be orthogonal. Since the columns of $\mathbf{U}_i^{(1),k+1}$ and $\mathbf{U}_i^{(2),k+1}$ are not normalized, simply truncating the SVD of $\mathbf{C}_i^{k+1}$ will not correctly compute the optimal low-rank approximation.

Since we are just in two dimensions, computing the matrix multiplication (4.3.5) and then computing the SVD has a reasonable computational cost. Assuming $N_{||} = N_{\perp} = N$ and letting $R_i^{k+1} = R$ be the rank (where R could be larger than N), the matrix multiplication requires $\mathcal{O}(NR^2 + N^2R)$ flops. Computing the SVD of the $N \times N$ solution then requires $\mathcal{O}(N^3)$ flops.

In short, we compute the SVD of solution (4.3.5) and keep the singular values (and the corresponding singular vectors) larger than some tolerance $\epsilon > 0$. The numerical tests in this chapter set $\epsilon \in [1.0E-06, 1.0E-04]$. Redefine

$$\mathbf{U}_i^{(1),k+1}\mathbf{C}_i^{k+1}(\mathbf{U}_i^{(2),k+1})^T \coloneqq \text{svd}\left(\mathbf{f}_{\alpha,i}^{k+1,\star}\right) = \text{svd}\left(\mathbf{U}_i^{(1),k+1}\mathbf{C}_i^{k+1}(\mathbf{U}_i^{(2),k+1})^T\right).$$

Redefining the rank R_i^{k+1} as the number of singular values larger than $\epsilon > 0$, the low-rank/truncated solution is

$$\mathbf{f}_{\alpha,i}^{k+1} \doteq \sum_{r=1}^{R_i^{k+1}} c_{i,r}^{k+1}\mathbf{U}_{i,r}^{(1),k+1} \otimes \mathbf{U}_{i,r}^{(2),k+1}. \tag{4.3.6}$$

4.3.3 The first-order scheme

We summarize the proposed first-order scheme below in Algorithm 4.3. The solution $\mathbf{f}_{\alpha,i}^{k+1}$ at each spatial node x_i has rank R_i^{k+1}, for $i = 1, 2, ..., N_x$. To measure

how the overall rank evolves in time, we compute and store the *average rank*

$$R^{k+1} \doteq \frac{1}{N_x} \sum_{i=1}^{N_x} R_i^{k+1}. \tag{4.3.7}$$

Algorithm 4.3. The first-order scheme with IMEX(1,1,1)

Inputs: $\mathbf{f}_{\alpha,i}^k$ and rank R_i^k, for $i = 1, 2, ..., N_x$.

Outputs: $\mathbf{f}_{\alpha,i}^{k+1}$ and rank R_i^{k+1}, for $i = 1, 2, ..., N_x$.

Compute $\mathbf{n}^{k+1}$, $(\mathbf{nu}_{||})^{k+1}$, $(\mathbf{nU})_{\alpha}^{k+1}$ and T_e^{k+1} using Algorithm 4.2.

do $i = 1, 2, ..., N_x$

1a. Compute the discretizations $\mathbf{A}_{||,i}$ and $\mathbf{A}_{\perp,i}$, see equations (4.3.2).

1b. Compute $\mathbf{b}_i$, see equation (4.3.4).

2. Solve (4.3.1) for $\mathbf{f}_{\alpha,i}^{k+1,\star}$ using Algorithm 4.1.

3. Compute the matrix product (4.3.5) and compute its SVD.

4. Truncate the SVD according to tolerance $\epsilon > 0$.

5. Define the low-rank solution $\mathbf{f}_{\alpha,i}^{k+1}$ and rank R_i^{k+1}, see equation (4.3.6).

4.4 Numerical tests

We present numerical results verifying the proposed scheme's performance, low-rank structure, and observed structure-preserving qualities. Only the single ion species case is considered, and the proposed scheme is tested on both the Vlasov-Leonard-Bernstein-Fokker-Planck and the Leonard-Bernstein-Fokker-Planck equations. The following results were computed on the same computer mentioned in Chapter 3, Section 3.3.1.

Unless otherwise stated, we use a time-stepping size $\Delta t = 0.3$ and a 7-point Gauss-Legendre quadrature (see Section 2.1.4) to approximate the integrals in the SPCC discretization. We are still limited by the CFL condition from the explicit

treatment of the Vlasov transport term, but $\Delta t = 0.3$ is small enough for the spatial meshes considered in this section. Parameter K denotes using $2K+1$ Stenger quadrature nodes/weights in Algorithm 4.1. Parameter $\epsilon > 0$ denotes the tolerance used to truncate the SVD, see Algorithm 4.3.

4.4.1 The 0D2V Leonard-Bernstein-Fokker-Planck equation

Low-rank structure and observed structure-preserving qualities are verified on the (nondimensionalized) Leonard-Bernstein-Fokker-Planck equation,

$$\frac{\partial f_\alpha}{\partial t} = \nu_{\alpha\alpha} \nabla_{\mathbf{v}} \cdot \left(\frac{T_\alpha}{m_\alpha} \nabla_{\mathbf{v}} f_\alpha + (\mathbf{v} - \mathbf{u}_\alpha) f_\alpha \right), \tag{4.4.1}$$

where the mass and charge are scaled to unity, $m_\alpha = 1$ and $q_\alpha = 1$, and the velocity mesh is $(v_{||}, v_\perp) \in [-14, 16] \times [0, 14]$. Equation (4.4.1) lacks physical-spatial dependence and is not coupled with the fluid-electron energy equation (4.2.2). As such, Algorithm 4.3 only acts on a single loop (i.e., a single spatial node) and does not require the quasi-Newton solver in Algorithm 4.2. Referring to Appendix E, the zeroth-, first- and second-order moments of equation (4.4.1) (without spatial dependence and $C_{\alpha e}$) imply that $\left[n, nu_{||}, (nU)_\alpha\right]^k = \left[n, nu_{||}, (nU)_\alpha\right]^{k+1}$. Hence, we use the local macroscopic quantities from the initial condition (or from the equilibrium solution) in the discretization of the LBFP collision operator.

The first-order IMEX scheme reduces to the first-order backward Euler scheme since $C_{\alpha\alpha}$ is evolved implicitly. Although time-stepping sizes larger than $\Delta t = 0.3$ can be used, we still use $\Delta t = 0.3$ to minimize the error from the time-stepping and stay consistent with the results for the VLBFP equation. The initial time-step is size $\Delta t^0 = 5 \times 10^{-3}$ to allow the initial dynamics to be captured more accurately, but $\Delta t = 0.3$ is used for all subsequent time-steps.

The presented results for equation (4.4.1) assume a Maxwellian equilibrium solution

$$f_M(v_{||}, v_\perp) = \frac{n}{(2\pi RT)^{3/2}} \exp\left(-\frac{(v_{||} - u_{||})^2 + v_\perp^2}{2RT} \right), \tag{4.4.2}$$

where gas constant $R = 1$, number density $n = \pi$, bulk velocity $\mathbf{u} = \mathbf{0}$ and temperature $T = 3$. The equilibrium distribution function is shown in Figure 4.3(a).

When testing the relaxation of the system, we set the initial distribution function as the sum of two randomly generated Maxwellians, $f(v_{||}, v_{\perp}, t = 0) = f_{M1}(v_{||}, v_{\perp}) + f_{M2}(v_{||}, v_{\perp})$, such that the total macroscopic parameters of the equilibrium distribution function are preserved. That is, the total number density, bulk velocity and temperature are $n = \pi$, $\mathbf{u} = \mathbf{0}$ and $T = 3$. The number densities, bulk velocities and temperatures of each Maxwellian are listed in Table 4.1. We set $u_{\perp} = 0$ so that the two Maxwellians only shift in the $\hat{\mathbf{v}}_{||}$-direction. The initial distribution function is shown in Figure 4.2(a).

	f_{M1}	f_{M2}
n	1.902813990281176	1.238778663308618
$u_{\|\|}$	-2.113532196926305	3.246470699196378
$u_{\perp}$	0	0
T	1.10608227396894	0.1087698976066122

Table 4.1: $R = 1$, $n = \pi$, $\overline{\mathbf{v}} = \mathbf{0}$ and $T = 3$.

Although the structure-preserving qualities of the SPCC discretization are non-trivial to prove for the proposed scheme, particularly due to truncation, we still observe some of the same behaviors as shown in the results in [139]. There are essentially four components that can affect the error: time-stepping size Δt, mesh size $\Delta v_{||}$ and $\Delta v_{\perp}$, the singular value tolerance $\epsilon > 0$, and the Stenger quadrature K. The time-stepping size is fixed for our simulations at $\Delta t = 0.3$. Unfortunately, depending on the problem, the Stenger quadrature as applied in Algorithms 4.1 and 4.3 roughly requires $K \in [15, 200]$ to achieve three to four digits of accuracy.

For the base test, we use mesh 400×400, Stenger quadrature $K = 200$, and tolerance $\epsilon = 1.0E - 05$. The base test is represented by the black lines in Figures 4.2 and 4.3. As seen in the figures, the base test observes low rank, equilibrium preservation as indicated by the L^1 decay, and discrete relative entropy dissipation. Moreover,

the parameters Δt, $\Delta v_{||}$, $\Delta v_{\perp}$, ϵ and K only captures the equilibrium solution with $\mathcal{O}(1.0E-04)$ accuracy. We adjust the mesh size and the SVD tolerance to check how they affect the observables.

In addition to the base test, Figures 4.2(b)-(d) show what happens if $\epsilon = 1.0E-02$ (while keeping the other parameters unchanged). Figure 4.2(b) shows that the rank decreases with smaller ϵ since the singular values are truncated to a smaller tolerance. Figure 4.2(c) shows that decreasing ϵ to $1.0E-02$ also lowered the accuracy with which the equilibrium solution is captured to $\mathcal{O}(1.0E-02)$. Figure 4.2(d) shows that the discrete relative entropy dissipates at the same rate for both tolerances.

In addition to the base test, Figures 4.3(b)-(d) show what happens if the mesh is 200×200 (while keeping the other parameters unchanged). Figure 4.3(b) shows that the rank remains the same for both mesh sizes. Figure 4.3(c) shows that the coarser mesh 200×200 lowered the accuracy with which the equilibrium solution is captured to $\mathcal{O}(1.0E-03)$. Figure 4.3(d) shows that the discrete relative entropy dissipates at the same rate for both mesh sizes.

The major takeaways from these results are that low-rank structure is enforced, and equilibrium-preservation and relative entropy dissipation are observed. Recall that the rank of the pre-truncated solution at each time-step is $\mathcal{O}(2K+1)$ which for $K = 200$ is quite massive. Figures 4.2(b) and 4.3(b) show that the truncated solution is less than rank 10. This is a huge savings in both storage and computational cost since the rank would grow exponentially without truncation; this is due to the solver used for updating the solution.

It is also important to note that Figures 4.2 and 4.3 only give insight to how $\Delta v_{||}$, $\Delta v_{\perp}$, ϵ and K can affect the solution. The extent to which changing these parameters affect the solution will slightly change depending on the problem.

4.4.2 The 1D2V Vlasov-Leonard-Bernstein-Fokker-Planck equation

Low-rank structure and the scheme's performance are presented. We simulate the 1D2V Mach-5 steady-state shock problem from [162]. The spatial domain is $x \in$

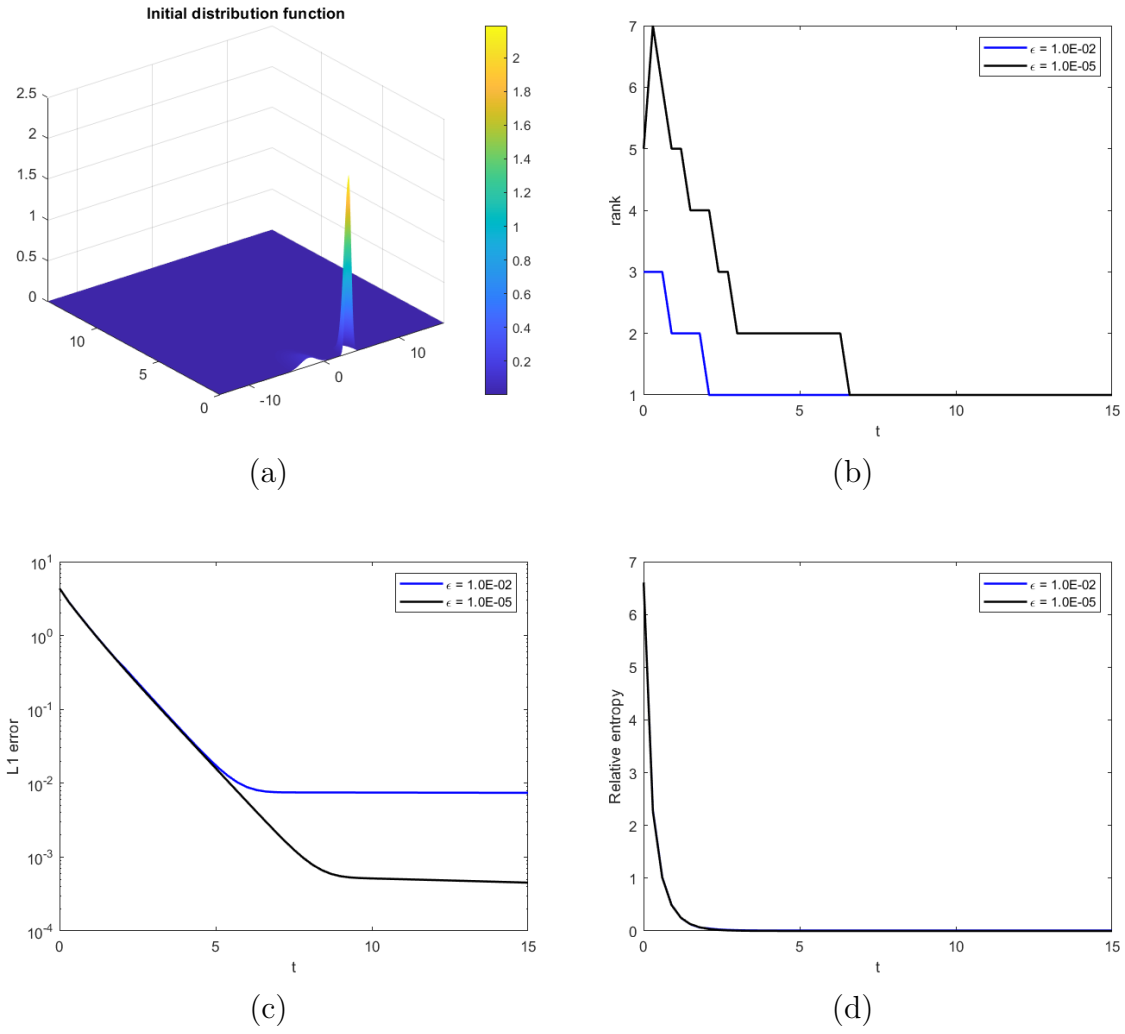


Figure 4.2: Mesh 400×400, time-stepping size $\Delta t = 0.3$, Stenger quadrature $K = 200$, tolerances $\epsilon = 1.0E-05$ and $\epsilon = 1.0E-02$. Figure (a): initial distribution of two Maxwellians defined by Table 4.1. Figure (b): rank evolution. Figure (c): L^1 error, $\|f_\alpha - f_M\|_1$. Figure (d): discrete Kullback relative entropy, $\mathcal{H}_\Delta(f_\alpha, f_M)$.

$[0, 200]$ and the velocity domain is $(v_{||}, v_\perp) \in [-8, 10] \times [0, 8]$. An initial shock occurs at $x = 100$ with constant downstream and upstream plasma conditions. The normalized plasma conditions downstream of the shock are number density $n = 1$, drift velocity $u_{||} = 0.8676$, and temperature $T = 1$. The normalized plasma conditions upstream of the shock are number density $n = 0.28$, drift velocity $u_{||} = 3.0984$, and temperature $T = 0.1152$. As seen in Figure 4.4(a), the initial shock is smoothed with hyperbolic

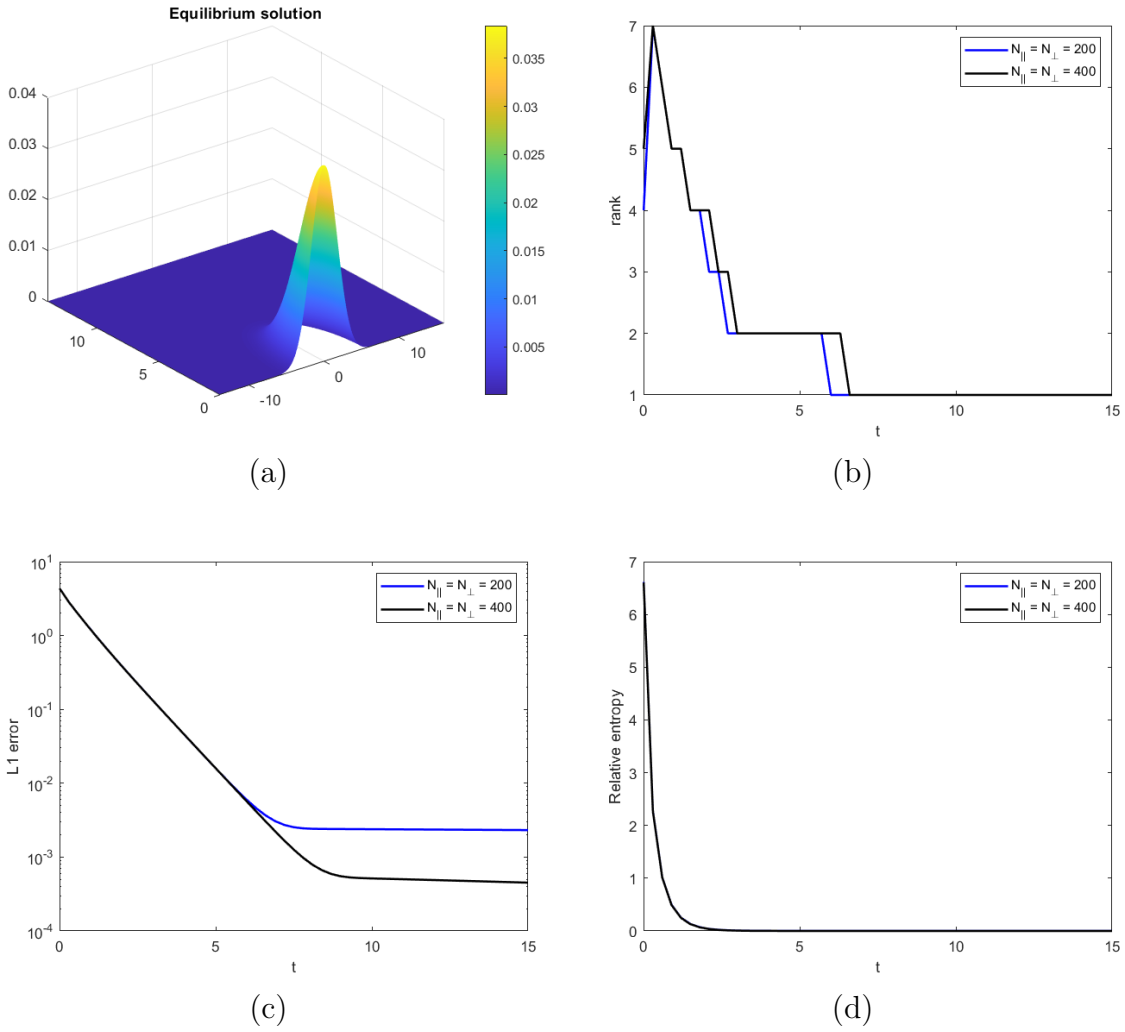


Figure 4.3: Time-stepping size $\Delta t = 0.3$, Stenger quadrature $K = 200$, tolerance $\epsilon = 1.0E-05$, meshes 200×200 and 400×400. Figure (a): equilibrium distribution function, f_M. Figure (b): rank evolution. Figure (c): L^1 error, $\|f_\alpha - f_M\|_1$. Figure (d): discrete Kullback relative entropy, $\mathcal{H}_\Delta(f_\alpha, f_M)$.

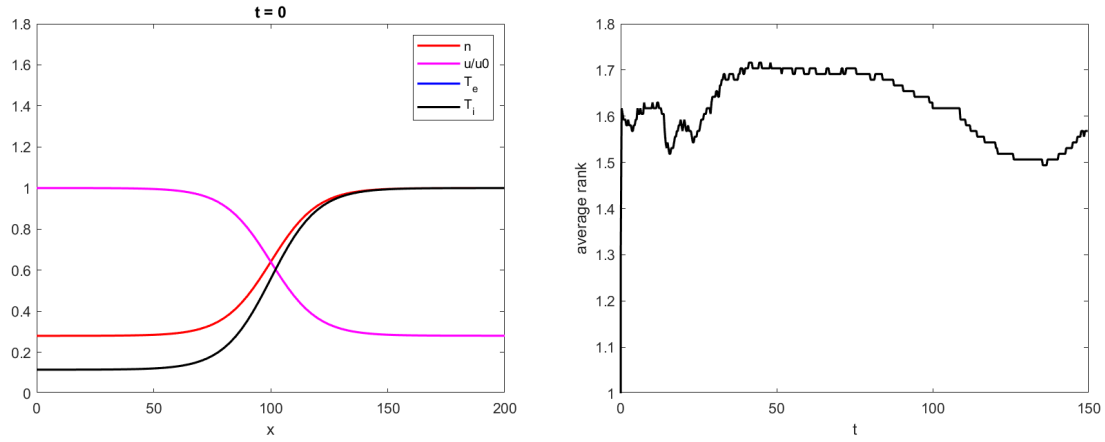
tangents whose profiles are given by

$$n(x, t=0) = 0.36 \tanh\left[0.05(x-100)\right] + 0.64, \tag{4.4.3a}$$

$$u_{||}(x, t=0) = -1.1154 \tanh\left[0.05(x-100)\right] + 1.983, \tag{4.4.3b}$$

$$T(x, t=0) = 0.4424 \tanh\left[0.05(x-100)\right] + 0.5576. \tag{4.4.3c}$$

The ion and electron temperatures are equal at the boundary, and $T_\alpha(x, t = 0) = T_e(x, t = 0)$. The other normalized parameters are proton mass $m_\alpha = 1$, electron mass $m_e = 1/1836$, proton charge $q_\alpha = 1$, and electron charge $q_e = -1$. Recall that the Maxwellian distributions constructed at the boundaries by the initial shock define the Dirichlet boundary conditions in space.



(a) (b)

Figure 4.4: Figure (a): initial shock (4.4.3) for the 1D2V VLBFP test. The smoothed number density, drift velocity, ion temperature and electron temperature profiles are shown. Spatial mesh $N_x = 80$. Figure (b): average nodal rank.

The results for this test use a spatial mesh $N_x = 80$, velocity mesh $N_{||} \times N_\perp = 120 \times 120$, singular value tolerance $\epsilon = 1.0E - 05$, and Stenger quadrature $K = 15$. The initial time-step is size $\Delta t^0 = 5 \times 10^{-3}$ to allow the initial dynamics to be captured more accurately, but $\Delta t = 0.3$ is used for all subsequent time-steps.

Figure 4.4(b) shows that the scheme is maintaining the low-rank structure of the solution, with the average nodal rank varying between 1.5 and 1.7. Given that the pre-truncated solution at each time-step has rank $\mathcal{O}(2K + 1) = \mathcal{O}(31)$, this indicates significant storage and computational savings. Figures 4.5(a)-(d) show the evolution of the total mass, momentum, energy and electron pressure. All four quantities grow indefinitely in our results, whereas they should relax when approaching relaxation/equilibrium. This indicates lack of conservation in our method, most likely

from a pumping or depleting of energy. Pumping/depleting of energy in a system can cause unphysical acoustic (pressure) waves to form. Aside from the electron temperature, the total electron pressure is the fastest to relax. Since all four quantities start growing around time $t = 50$, this suggests that energy pumping/depletion might be source of conservation loss (from a physics-informed perspective).

Mathematically, the loss of conservation is caused by the SVD truncation. Although the basis removal procedure offers the optimal low-rank solution (by virtue of the SVD), it is well-known that the SVD destroys conservation [84, 85]. However, the focus of the proposed method was computing low-rank solutions, not enforcing conservation. To enforce conservation, other truncation algorithms need to be considered. One such algorithm is the local macroscopic conservative (LoMaC) low-rank tensor method [85]. This truncation algorithm enjoys the exact local conservation of macrosopic mass, momentum and energy at the discrete level.

The profiles of the macroscopic quantities are shown in Figure 4.6 at various snapshots in time. Since conservation is not enforced, one cannot hope to observe the correct profiles, which can be found in [162]. The spatial boundary conditions taken from the initial distribution function in Figure 4.4(a) are not maintained in the solution. The number density at $x = 200$ is not $n = 1$ at equilibrium, and the spatial slopes at $x = 200$ of the number density and temperatures are not zero. Once conservation is enforced in the proposed scheme, we hope to obtain results consistent with those in [162].

Table 4.2 shows the average CPU runtime per time-step using Algorithm 4.3, as well as how much of the CPU runtime was spent on the solving the linear system. As seen in the table, 99.16% of the CPU runtime was spent on solving the linear system for the pre-truncated updated solution. Moreover, these results were for a very modest Stenger quadrature of $K = 15$. Hence, the proposed scheme is prohibitively expensive for large K and motivates the need for other solvers in future work. Note that the CPU runtimes shown in Table 4.2 accounts for a single time-step over *all* $N_x = 80$ spatial nodes. There are two silver linings to this result. First, the proposed scheme is

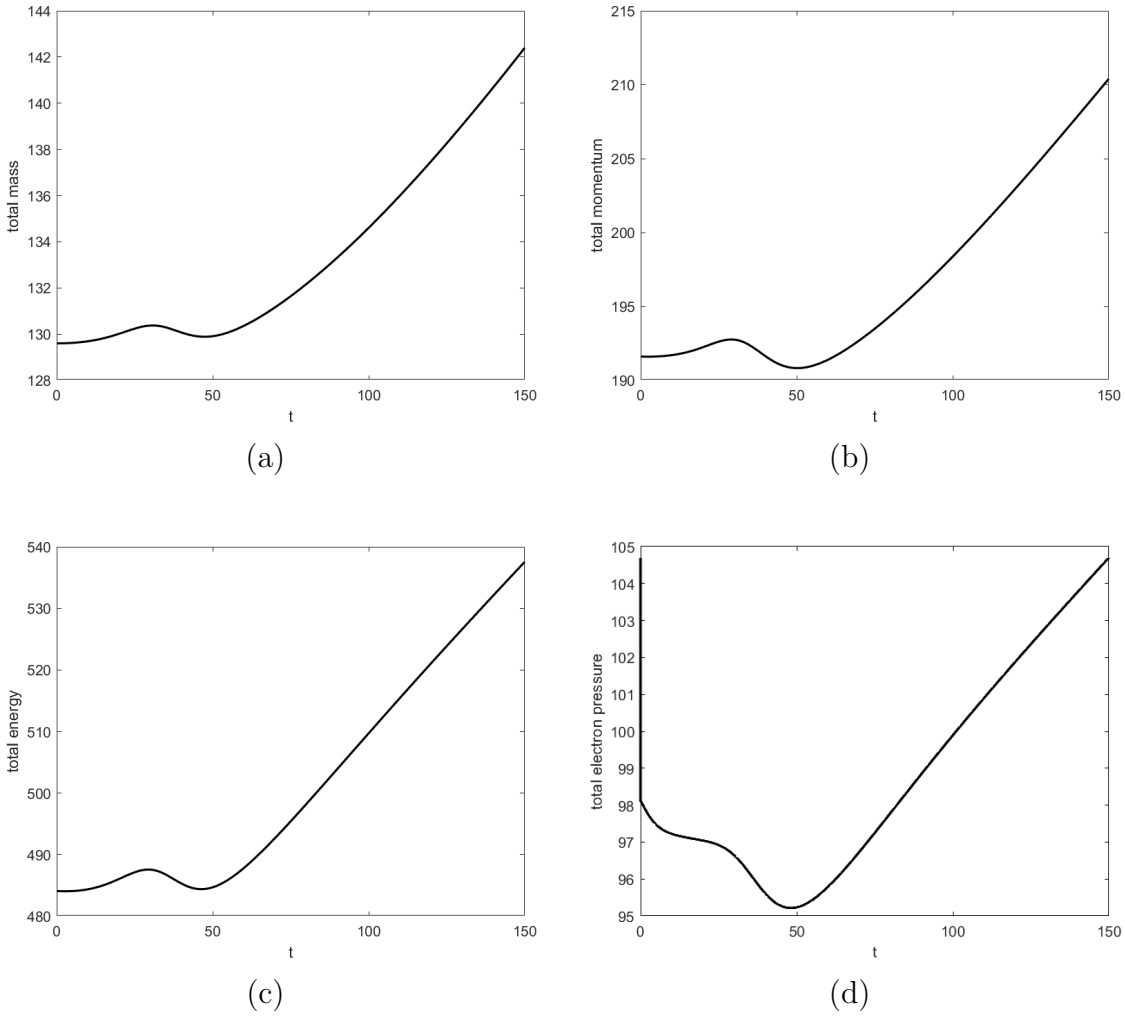


Figure 4.5: Time-stepping size $\Delta t^0 = 5 \times 10^{-3}$ and $\Delta t^k = 0.3$, $k = 1, ..., N_t$, Stenger quadrature $K = 15$, tolerance $\epsilon = 1.0E-05$, spatial mesh $N_x = 80$, velocity mesh 120×120. Figure (a): total mass (4.2.8a). Figure (b): total momentum (4.2.8b). Figure (c): total energy (4.2.8c). Figure (d): total electron pressure $p_e(x,t) = (n_e T_e)(x,t)$.

highly parallelizable since the solution at each spatial node can be solved independently. Although we did not implement this in our test, doing so would theoretically reduce the CPU runtime by a factor of N_x. Second, there is much room for improvement. Using more efficient solvers, such as Krylov subspace solvers, to update the solution can significantly improve the usability of the scheme.

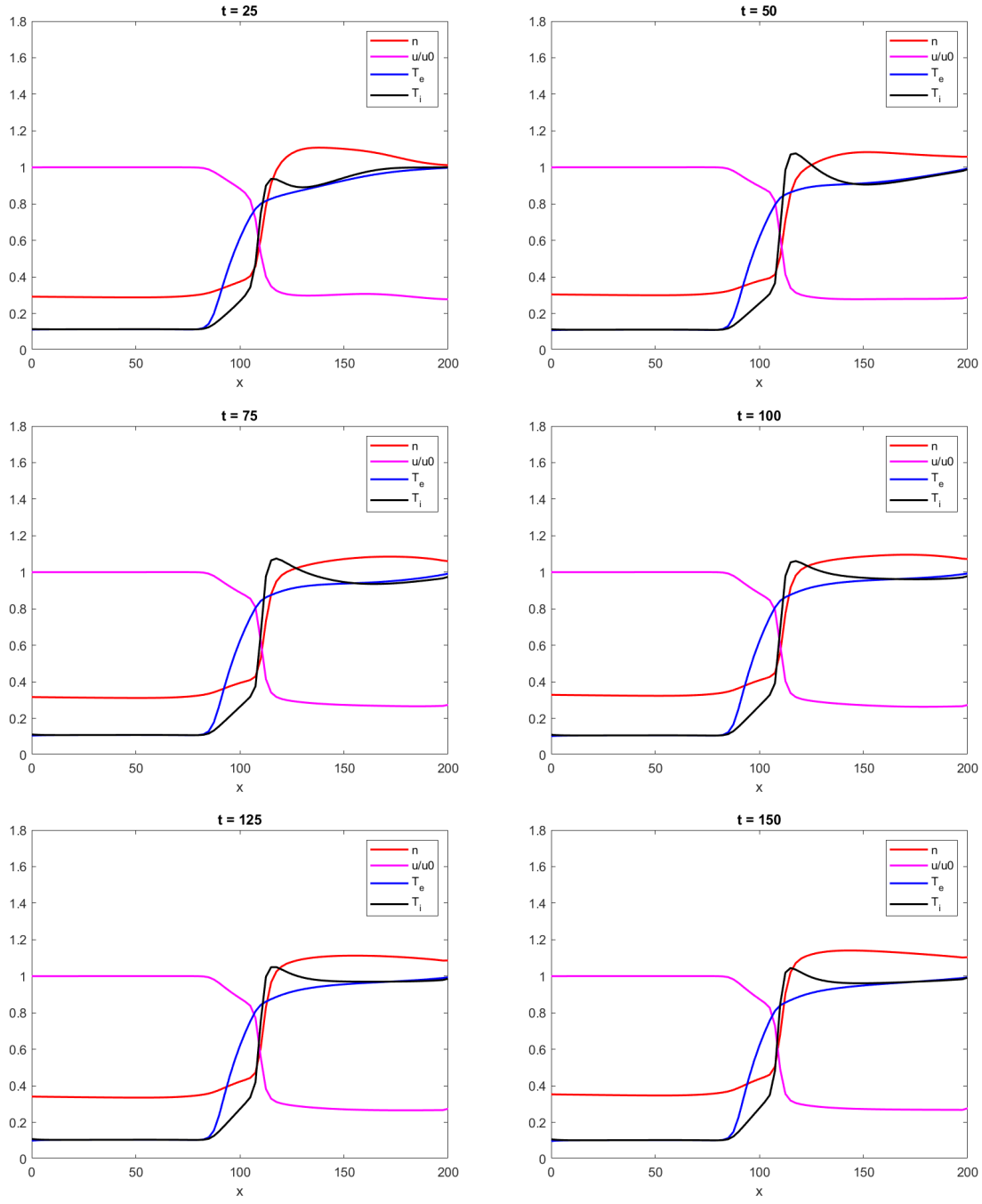


Figure 4.6: Various snapshots of the numerical solution to the 1D2V VLBFP equation with initial distribution 4.4.3. Time-stepping size $\Delta t^0 = 5 \times 10^{-3}$ and $\Delta t^k = 0.3$, $k = 1, ..., N_t$, Stenger quadrature $K = 15$, tolerance $\epsilon = 1.0E - 05$, spatial mesh $N_x = 80$, velocity mesh 120×120. Times: 25, 50, 75, 100, 125, 150.

CPU runtime for Step 2 (Algorithm 4.3)	Total CPU runtime for Algorithm 4.3	Ratio: (Step 2 runtime)/(Total runtime)
91.7946 seconds	92.5764 seconds	99.16%

Table 4.2: The average CPU runtime per time-step simulating the VLBFP equation to relaxation. Spatial mesh $N_x = 80$, velocity mesh $N_{||} \times N_{\perp} = 120 \times 120$, singular value tolerance $\epsilon = 1.0E-05$, Stenger quadrature $K = 15$.

4.5 Conclusions and follow-up work

In this chapter we proposed a low-rank scheme for solving the 1D2V VLBFP equation using a hybrid kinetic-ion fluid-electron model. The low-rank structure was incorporated by discretizing in space and evolving a low-rank 2V solution at each spatial node. The LBFP collision operator was discretized using the structure-preserving SPCC method [139], which we extended to cylindrical coordinates. The solution was updated using a solver for linear systems of tensor product structure [79], and the SVD was used to truncate the updated solution. The proposed scheme was tested on the 0D2V LBFP equation and 1D2V VLBFP equation at relaxation. Results showed the structure-preserving qualities from the SPCC discretization are observed, and that the low-rank structure is enjoyed. Despite these two positives, the scheme is not conservative by virtue of the SVD truncation, and the implicit solver [79] made up a majority of the runtime. Furthermore, the Stenger quadrature used in the linear solver has a great affect on the accuracy of the results; $K \in [15, 200]$ only gave three to four digits of accuracy in our tests. However, this opens the door for significant improvements. Ongoing and future work includes using the DLR inspired algorithm in Chapter 3 to replace the implicit solver, and implementing the conservative truncation presented in [85]. Both modifications would address and improve the two downsides to the proposed scheme. Lastly, the current scheme is held back by a CFL condition due to the explicit treatment of the Vlasov transport term. Ultimately, we would like to use a semi-Lagrangian [142] or Eulerian-Lagrangian [133] method (see Chapter 2) to evolve the Vlasov transport term to allow large time-stepping sizes.

BIBLIOGRAPHY

[1] I.G. Abel, et al., Linearized model Fokker-Planck collision operators for gyrokinetic simulations. I. Theory, *Physics of Plasmas*, **15:12** (2008), pp. 122509.

[2] E. Abreu, W. Lambert, J. Perez, and A. Santo, A new finite volume approach for transport models and related applications with balancing source terms, *Math. and Comput. in Simul.*, **137** (2017), pp. 2-28.

[3] M. Ahn, et al., On large-scale dynamic topic modeling with nonnegative CP tensor decomposition, *Advances in Data Science*, (2021), pp. 181-210.

[4] R. Alexander, Diagonally implicit Runge-Kutta methods for stiff ODE's, *SIAM J. Numer. Anal.*, **14:6** (1977), pp. 1006-1021.

[5] E. Anderson, et al., LAPACK users' guide, *SIAM*, (1999).

[6] S.E. Anderson, W.T. Taitano, L. Chacon, and A.N. Simikov, An efficient, conservative, time-implicit solver for the fully kinetic arbitrary-species 1D-2V Vlasov-Ampere system, *J. Comput. Phys.*, **419** (2020), pp. 109686.

[7] T. Arbogast, C.-S. Huang, and X. Zhao, Finite volume WENO schemes for nonlinear parabolic problems with degenerate diffusion on non-uniform meshes, *J. Comput. Phys.*, **399** (2019), pp. 108921.

[8] T. Arbogast, C.-S. Huang, X. Zhao, and D.N. King, A third order, implicit, finite volume, adaptive Runge–Kutta WENO scheme for advection–diffusion equations, *Comp. Meth. in Appl. Mech. and Eng.*, **368** (2020), pp. 113155.

[9] A.A. Arsen'ev and O.E. Buryac, On the connection between a solution of the Boltzmann equation and a solution of the Landau-Fokker-Planck equation, *USSR Comput. Maths math. Phys.*, **17** (1991), pp. 241-246.

[10] U.M. Ascher, S.J. Ruuth, and R.J. Spiteri, Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations, *App. Numer. Math.*, **25:2-3** (1997), pp. 151-167.

[11] D.S. Balsara, S. Garain, and C.-W. Shu, An efficient class of WENO schemes with adaptive order, *J. Comput. Phys.*, **326** (2016), pp. 780-804.

[12] D.S. Balsara, S. Garain, V. Florinski, and W. Boscheri, An efficient class of WENO schemes with adaptive order for unstructured meshes, *J. Comput. Phys.*, **404** (2020), pp. 109062.

[13] M. Barnes, et al., Linearized model Fokker-Planck collision operators for gyrokinetic simulations. II. Numerical implementation and tests, *Physics of Plasmas*, **16:7** (2009), pp. 072107.

[14] R.H. Bartels and G.W. Stewart, Solution of the matrix equation AX+BX=C, *Communications of the ACM*, **15:9** (1972), pp. 820-826.

[15] F. Benkhaldoun, S. Sari, and M. Seaid, A family of finite volume Eulerian-Lagrangian methods for two-dimensional conservation laws, *J. Comput. and App. Math.*, **285** (2015), pp. 181-205.

[16] P. Benner, et al., SLICOT - A Subroutine Library in Systems and Control Theory, *Applied and Computational Control, Signals, and Circuits (Birkhauser)*, **1** (1999), pp. 505-546.

[17] C.K. Birdsall and A.B. Langdon, Plasma physics via computer simulation, *CRC Press*, 2004.

[18] L.S. Blackford, et al., Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard, *Int. J. High Perform. Comput.*, **16** (2002), pp. 1-2. (also available at www.netlib.org/blas/blast-forum).

[19] R. Borges, M. Carmona, B. Costa, and W.S. Don, An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws, *J. Comput. Phys.*, **227:6** (2008), pp. 3101-3211.

[20] S. Boscarino, F. Filbet, and G. Russo, High order semi-implicit schemes for time dependent partial differential equations, *J. Sci. Comput.*, **68** (2016), pp. 975-1001.

[21] W. Boscheri and M. Dumbser, A direct Arbitrary-Lagrangian–Eulerian ADER-WENO finite volume scheme on unstructured tetrahedral meshes for conservative and non-conservative hyperbolic systems in 3D, *J. Comput. Phys.*, **275** (2014), pp. 484-523.

[22] W. Boscheri and M. Dumbser, Arbitrary-Lagrangian-Eulerian One-Step WENO Finite Volume Schemes on Unstructured Triangular Meshes, *Comm. in Comput. Phys.*, **14:5** (2013), pp. 1174-1206.

[23] W. Boscheri, R. Loubere, and M. Dumbser, Direct Arbitrary-Lagrangian–Eulerian ADER-MOOD finite volume schemes for multidimensional hyperbolic conservation laws, *J. Comput. Phys.*, **292** (2015), pp. 56-87.

[24] W. Boscheri, High Order Direct Arbitrary-Lagrangian-Eulerian (ALE) Finite Volume Schemes for Hyperbolic Systems on Unstructured Meshes, *Arch. Comput. Meth. Eng.*, **24** (2017), pp. 751-801.

[25] N. Boullé and A. Townsend, A generalization of the randomized singular value decomposition, *In: International Conference on Learning Representations* (2022b).

[26] C. Buet and S. Dellacherie, On the Chang and Cooper scheme applied to a linear Fokker-Planck equation, *Commun. Math. Sci.*, **8:4** (2010), pp. 1079-1090.

[27] C. Buet and K.C. Le-Thanh, Positive, conservative, equilibrium state preserving and implicit difference schemes for the isotropic Fokker-Planck-Landau equation, *HAL-00142408* (2007).

[28] X. Cai, W. Guo, and J.-M. Qiu, A high order conservative semi-Lagrangian discontinuous Galerkin method for two-dimensional transport simulations, *J. Sci. Comput.*, **73** (2017) pp. 514-542.

[29] J.A. Carrillo and F. Vecil, Nonoscillatory interpolation methods applied to Vlasov-Based models, *SIAM J. Sci. Comput.*, **29** (2007), pp. 1179-1206.

[30] M.A. Celia, T.F. Russell, I. Herrera, and R.E. Ewing, An Eulerian-Lagrangian localized adjoint method for the advection-diffusion equation, *Adv. Water Resour.*, **13** (1990), pp. 187-206.

[31] C. Cercignani, R. Illner, and M. Pulvirenti, The mathematical theory of dilute gases, *Applied Mathematical Sciences, 106, Springer*, New York, 1994.

[32] G. Ceruti, J. Kusch, and C. Lubich, A rank-adaptive robust integrator for dynamical low-rank approximation, *BIT Numer. Math.*, **62** (2022), pp. 1149-1174.

[33] G. Ceruti and C. Lubich, An unconventional robust integrator for dynamical low-rank approximation, *BIT Numer. Math.*, **62** (2022), pp. 23-44.

[34] G. Ceruti, C. Lubich, and H. Walach, Time integration of tree tensor networks, *SIAM J. Numer. Anal.*, **59:1** (2021), pp. 289-313.

[35] J. Cervi and R.J. Spiteri, High-order operator splitting for the bidomain and monodomain models, *SIAM J. Sci. Comput.*, **40:2** (2018), pp. A769-A786.

[36] J.S. Chang and G. Cooper, S practical difference scheme for Fokker-Planck equations, *J. Comput. Phys.*, **6:1** (1970), pp. 1-16.

[37] S. Chapman and T.G. Cowling, The mathematical theory of non uniform gases, 3rd edn, *Cambridge University Press*, 1970.

[38] J. Chen, Characteristics-based high-order methods with WENO reconstructions for linear and nonlinear dynamics, *University of Delaware*, (2022).

[39] J. Chen, X. Cai, J. Qiu, and J.-M. Qiu, Adaptive Order WENO Reconstructions for the Semi-Lagrangian Finite Difference Scheme for Advection Problem, *Comm. in Comput. Phys.*, **30:1** (2021), pp. 67-96.

[40] J. Chen, J. Nakao, and J.-M. Qiu, High-order Eulerian-Lagrangian Runge-Kutta finite volume (EL-RK-FV) methods for nonlinear hyperbolic problems with shocks, *In preparation.*

[41] H. Cho, D. Venturi, and G.E. Karniadakis, Numerical methods for high-dimensional kinetic equations, *In: S. Jin, L. Pareschi (eds) Uncertainty quantification for kinetic and hyperbolic equations*, *Springer*, Berlin, 2017.

[42] A.K. Cline and I.S. Dhillon, Computation of the singular value decomposition, In: Handbook of linear algebra, L. Hogben (ed.), *Chapman and Hall/CRC*, 2006.

[43] B. Cockburn, C. Johnson, C.-W. Shu, and E. Tadmor, Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, A. Quarteroni (ed.), *Lecture Notes in Math 1697, Springer*, New York, 1997.

[44] S. Conde, S. Gottlieb, Z.J. Grant, and J.N. Shadid, Implicit and Implicit-Explicit Strong Stability Preserving Runge-Kutta Methods with High Linear Order, *J. Sci. Comput.*, **73** (2017), pp. 667-690.

[45] J. Coughlin and J. Hu, Efficient dynamical low-rank approximation for the Vlasov-Ampere-Fokker-Planck system, *J. Comput. Phys.*, **470** (2022), pp. 111590.

[46] N. Crouseilles, M. Mehrenberger, and E. Sonnendrucker, Conservative semi-Lagrangian schemes for Vlasov equations, *J. Comput. Phys.*, **229:6** (2010), pp. 1927-1953.

[47] J. Dawson, Particle simulation of plasmas, *Rev. Mod. Phys.*, **55:2** (1983), pp. 403.

[48] L. De Lathauwer, B. De Moor, and J. Vandewalle, A multilinear singular value decomposition, *SIAM J. Matrix Anal. Appl.*, **21:4** (2000), pp. 1253-1278.

[49] P. Degond and B. Lucquin-Desreux, An entropy scheme for the Fokker-Planck collision operator of plasma kinetic theory, *Numer. Math.*, **68** (1994), pp. 239-262.

[50] P. Degond and B. Lucquin-Desreux, The Fokker-Planck asymptotics of the Boltzmann collision operator in the Coulomb case, *Math. Models Meth. Appl. Sci.*, **2:2** (1992), pp. 167-182.

[51] P. Degond, et al., Asymptotic-preserving particle-in-cell method for the Vlasov-Poisson system near equilibrium, *J. Comput. Phys.*, **229:16** (2010), pp. 5630-5652.

[52] A. Dektor and D. Venturi, Dynamically orthogonal tensor methods for high-dimensional nonlinear pdes, *J. Comput. Phys.*, **404** (2020), pp. 109125.

[53] L. Desvillettes, Plasma kinetic models: the Fokker-Planck-Landau equation, *In: Modeling and Computational Methods for Kinetic Equations, Model. Simul. Sci. Eng. Technol.*, Birkhauser Boston, Boston, MA, 2004.

[54] L. Desvillettes and C. Villano, On the trend to global equilibrium in spatially inhomogeneous entropy-dissipating systems: the linear Fokker-Planck equation, *Comm. Pure Appl. Math.*, **54:1** (2001), pp. 1-42.

[55] M.-C. Ding, X. Cai, W. Guo, and J.-M. Qiu, A semi-Lagrangian discontinuous Galerkin (DG)-local DG method for solving convection-diffusion equations, *J. Comput. Phys.*, **409** (2020), pp. 109295.

[56] M.-C. Ding, J.-M. Qiu, and R. Shu, Accuracy and stability analysis of the semi-Lagrangian method for stiff hyperbolic relaxation systems and kinetic BGK model, *Multi. Model. and Simul.*, **21:2** (2023), pp. 143-167.

[57] S.V. Dolgov, B.N. Khoromskij, and I.V. Oseledets, Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker-Planck equation, *SIAM J. Sci. Comput.*, **34:6** (2012), pp. A3016-A3038.

[58] J. Donea, A. Huerta, J.-Ph. Ponthot, and A. Rodriguez-Ferran, Arbitrary Lagrangian-Eulerian Methods, Chapter 14 in *The Encyclopedia of Computational Mechanics, Volume 1*, Wiley (2004), pp. 413-437.

[59] J. Douglas, Jr., On the numerical integration of $u_{xx}+u_{yy} = u_t$ by implicit methods, *SIAM*, **3:1** (1955), pp. 42-65.

[60] D.A. Dunavant, High degree efficient symmetrical Gaussian quadrature rules for the triangle, *Int. J. for Num. Meth. in Engin.*, **21** (1985), pp. 1129-1148.

[61] G. Eckart and G. Young, The approximation of one matrix by another of lower rank, *Psychometrika*, **1** (1936), pp. 211-218.

[62] V. Ehrlacher and D. Lombardi, A dynamical adaptive tensor method for the Vlasov-Poisson system, *J. Comput. Phys.*, **339** (2017), pp. 285-306.

[63] L. Einkemmer, J. Hu and Y. Wang, An asymptotic-preserving dynamical low-rank method for the multi-scale multi-dimensional linear transport equation, *J. Comput. Phys.*, **439** (2021), pp. 110353.

[64] L. Einkemmer and I. Joseph, A mass, momentum, and energy conservative dynamical low-rank scheme for the Vlasov equation, *J. Comput. Phys.*, **443** (2021), pp. 110495.

[65] L. Einkemmer and C. Lubich, A low-rank projector-splitting integrator for the Vlasov-Poisson equations, *SIAM J. Sci. Comput.*, **40:5** (2018), pp. B1330-B1360.

[66] L. Einkemmer and C. Lubich, A quasi-conservative dynamical low-rank algorithm for the Vlasov equation, *SIAM J. Sci. Comput.*, **41:5** (2019), pp. B1061-1081.

[67] E.M. Epperlein, Implicit and conservative difference scheme for the Fokker-Planck equation, *J. Comput. Phys.*, **112** (1994), pp. 291-297.

[68] M.W. Evans, F.H. Harlow, and E. Bromberg, The particle-in-cell method for hydrodynamics calculations, *Los Alamos National Lab NM*, (1957).

[69] F. Filbet and T. Rey, A rescaling velocity method for dissipative kinetic equations. Applications to granular media, *J. Comput. Phys.*, **248** (2013), pp. 177-199.

[70] F. Filbet and L.M. Rodrigues, Asymptotically stable particle-in-cell methods for the Vlasov-Poisson system with a strong external magnetic field, *SIAM J. Numer. Anal.*, **54:2** (2016), pp. 1120-1146.

[71] F. Filbet and E. Sonnendrücker, Comparison of Eulerian Vlasov solvers, *Comp. Phys. Comm.*, **150:3** (2001), pp. 247-266.

[72] F. Filbet, E. Sonnendrücker, an P. Bertrand, Conservative numerical schemes for the Vlasov equation, *J. Comput. Phys.*, **172:1** (2001), pp. 166-187.

[73] E. Forest and R.D. Ruth, Fourth-order symplectic integration, *Phys. D Nonlinear Phenom.*, **43** (1990), pp. 105-117.

[74] D. Goldman and TJ. Kaper, Nth-order operator splitting schemes and nonreversible systems, *SIAM J. Numer. Anal.*, **33:1** (1996), pp. 349-367.

[75] G.H. Golub, S. Nash, and C.F. Van Loan, A Hessenberg-Schur method for the problem AX+XB=C, *IEEE Transactions on Automatic Control*, **24:6** (1979), pp. 909-913.

[76] G.H. Golub and C.F. Van Loan, Matrix Computations, *JHU Press*, 2013.

[77] S. Gottlieb, C.-W. Shu, and E. Tadmor, Strong stability-preserving high-order time discretization methods, *SIAM Review*, **43:1** (2001), pp. 89-112.

[78] T. Goudon, On Boltzmann equations and Fokker-Planck asymptotics: Influence of grazing collisions, *J. Stat. Phys.*, **89:3/4** (1997), pp. 751-776.

[79] L. Grasedyck, Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure, *Computing*, **72:3** (2004), pp. 247-265.

[80] L. Grasedyck, Hierarchical singular value decomposition of tensors, *SIAM J. Matrix Anal. Appl.*, **31:4** (2010), pp. 2029-2054.

[81] L. Grasedyck and W. Hackbusch, Construction and arithmetics of hierarchical matrices, *Computing*, **70** (2003), pp. 295-334.

[82] L. Grasedyck, D. Kressner, and C. Tobler, A literature survey of low-rank tensor approximation techniques, *GAMM-Mitteilungen*, **36:1** (2013), pp. 53-78.

[83] Y.N. Grigoryev, V.A. Vshivkov, and M.P. Fedoruk, Numerical "Particle-In-Cell" Methods, *VSP: Utrech*, Boston, 2002.

[84] W. Guo and J.-M. Qiu, A conservative low rank tensor method for the Vlasov dynamics, *arXiv preprint* (2022), arXiv:2201.10397.

[85] W. Guo and J.-M. Qiu, A Local Macroscopic Conservative (LoMaC) low rank tensor method for the Vlasov dynamics, *arXiv preprint* (2022), arXiv:2207.00518.

[86] W. Guo and J.-M. Qiu, A low rank tensor representation of linear transport and nonlinear Vlasov solutions and their associated flow maps, *J. Comput. Phys.*, **458** (2022), pp. 111089.

[87] D.A. Gurnett and A. Bhattacharjee, Introduction to plasma physics: with space, laboratory and astrophysical applications, 2nd edn, *Cambridge Univ. Press*, 2017.

[88] W. Hackbusch, Tensor schemes and numerical tensor calculus, *Springer Berlin, Vol. 42*, 2012.

[89] W. Hackbusch and S. Kühn, A new scheme for the tensor representation, *J. Fourier Anal. Appl.*, **15:5** (2009), pp. 706-722.

[90] N. Halko, P.G. Martinsson, and J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, *SIAM Review*, **53:2** (2011), pp. 217-288.

[91] R.D. Hazeltine and J.D. Meiss, Plasma confinement, *Addison-Wesly Publishing Company*, Redwood City, CA, USA, 1991.

[92] I. Higueras, N. Happenhofer, O. Koch, and F. Kupka, Optimized strong stability preserving IMEX Runge-Kutta methods, *J. Comput. and Appl. Math.*, **272** (2014), pp. 116-140.

[93] S.P. Hirschman and D.J. Sigmar, Approximate Fokker-Planck collision operator for transport theory applications, *The Physics of Fluids*, **19:10** (1976), pp. 1532-1540.

[94] C. Hirt, A. Amsden, and J. Cook, An arbitrary lagrangian eulerian computing method for all flow speeds, *J. Comput. Phys.*, **14** (1974), pp. 227253.

[95] R.W. Hockney and J.W. Eastwood, Computer Simulation Using Particles, *Taylor & Francis Inc.*, USA, January 1988.

[96] J. Hu, R. Shu, and X. Zhang, Asymptotic-preserving and positivity-preserving implicit-explicit schemes for the stiff BGK equation, *SIAM J. Numer. Anal.*, **56:2** (2018), pp. 942-973.

[97] J. Hu and X. Zhang, Positivity-preserving and energy-dissipative finite difference schemes for the Fokker-Planck and Keller-Segel equations, *arXiv preprint* (2021), arXiv:2103.16790.

[98] C.-S. Huang, T. Arbogast, and J. Qiu, An Eulerian-Lagrangian WENO finite volume scheme for advection problems, *J. Comput. Phys.*, **231** (2012), pp. 4028-4052.

[99] C.-S. Huang and T. Arbogast, An Eulerian-Lagrangian Weighted Essentially Nonoscillatory scheme for Nonlinear Conservation Laws, *Num. Meth. for Part. Diff. Eq.*, **33:3** (2017), pp. 651-680.

[100] F. Huot, A. Ghizzo, P. Bertrand, E. Sonnendrucker, and O. Couland, Instability of the time splitting scheme for the one-dimensional and relativistic Vlasov-Maxwell system, *J. Comput. Phys.*, **185:2** (2003), pp. 512-531.

[101] E. Isaacson and H.B. Keller, Analysis of Numerical Methods, *Wiley*, New York, 1966.

[102] D. Jarema, et al., Block-structured grids for Eulerian gyrokinetic simulations, *Comp. Phys. Comm.*, **198** (2016), pp. 105-117.

[103] S. Jin and L. Wang, An asymptotic preserving scheme for the Vlasov-Poisson-Fokker-Planck system in the high field regime, *Acta Math. Sci.*, **31B:6** (2011), pp. 2219-2232.

[104] S. Jin and B. Yan, A class of asymptotic-preserving schemes for the Fokker-Planck-Landau equation, *J. Comput. Phys.*, **230** (2011), pp. 6420-6437.

[105] E. Kieri, C. Lubich, and H. Walach, Discretized dynamical low-rank approximation in the presence of small singular values, *SIAM J. Numer. Anal.*, **54:2** (2016), pp. 1020-1038.

[106] H.A.L. Kiers, Towards a standardized notation and terminology in multiway analysis, *J. Chemometrics*, **14** (2000), pp. 105-122.

[107] O. Koch and C. Lubich, Dynamical low-rank approximation, *SIAM J. Matrix Anal. Appl.*, **29:2** (2007), pp. 434-454.

[108] O. Koch and C. Lubich, Dynamical tensor approximation, *SIAM J. Matrix Anal. Appl.*, **31:5** (2010), pp. 2360-2375.

[109] T.G. Kolda and B.W. Bader, Tensor Decompositions and Applications, *SIAM Review*, **51:3** (2009), pp. 455-500.

[110] J. Koo, R. Martin, and E.M. Sousa, High fidelity modeling of field reserved configuration (FRC) thrusters, *AFRL/RQRS*, (2017).

[111] K. Kormann, A semi-Lagrangian Vlasov solver in tensor train format, *SIAM J. Sci. Comput.*, **37:4** (2015), pp. B613-B632.

[112] K. Kormann, Low-rank tensor discretization for high-dimensional problems, *Vorlesung* (2017).

[113] D. Kressner and C. Tobler, Krylov subspace methods for linear systems with tensor product structure, *SIAM J. Matrix Anal. Appl.*, **31:4** (2010), pp. 1688-1714.

[114] P.K. Kundu, I.M. Cohen, and D.R. Dowling, Fluid mechanics, 6th edn, *Academic Press*, 2015.

[115] O. Larroche, Kinetic simulations of fuel ion transport in ICF target implosions, *Eur. Phys. J. D*, **27** (2003), pp. 131-146.

[116] E.W. Larsen, C.D. Levermore, G. Pomraning, and J.G. Sanderson, Discretization methods for one-dimensional Fokker-Planck operators, *J. Comput. Phys.*, **61** (1985), pp. 359-390.

[117] R.J. Leveque, High-resolution conservative algorithms for advection in incompressible flow, *SIAM J. Numer. Anal.*, **33:2** (1996), pp. 627–665.

[118] R.J. Leveque, Numerical methods for conservation laws, *Basel: Birkhäuser, Vol. 214*, 1992.

[119] R.J. Leveque, Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems, *SIAM*, 2007.

[120] D. Levy, G. Puppo, and G. Russo, Central WENO schemes for hyperbolic systems of conservation laws, *ESAIM: Math. Model. and Numer. Anal.*, **33:3** (1999), pp. 547-571.

[121] L. Li, J. Qiu, and G. Russo, A High-Order Semi-Lagrangian Finite Difference Method for Nonlinear Vlasov and BGK Models. *Comm. on Applied Math. and Comput.*, (2022), pp. 1-29.

[122] R. Li, T. Tang, and P.W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, *J. Comput. Phys.*, **170** (2001), pp. 562-588.

[123] S. Li and L. Petzold, Moving mesh methods with upwinding schemes for time-dependent PDEs, *J. Comput. Phys.*, **131** (1997), pp. 368-377.

[124] C. Lubich, Time integration in the multiconfiguration time-dependent Hartree method of molecular quantum dynamics, *Appl. Math. Res. Express.*, AMRX **2015:2** (2015), pp. 311-328.

[125] C. Lubich and I.V. Oseledets, A projector-splitting integrator for dynamical low-rank approximation, *BIT Numer. Math.*, **54** (2014), pp. 171-188.

[126] C. Lubich, I.V. Oseledets, and B. Vandereycken, Time integration of tensor trains, *SIAM J. Numer. Anal.*, **53:2** (2015), pp. 917-941.

[127] C. Lubich, B. Vandereycken, and H. Walach, Time integration of rank-constrained Tucker tensors, *SIAM J. Numer. Anal.*, **56:3** (2018), pp. 1273-1290.

[128] C. Lubich, et al., Dynamical approximation by hierarchical Tucker and tensor-train tensors, *SIAM J. Matrix Anal. Appl.*, **34:2** (2013), pp. 470-494.

[129] D. Luo, W. Huang, and J. Qiu, A quasi-Lagrangian moving mesh discontinuous Galerkin method for hyperbolic conservation laws, *J. Comput. Phys.*, **396** (2019), pp. 544-578.

[130] M.E. Mincsovics and T. Kalmar-Nagy, Splitting headache: How well do splitting methods preserve stability?, *Int. J. of Non. Mech.*, **149** (2023), pp. 104309.

[131] M. Mohammadi and A. Borzi, Analysis of the Chang-Cooper discretization scheme for a class of Fokker-Planck equations, *J. Numer. Math.*, **23:3** (2015), pp. 271-288.

[132] D.C. Montgomery and D.A. Tidman, Plasma Kinetic Theory, *McGraw Hill*, 1964.

[133] J. Nakao, J. Chen, and J.-M. Qiu, An Eulerian-Lagragian Runge-Kutta finite volume (EL-RK-FV) method for solving convection and convection-diffusion equations, *J. Comput. Phys.*, **470** (2022), pp. 111589.

[134] J. Nieto, F. Poupaud, and J. Soler, High-field limit for the Vlasov-Poisson-Fokker-Planck system, *Archive for Rational Mechanics and Analysis*, **158** (2001), pp. 29-59.

[135] I.V. Oseledets, Tensor-train decomposition, *SIAM J. Sci. Comput.*, **33:5** (2011), pp. 2295-2317.

[136] I.V. Oseledets and S.V. Dolgov, Solution of linear systems and matrix inversion in the TT-format, *SIAM J. Sci. Comput.*, **34:5** (2012), pp. A2718-A2739.

[137] E.E. Papalexakis, et al., Large scale tensor decompositions: algorithmic developments and applications', *IEEE Data Eng. Bull.*, **36:3** (2013), pp. 59-66.

[138] L. Pareschi and G. Russo, Implicit-explicit Runge-Kutta schemes and applications to hyperbolic systems with relaxation, *J. Sci. Comput.*, **25** (2005), pp. 129-155.

[139] L. Pareschi and M. Zanella, Structure preserving schemes for nonlinear Fokker-Planck equations and applications, *J. Sci. Comput.*, **74:3** (2018), pp. 1575-1600.

[140] D.W. Peaceman and H.H. Rachford, Jr., The numerical solution of parabolic and elliptic differential equations, *SIAM*, **3:1** (1955), pp. 28-41.

[141] J.S. Peery and D.E. Carroll, Multi-material ale methods in unstructured grids, *Comp. Meth. in App. Mech. and Eng.*, **187** (2000), pp. 591-619.

[142] J.-M. Qiu and A. Christlieb, A conservative high order semi-Lagrangian WENO method for the Vlasov equation, *J. Comput. Phys.*, **229:4** (2010), pp. 1130-1149.

[143] J.-M. Qiu and C.-W. Shu, Conservative high order semi-Lagrangian finite difference WENO methods for advection in incompressible flow, *J. Comput. Phys.*, **230** (2011), pp. 863-889.

[144] J.-M. Qiu and C.-W. Shu, Positivity preserving semi-Lagrangian discontinuous Galerkin formulation: Theoretical analysis and application to the Vlasov-Poisson system, *J. Comput. Phys.*, **230** (2011), pp. 8386-8409.

[145] A. Quarteroni, R. Sacco, and F. Saleri, Numerical mathematics, *Springer Science and Business Media, Vol. 37*, 2010.

[146] S. Rabanser, O. Shchur, and S. Günnemann, Introduction to tensor decompositions and their applications in machine learning, *arXiv preprint* (2018), arXiv:1711.10781.

[147] A. Ralston and P. Rabinowitz, A first course in numerical analysis, *McGraw-Hill, 2nd edition*, New York, 1978.

[148] C.P. Ridgers, R.J. Kingham, and A.G.R. Thomas, Magnetic cavitation and the reemergence of nonlocal transport in laser plasmas, *Phys. Rev. Lett.*, **100** (2008), pp. 075003.

[149] A. Rodgers, A. Dektor, and D. Venturi, Adaptive integration of nonlinear evolution equations on tensor manifolds, *J. Sci. Comput.*, **92:2** (2022), pp. 39.

[150] A. Rodgers and D. Venturi, Implicit step-truncation integration of nonlinear PDEs on low-rank tensor manifolds, *arXiv preprint* (2022), arXiv:2207.01962.

[151] J.A. Rossmanith and D.C. Seal, A positivity-preserving high-order semi-Lagrangian discontinuous Galerkin scheme for the Vlasov-Poisson equations, *J. Comput. Phys.*, **230** (2011), pp. 6203-6232.

[152] T.F. Russell and M.A. Celia, An overview of research on Eulerian-Lagrangian localized adjoint methods (ELLAM), *Adv. Water Resour.* **25** (2002), pp. 1215-1231.

[153] M. Seydaoglu, U. Erdogan, and T. Ozis, Numerical solution of Burgers' equation with high order splitting methods, *J. Comput. and Appl. Math.*, **291** (2016), pp. 410-421.

[154] T. Shi and A. Townsend, On the compressibility of tensors, *SIAM J. Matrix Anal. Appl.*, **42:1** (2021), pp. 275-298.

[155] T. Shiroto and Y. Sentoku, Structure-preserving strategy for conservative simulation of the relativistic nonlinear Landau-Fokker-Planck equation, *Phys. Rev. E*, **99:5** (2019), pp. 053309.

[156] E. Sonnendrücker, et al., The semi-Lagrangian method for the numerical resolution of the Vlasov equation, *J. Comput. Phys.*, **149:2** (1999), pp. 201-222.

[157] C.-W. Shu, High Order Weighted Essentially Nonoscillatory Schemes for Convection Dominated Problems, *SIAM Review*, **51:1** (2009), pp. 82-126.

[158] C.-W. Shu, Total-variation-diminishing time discretizations, *SIAM J. Sci. Statist. Comput.*, **9:6** (1988), pp. 1073-1084.

[159] C.-W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, *J. Comput. Phys.*, **77:2** (1988), pp. 439-471.

[160] F. Stenger, Numerical methods based on Sinc and analytic functions, *Springer*, New York, 1993.

[161] J.M. Stockie, J.A. Mackenzie, and R.D. Russell, A moving mesh method for one-dimensional hyperbolic conservation laws, *SIAM J. Sci. Comput.*, **22** (2001), pp. 1791-1813.

[162] W.T. Taitano, et al., A conservative phase-space moving-grid strategy for a 1D-2V Vlasov-Fokker-Planck solver, *Comp. Phys. Comm.*, **258** (2021), pp. 107547.

[163] W.T. Taitano, et al., A mass, momentum, and energy conserving, fully implicit, scalable algorithm for the multi-dimensional, multi-species Rosenbluth-Fokker-Planck equation, *J. Comput. Phys.*, **297** (2015), pp. 357-380.

[164] W.T. Taitano, et al., An Eulerian Vlasov-Fokker-Planck algorithm for spherical implosion simulations of intertial confinement fusion capsules, *Comp. Phys. Comm.*, **263** (2021), pp. 107861.

[165] W.T. Taitano and L. Chacon, Charge-and-energy conserving moment-based accelerator for a multi-species Vlasov-Fokker-Planck-Ampere system, part I: collisionless aspects, *J. Comput. Phys.*, **284** (2015), pp. 718-726.

[166] W.T. Taitano, L. Chacon, and A.N. Simakov, An equilibrium-preserving discretization for the nonlinear Rosenbluth-Fokker-Planck operator in arbitrary multi-dimensional geometry, *J. Comput. Phys.*, **339** (2017), pp. 458-460.

[167] W.T. Taitano, D.A. Knoll, and L. Chacon, Charge-and-energy conserving moment-based accelerator for a multi-species Vlasov-Fokker-Planck-Ampere system, part II: collisional aspects, *J. Comput. Phys.*, **284** (2015), pp. 737-757.

[168] H. Tang and T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, *SIAM J. Numer. Anal.*, **41:2** (2003), pp. 487-515.

[169] A.G.R. Thomas, et al., A review of Vlasov-Fokker-Planck numerical modeling of inertial confinement fusion plasma, *J. Comput. Phys.*, **231:3** (2012), pp. 1051-1079.

[170] G. Toscani, Entropy production and the rate of convergence to equilibrium for the Fokker-Planck equation, *Quar. Appl. Math.*, **57:3** (1999), pp. 521-541.

[171] L.N. Trefethen, Is Gauss quadrature better than Clenshaw-Curtis?, *SIAM Review*, **50:1** (2008), pp. 67-87.

[172] L.N. Trefethen and D. Bau, Numerical linear algebra, *SIAM*, Philadelphia, PA, 1997.

[173] D. Tskhakaya, K. Matyash, R. Schneider, and F. Taccogna, The Particle-In-Cell Method, *Contributions to Plasma Physics*, **47:8-9** (2007), pp. 563-594.

[174] L.R. Tucker, Some mathematical notes on three-mode factor analysis, *Psychometrika*, **31:3** (1966), pp. 279-311.

[175] M. Udell and A. Townsend, Why are big data matrices approximately low rank?, *SIAM J. Math. Data Sci.*, **1:1** (2019), pp. 144-160.

[176] S. Van Huffel, et al., Development of high performance numerical software for control, *IEEE Control systems Magazine*, **24:1** (2004), pp. 60-76.

[177] H. Wang, S. Wang, Q. Zhang, and C.-W. Shu, Local discontinuous Galerkin methods with implicit-explicit time-marching for multi-dimensional convection-diffusion problems, *ESAIM: Math. Model. and Numer. Anal.*, **50:4** (2016), pp. 1083-1105.

[178] T. Xiong, G. Russo, and J.-M. Qiu, Conservative Multi-dimensional Semi-Lagrangian Finite Difference Scheme: Stability and Applications to the Kinetic and Fluid Simulations, *J. Sci. Comp.*, **79** (2019), pp. 1241-1270.

[179] D. Xiu and G.E. Karniadakis, A semi-Lagrangian high-order method for Navier-Stokes equations, *J. Comput. Phys.*, **172** (2001), pp. 658-684.

[180] Y. Yang, J. Chen, and J.-M. Qiu, Stability analysis of the Eulerian-Lagrangian finite volume methods for nonlinear hyperbolic equations in one space dimension, *arXiv preprint* (2023), arXiv:2302.07291.

[181] E.S. Yoon and C.S. Chang, A Fokker-Planck-Landau collision equation solver on two-dimensional velocity grid and its application to particle-in-cell simulation, *Phys. Plasmas*, **21** (2014), pp. 032503.

[182] H. Yoshida, Construction of higher order symplectic integrators, *Phys. Lett. A*, **150** (1990), pp. 262-268.

[183] M. Zennaro, Natural continuous extensions of Runge-Kutta methods, *Math Comp.*, **46** (1986), pp. 119-133.

[184] J. Zhu and J. Qiu, A new third order finite volume weighted essentially non-oscillatory scheme on tetrahedral meshes, *J. Comput. Phys.*, **349** (2017), pp. 220-232.

[185] J. Zhu and J. Qiu, A New Type of Finite Volume WENO Schemes for Hyperbolic Conservation Laws, *J. Sci. Comput.*, **73** (2017), pp. 1338-1359.

[186] J. Zhu and C.-W. Shu, A new type of multi-resolution WENO schemes with increasingly higher order of accuracy on triangular meshes, *J. Comput. Phys.*, **392** (2019), pp. 19-33.

[187] J. Zhu and J. Qiu, A new fifth order finite difference WENO scheme for solving hyperbolic conservation laws, *J. Comput. Phys.*, **318** (2016), pp. 110-121.

[188] J. Zhu and J. Qiu, New finite volume weighted essentially nonoscillatory schemes on triangular meshes, *SIAM J. Sci. Comput.*, **40:2** (2018), pp. A903-A928.

Appendix A

AN ILLUSTRATIVE EXAMPLE WITH IMEX(2,2,2)

In this section, we couple the EL-RK-FV algorithm with IMEX(2,2,2), that is, two-stage implicit, two-stage explicit, and of combined order two. This scheme is L-stable and uses a second order DIRK method. Figure A.1 shows the lone sub-space-time region ${}_1\Omega_j$.

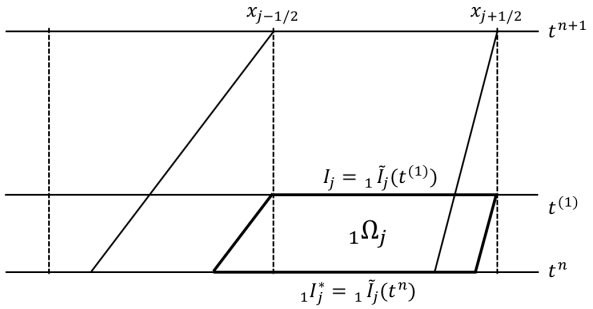


Figure A.1: The space-time region ${}_1\Omega_j$ for IMEX(2,2,2).

Step 0a. Compute the approximate characteristic speeds using equation (2.2.2). After defining the space-time region Ω_j, compute the possibly nonuniform traceback cell averages $\tilde{u}_j(t^n)$ using Algorithm 2.1.

Step 0b. Use the possibly nonuniform traceback cell averages $\tilde{u}_j(t^n)$ in Algorithm 2.2 to compute $\hat{K}_1 = \mathcal{F}(U^n; t^n))$.

Step 1a. Using the same approximate characteristic speeds from step 1a, define the sub-space-time region ${}_1\Omega_j$ as seen in Figure A.1. Compute the possibly nonuniform traceback cell averages ${}_1\tilde{u}_j^n$ using Algorithm 2.1.

Step 1b. Use the possibly nonuniform traceback cell averages ${}_1\tilde{u}_j^n$ in Algorithm 2.2 to compute ${}_1\hat{K}_1 = \mathcal{F}({}_1U^n; t^n))$.

Step 1c. Recalling equation (2.3.4), solve equation (2.3.10) by solving the linear system

$$\left(\mathbf{I} - \frac{\gamma\epsilon\Delta t}{\Delta x^2}\mathbf{D}_4\right) {}_1\vec{U}^{(1)} = {}_1\vec{U}^{n} + \gamma\Delta t_1\vec{\hat{K}}_1 + \gamma\Delta t\,\vec{g}(x, t^{(1)}), \tag{A.0.1}$$

where $\vec{g}_j(x, t^{(1)}) = \int_{I_j} g(x, t^{(1)})dx$ can be computed with a Gaussian quadrature.

Step 1d. Compute the uniform cell averages $\overline{u}_j^{(1)} = {}_1U_j^{(1)}/\Delta x$.

Step 1e. Compute the uniform cell averages $\overline{u}_{xx,j}^{(1)}$ using equation (2.3.4),

$$\vec{\overline{u}}_{xx}^{(1)} = \frac{1}{\Delta x^2}\mathbf{D}_4\vec{\overline{u}}^{(1)}. \tag{A.0.2}$$

Step 1f. Compute the possibly nonuniform traceback cell averages $\tilde{u}_j^{(1)}$ and $\tilde{u}_{xx,j}^{(1)}$ (we are now in the space-time region Ω_j) using Algorithm 2.1.

Step 1g. Compute $K_1 = \mathcal{G}(U^{(1)}; t^{(1)})$,

$$K_1 = \epsilon\Delta\tilde{x}_j^{(1)}\tilde{u}_{xx,j}^{(1)} + \int_{\tilde{I}_j(t^{(1)})} g(x, t^{(1)})dx, \tag{A.0.3}$$

where the definite integral involving $g(x, t)$ can be evaluated using a Gaussian quadrature.

Step 1h. Use the possibly nonuniform traceback cell averages $\tilde{u}_j^{(1)}$ in Algorithm 2.2 to compute $\hat{K}_2 = \mathcal{F}(U^{(1)}; t^{(1)})$.

Step 2. Recalling equation (2.3.4), solve equation (2.3.7a) by solving the linear system

$$\left(\mathbf{I} - \frac{\gamma\epsilon\Delta t}{\Delta x^2}\mathbf{D}_4\right)\vec{U}^{n+1} = \vec{U}^{n} + (1-\gamma)\Delta t\vec{\hat{K}}_1 + \Delta t(\delta\vec{\hat{K}}_1 + (1-\delta)\vec{\hat{K}}_2) + \gamma\Delta t\,\vec{g}(x, t^{n+1}), \tag{A.0.4}$$

where $\vec{g}_j(x, t^{n+1}) = \int_{I_j} g(x, t^{n+1})dx$ can be computed with a Gaussian quadrature.

Appendix B

THE SECOND-ORDER SCHEME WITH CRANK-NICOLSON

We treat Crank-Nicolson as a two stage integrator similar to the stiffly-accurate DIRK2. The first stage is simply the first-order backward Euler approximation. The second stage is the second-order Crank-Nicolson approximation. In this way, the proposed second-order scheme using Crank-Nicolson is nearly identical to the second-order scheme using stiffly-accurate DIRK2 in Algorithm 3.2. As such, we omit the finer details and outline the main equations.

The first-order backward Euler method and second-order Crank-Nicolson method are respectively

$$U^{(1)} = U^n + \Delta t \mathcal{L}(U^{(1)}; t^{(1)}), \tag{B.0.1a}$$

$$U^{n+1} = U^n + \frac{\Delta t}{2}\mathcal{L}(U^n; t^n) + \frac{\Delta t}{2}\mathcal{L}(U^{n+1}; t^{n+1}), \tag{B.0.1b}$$

where $\mathcal{L}(U;t) = (d_1^2\partial_x^2 + d_2^2\partial_y^2)U$ and $t^{(1)} = t^{n+1}$.

$K-L-S$ phase 1

The first stage is the backward Euler integrator over a full time-step. Following Algorithm 3.1, we obtain the low-rank solution $\mathbf{U}^{(1)} = \mathbf{V}^{x,(1)}\mathbf{S}^{(1)}(\mathbf{V}^{y,(1)})^T$ of rank $r^{(1)}$.

$K-L-S$ phase 2: K and L steps

Discretizing equation (3.2.2) using Crank-Nicolson,

$$\begin{aligned}\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T - \frac{\Delta t}{2}\mathbf{D}^x\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T \\ - \frac{\Delta t}{2}\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{D}^y\mathbf{V}^{y,n+1})^T = RHS,\end{aligned} \tag{B.0.2a}$$

$$RHS = \mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T + \frac{\Delta t}{2}\mathbf{D}^x\mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{V}^{y,n})^T + \frac{\Delta t}{2}\mathbf{V}^{x,n}\mathbf{S}^n(\mathbf{D}^y\mathbf{V}^{y,n})^T. \quad \text{(B.0.2b)}$$

The approximate bases 3.2.5 used in the first-order scheme will not suffice since the $\mathcal{O}(\Delta t)$ error will destroy the desired second-order accuracy. Consider the augmented bases

$$\begin{bmatrix}\mathbf{V}^{x,n} & | & \mathbf{V}^{x,(1)}\end{bmatrix} \in \mathbb{R}^{N_x \times (r^n + r^{(1)})}, \quad \begin{bmatrix}\mathbf{V}^{y,n} & | & \mathbf{V}^{y,(1)}\end{bmatrix} \in \mathbb{R}^{N_y \times (r^n + r^{(1)})}.$$

Computing the reduced SVDs of the augmented bases, let $\mathbf{U}^x$ be the left singular vectors of $\left[\mathbf{V}^{x,n}|\mathbf{V}^{x,(1)}\right]$, and $\mathbf{U}^y$ be the left singular vectors of $\left[\mathbf{V}^{y,n}|\mathbf{V}^{y,(1)}\right]$. Further let r^x and r^y be the respectively number of singular values greater than the same tolerance ϵ. The approximate bases for the second stage of the second-order scheme are defined by

$$\mathbf{V}^{x,\star} \coloneqq \mathbf{U}^x(:, 1:r), \quad \text{(B.0.3a)}$$

$$\mathbf{V}^{y,\star} \coloneqq \mathbf{U}^y(:, 1:r), \quad \text{(B.0.3b)}$$

where $r = \max(r^x, r^y)$. After substituting $\mathbf{V}^{y,n+1}$ with $\mathbf{V}^{y,\star}$, *projecting* equation (B.0.2) onto the column space of $\mathbf{V}^{y,\star}$ yields the Sylvester equation

$$\left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{D}^x\right)\mathbf{K}^{n+1} + \mathbf{K}^{n+1}\left(-\frac{\Delta t}{2}(\mathbf{D}^y\mathbf{V}^{y,\star})^T\mathbf{V}^{y,\star}\right) = (RHS)\mathbf{V}^{y,\star}, \quad \text{(B.0.4)}$$

where $\mathbf{K}^n = \mathbf{V}^{x,n}\mathbf{S}^n$. Similarly, substituting $\mathbf{V}^{x,n+1}$ with $\mathbf{V}^{x,\star}$ and projecting equation (B.0.2) onto the column space of $\mathbf{V}^{x,\star}$ yields the Sylvester equation

$$\left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{D}^y\right)\mathbf{L}^{n+1} + \mathbf{L}^{n+1}\left(-\frac{\Delta t}{2}(\mathbf{D}^x\mathbf{V}^{x,\star})^T\mathbf{V}^{x,\star}\right) = (RHS)^T\mathbf{V}^{x,\star}, \quad \text{(B.0.5)}$$

where $\mathbf{L}^n = \left(\mathbf{S}^n(\mathbf{V}^{y,n})^T\right)^T = \mathbf{V}^{y,n}(\mathbf{S}^n)^T$.

Computing the reduced QR factorizations $\mathbf{K}^{n+1} = \mathbf{Q}_x\mathbf{R}_x$ and $\mathbf{L}^{n+1} = \mathbf{Q}_y\mathbf{R}_y$,

the updated bases are defined by

$$\mathbf{V}^{x,n+1} \coloneqq \mathbf{Q}_x, \qquad \mathbf{V}^{y,n+1} \coloneqq \mathbf{Q}_y.$$

Remark B.1. If the diagonalized variant of the second-order scheme is desired, then the matrices $\mathbf{I} - \frac{\Delta t}{2}\mathbf{D}^x$ and $\mathbf{I} - \frac{\Delta t}{2}\mathbf{D}^y$ will need to be diagonalized. Instead of solving equation (B.0.6), the second stage of the second-order scheme will solve

$$\mathbf{Z}^x\tilde{\mathbf{K}}^{n+1} + \tilde{\mathbf{K}}^{n+1}\left(-\frac{\Delta t}{2}(\mathbf{D}^y\mathbf{V}_1^{y,\star})^T\mathbf{V}_1^{y,\star}\right) = (\mathbf{W}^x)^T\left((RHS)\mathbf{V}^{y,\star}\right). \tag{B.0.6}$$

The diagonalized variant of equation (B.0.5) follows similarly.

$K-L-S$ **phase 2:** S **step**

Projecting equation (B.0.2) in both dimensions onto the column spaces of the updated bases $\mathbf{V}^{x,n+1}$ and $\mathbf{V}^{y,n+1}$,

$$\begin{aligned}\mathbf{S}^{n+1} - \frac{\Delta t}{2}(\mathbf{V}^{x,n+1})^T\mathbf{D}^x\mathbf{V}^{x,n+1}\mathbf{S}^{n+1} \\ - \frac{\Delta t}{2}\mathbf{S}^{n+1}(\mathbf{D}^y\mathbf{V}^{y,n+1})^T\mathbf{V}^{y,n+1} = (\mathbf{V}^{x,n+1})^T(RHS)\mathbf{V}^{y,n+1}.\end{aligned} \tag{B.0.7}$$

Compute the eigenvalue decompositions of the real symmetric matrices

$$-\frac{\Delta t}{2}(\mathbf{V}^{x,n+1})^T\mathbf{D}^x\mathbf{V}^{x,n+1} = \mathbf{Q}^x\mathbf{\Lambda}^x(\mathbf{Q}^x)^T, \qquad -\frac{\Delta t}{2}(\mathbf{D}^y\mathbf{V}^{y,n+1})^T\mathbf{V}^{y,n+1} = \mathbf{Q}^y\mathbf{\Lambda}^y(\mathbf{Q}^y)^T.$$

Letting

$$\tilde{\mathbf{S}}^{n+1} = (\mathbf{Q}^x)^T\mathbf{S}^{n+1}\mathbf{Q}^y, \tag{B.0.8a}$$

$$\tilde{\mathbf{B}} = (\mathbf{V}^{x,n+1}\mathbf{Q}^x)^T(RHS)\mathbf{V}^{y,n+1}\mathbf{Q}^y, \tag{B.0.8b}$$

equation (B.0.7) becomes the Sylvester equation

$$(\mathbf{I} + \mathbf{\Lambda}^x)\tilde{\mathbf{S}}^{n+1} + \tilde{\mathbf{S}}^{n+1}\mathbf{\Lambda}^y = \tilde{\mathbf{B}}. \tag{B.0.9}$$

Solving equation (B.0.9) has a relatively small computational complexity. The updated $\mathbf{S}^{n+1}$ is obtained by

$$\mathbf{S}^{n+1} = \mathbf{Q}^x \tilde{\mathbf{S}}^{n+1} (\mathbf{Q}^y)^T.$$

Compressing the updated solution $\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T$ is done just like in the first-order scheme. Let $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ be the SVD of $\mathbf{S}^{n+1}$ and r^{n+1} be the number of singular values larger than some small tolerance $\epsilon > 0$. Redefine the bases to be

$$\mathbf{V}^{x,n+1} \coloneqq \mathbf{V}^{x,n+1}\mathbf{U}_{:,1:r^{n+1}}, \quad \mathbf{S}^{n+1} \coloneqq \boldsymbol{\Sigma}_{1:r^{n+1},1:r^{n+1}}, \quad \mathbf{V}^{y,n+1} \coloneqq \mathbf{V}^{y,n+1}\mathbf{V}_{:,1:r^{n+1}}. \tag{B.0.10}$$

We opt not to outline the second-order algorithm via Crank-Nicolson since it follows near identically to Algorithm 3.2.

Appendix C

THE SECOND-ORDER SCHEME WITH BDF2

The second-order scheme using BDF2 follows similarly to the second-order scheme using Crank-Nicolson in Appendix B. However, BDF2 is a linear multistep method in which the solution at t^{n+1} is dependent on the solution at t^n and t^{n-1}. We use the second-order Crank-Nicolson method to initialize the solution at t^1. The subsequent steps can then update the solution from t^n to t^{n+1}. Just like the second-order scheme using Crank-Nicolson, the first $K-L-S$ phase will be the backward Euler approximation; and then the second $K-L-S$ phase will be the BDF2 approximation. Figure C.1 provides a visual of this scheme.

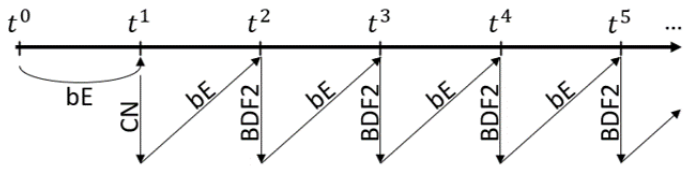


Figure C.1: The second-order scheme with BDF2.

$K-L-S$ phase 1

The first stage is the backward Euler integrator over a full time-step. Following Algorithm 3.1, we obtain the low-rank solution $\mathbf{U}^{(1)} = \mathbf{V}^{x,(1)}\mathbf{S}^{(1)}(\mathbf{V}^{y,(1)})^T$ of rank $r^{(1)}$.

$K-L-S$ phase 2: K and L steps

Discretizing equation (3.2.2) using BDF2,

$$\begin{aligned}\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T - \frac{2\Delta t}{3}\mathbf{D}^x\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T \\ - \frac{2\Delta t}{3}\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{D}^y\mathbf{V}^{y,n+1})^T = RHS,\end{aligned} \tag{C.0.1a}$$

$$RHS = \frac{4}{3}\mathbf{V}^{x,n}\mathbf{S}^{n}(\mathbf{V}^{y,n})^{T} - \frac{1}{3}\mathbf{V}^{x,n-1}\mathbf{S}^{n-1}(\mathbf{V}^{y,n-1})^{T}. \tag{C.0.1b}$$

The approximate bases 3.2.5 used in the first-order scheme will not suffice since the $\mathcal{O}(\Delta t)$ error will destroy the desired second-order accuracy. Consider the augmented bases

$$\left[\mathbf{V}^{x,n-1} \mid \mathbf{V}^{x,n} \mid \mathbf{V}^{x,(1)}\right] \in \mathbb{R}^{N_x \times (r^{n-1}+r^{n}+r^{(1)})},$$

$$\left[\mathbf{V}^{y,n-1} \mid \mathbf{V}^{y,n} \mid \mathbf{V}^{y,(1)}\right] \in \mathbb{R}^{N_y \times (r^{n-1}+r^{n}+r^{(1)})}.$$

Computing the reduced SVDs of the augmented bases, let $\mathbf{U}^x$ be the left singular vectors of $\left[\mathbf{V}^{x,n-1}|\mathbf{V}^{x,n}|\mathbf{V}^{x,(1)}\right]$, and $\mathbf{U}^y$ be the left singular vectors of $\left[\mathbf{V}^{y,n-1}|\mathbf{V}^{y,n}|\mathbf{V}^{y,(1)}\right]$. Further let r^x and r^y be the respectively number of singular values greater than the same tolerance ϵ. The approximate bases for the second stage of the second-order scheme are defined by

$$\mathbf{V}^{x,\star} := \mathbf{U}^{x}(:, 1:r), \tag{C.0.2a}$$

$$\mathbf{V}^{y,\star} := \mathbf{U}^{y}(:, 1:r), \tag{C.0.2b}$$

where $r = \max(r^x, r^y)$. After substituting $\mathbf{V}^{y,n+1}$ with $\mathbf{V}^{y,\star}$, *projecting* equation (C.0.1) onto the column space of $\mathbf{V}^{y,\star}$ yields the Sylvester equation

$$\left(\mathbf{I} - \frac{2\Delta t}{3}\mathbf{D}^{x}\right)\mathbf{K}^{n+1} + \mathbf{K}^{n+1}\left(-\frac{2\Delta t}{3}(\mathbf{D}^{y}\mathbf{V}^{y,\star})^{T}\mathbf{V}^{y,\star}\right) = (RHS)\mathbf{V}^{y,\star}, \tag{C.0.3}$$

where $\mathbf{K}^{n} = \mathbf{V}^{x,n}\mathbf{S}^{n}$. Similarly, substituting $\mathbf{V}^{x,n+1}$ with $\mathbf{V}^{x,\star}$ and projecting equation (C.0.1) onto the column space of $\mathbf{V}^{x,\star}$ yields the Sylvester equation

$$\left(\mathbf{I} - \frac{2\Delta t}{3}\mathbf{D}^{y}\right)\mathbf{L}^{n+1} + \mathbf{L}^{n+1}\left(-\frac{2\Delta t}{3}(\mathbf{D}^{x}\mathbf{V}^{x,\star})^{T}\mathbf{V}^{x,\star}\right) = (RHS)^{T}\mathbf{V}^{x,\star}, \tag{C.0.4}$$

where $\mathbf{L}^{n} = \left(\mathbf{S}^{n}(\mathbf{V}^{y,n})^{T}\right)^{T} = \mathbf{V}^{y,n}(\mathbf{S}^{n})^{T}$.

Computing the reduced QR factorizations $\mathbf{K}^{n+1} = \mathbf{Q}_x\mathbf{R}_x$ and $\mathbf{L}^{n+1} = \mathbf{Q}_y\mathbf{R}_y$,

the updated bases are defined by

$$\mathbf{V}^{x,n+1} \coloneqq \mathbf{Q}_x, \qquad \mathbf{V}^{y,n+1} \coloneqq \mathbf{Q}_y.$$

Remark C.1. If the diagonalized variant of the second-order scheme is desired, then the matrices $\mathbf{I} - \frac{2\Delta t}{3}\mathbf{D}^x$ and $\mathbf{I} - \frac{2\Delta t}{3}\mathbf{D}^y$ will need to be diagonalized. Instead of solving equation (C.0.5), the second stage of the second-order scheme will solve

$$\mathbf{Z}^x\tilde{\mathbf{K}}^{n+1} + \tilde{\mathbf{K}}^{n+1}\left(-\frac{2\Delta t}{3}(\mathbf{D}^y\mathbf{V}_1^{y,\star})^T\mathbf{V}_1^{y,\star}\right) = (\mathbf{W}^x)^T\left((RHS)\mathbf{V}^{y,\star}\right). \tag{C.0.5}$$

The diagonalized variant of equation (C.0.4) follows similarly.

$K-L-S$ **phase 2:** S **step**

Projecting equation (C.0.1) in both dimensions onto the column spaces of the updated bases $\mathbf{V}^{x,n+1}$ and $\mathbf{V}^{y,n+1}$,

$$\begin{aligned}\mathbf{S}^{n+1} - \frac{2\Delta t}{3}(\mathbf{V}^{x,n+1})^T\mathbf{D}^x\mathbf{V}^{x,n+1}\mathbf{S}^{n+1} \\ - \frac{2\Delta t}{3}\mathbf{S}^{n+1}(\mathbf{D}^y\mathbf{V}^{y,n+1})^T\mathbf{V}^{y,n+1} = (\mathbf{V}^{x,n+1})^T(RHS)\mathbf{V}^{y,n+1}.\end{aligned} \tag{C.0.6}$$

Compute the eigenvalue decompositions of the real symmetric matrices

$$-\frac{2\Delta t}{3}(\mathbf{V}^{x,n+1})^T\mathbf{D}^x\mathbf{V}^{x,n+1} = \mathbf{Q}^x\mathbf{\Lambda}^x(\mathbf{Q}^x)^T, \qquad -\frac{2\Delta t}{3}(\mathbf{D}^y\mathbf{V}^{y,n+1})^T\mathbf{V}^{y,n+1} = \mathbf{Q}^y\mathbf{\Lambda}^y(\mathbf{Q}^y)^T.$$

Letting

$$\tilde{\mathbf{S}}^{n+1} = (\mathbf{Q}^x)^T\mathbf{S}^{n+1}\mathbf{Q}^y, \tag{C.0.7a}$$

$$\tilde{\mathbf{B}} = (\mathbf{V}^{x,n+1}\mathbf{Q}^x)^T(RHS)\mathbf{V}^{y,n+1}\mathbf{Q}^y, \tag{C.0.7b}$$

equation (C.0.6) becomes the Sylvester equation

$$(\mathbf{I} + \mathbf{\Lambda}^x)\tilde{\mathbf{S}}^{n+1} + \tilde{\mathbf{S}}^{n+1}\mathbf{\Lambda}^y = \tilde{\mathbf{B}}. \tag{C.0.8}$$

Solving equation (C.0.8) has a relatively small computational complexity. The updated $\mathbf{S}^{n+1}$ is obtained by

$$\mathbf{S}^{n+1} = \mathbf{Q}^x \tilde{\mathbf{S}}^{n+1} (\mathbf{Q}^y)^T.$$

Compressing the updated solution $\mathbf{V}^{x,n+1}\mathbf{S}^{n+1}(\mathbf{V}^{y,n+1})^T$ is done just like in the first-order scheme. Let $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ be the SVD of $\mathbf{S}^{n+1}$ and r^{n+1} be the number of singular values larger than some small tolerance $\epsilon > 0$. Redefine the bases to be

$$\mathbf{V}^{x,n+1} \coloneqq \mathbf{V}^{x,n+1}\mathbf{U}_{:,1:r^{n+1}}, \quad \mathbf{S}^{n+1} \coloneqq \boldsymbol{\Sigma}_{1:r^{n+1},1:r^{n+1}}, \quad \mathbf{V}^{y,n+1} \coloneqq \mathbf{V}^{y,n+1}\mathbf{V}_{:,1:r^{n+1}}. \tag{C.0.9}$$

We opt not to outline the second-order algorithm via BDF2 since it follows near identically to Algorithm 3.2.

Remark C.2. It is common practice to define the time-step Δt in terms of the spatial mesh, e.g., $\Delta t = (CFL)\Delta x$ for some $CFL > 0$. Since the desired final time $t = T_f$ might not be an integer multiple of Δt, the final time-step is usually smaller than the other time-steps, $\Delta t^{N_t} < \Delta t$. As such, BDF2 applied to the final time-step will not output the solution at time $t = T_f$ since BDF2 is a linear multistep method that assumes $\Delta t^{n+1} = \Delta t^n$. To remedy this issue, we apply the "one step of Crank-Nicolson, and then one step of BDF2" strategy over the final time-step $[t^{N_t-1}, t^{N_t}]$ that we use in the initialization; see Figure C.1. By taking a half time-step $(t^{N_t} - t^{N_t-1})/2$ with Crank-Nicolson, and then a half time-step $(t^{N_t} - t^{N_t-1})/2$ with BDF2, the final solution will be at time $t = T_f$. Another common strategy with linear multistep methods is to interpolate the solution at the final time, but we opted not to do this since the chosen approach is already implemented in the initialization.

Appendix D

NONDIMENSIONALIZING THE 1D2V VLASOV-LEONARD-BERNSTEIN-FOKKER-PLANCK EQUATION IN CYLINDRICAL COORDINATES

We first present the *dimensional* kinetic-ion and fluid-electron model used in Chapter 4. The dimensional kinetic ion Vlasov-Leonard-Bernstein-Fokker-Planck (VLBFP) equation in cylindrical coordinates is [162]

$$\frac{\partial f_\alpha}{\partial t} + v_{||}\frac{\partial f_\alpha}{\partial x} + \frac{q_\alpha}{m_\alpha}E_{||}\frac{\partial f_\alpha}{\partial v_{||}} = C_{\alpha\alpha} + C_{\alpha e}, \tag{D.0.1a}$$

$$C_{\alpha\alpha} = \nu_{\alpha\alpha}\nabla_{\mathbf{v}}\cdot\left(\frac{T_\alpha}{m_\alpha}\nabla_{\mathbf{v}}f_\alpha + (\mathbf{v}-\mathbf{u}_\alpha)f_\alpha\right), \tag{D.0.1b}$$

$$C_{\alpha e} = \nu_{\alpha e}\nabla_{\mathbf{v}}\cdot\left(\frac{T_e}{m_\alpha}\nabla_{\mathbf{v}}f_\alpha + (\mathbf{v}-\mathbf{u}_e)f_\alpha\right), \tag{D.0.1c}$$

where f_α is the distribution function for the single ion species α, and the charge, mass, temperature, bulk velocity, and collision frequencies for the ion species and electron are respectively denoted by q, m, T, $\mathbf{u}$, and ν. The fluid-electron energy equation is [162]

$$\frac{3}{2}\frac{\partial p_e}{\partial t} + \frac{5}{2}\frac{\partial}{\partial x}\left(u_{e,||}p_e\right) - u_{e,||}\frac{\partial p_e}{\partial x} - \frac{\partial}{\partial x}\left(\kappa_{e,||}\frac{\partial T_e}{\partial x}\right) = W_{e\alpha}, \tag{D.0.2a}$$

$$W_{e\alpha} = -\left\langle \frac{m_\alpha|\mathbf{v}|^2}{2}, C_{\alpha e}\right\rangle = 3\nu_{\alpha e}n_\alpha(T_\alpha - T_e), \tag{D.0.2b}$$

where $p_e = n_eT_e$ is the electron pressure, $\kappa_{e,||}$ is the thermal conductivity, and the velocity space L^2 inner product is denoted

$$\langle F(\mathbf{v}), G(\mathbf{v})\rangle \doteq 2\pi\int_{-\infty}^{\infty}\int_0^{\infty} F(\mathbf{v})G(\mathbf{v})v_\perp dv_\perp dv_{||}. \tag{D.0.3}$$

Refer to Section 4.2 for how to get the simplified form of the second-order moment of the Leonard-Bernstein-Fokker-Planck operator. The collision frequencies and thermal conductivity are given by

$$\nu_{\alpha\alpha} = \frac{\zeta}{\sqrt{m_\alpha}} \frac{n_\alpha}{T_\alpha^{3/2}}, \tag{D.0.4a}$$

$$\frac{\nu_{\alpha e}}{\sqrt{2}} = \frac{\zeta}{\sqrt{m_\alpha}} \sqrt{\frac{m_e}{m_\alpha}} \frac{n_e}{T_e^{3/2}}, \tag{D.0.4b}$$

$$\kappa_{e,||} = \frac{1}{\sqrt{2}} \frac{3.2}{\zeta} \frac{n_\alpha}{n_e} \frac{T_e^{5/2}}{\sqrt{m_e}}, \tag{D.0.4c}$$

for some constant ζ [87, 91, 162]. To ensure discrete conservation of the zeroth, first, and second order moments, we assume quasi-neutrality $n = n_\alpha = n_e$, ambipolarity $\mathbf{u} = \mathbf{u}_\alpha = \mathbf{u}_e$, and that the electric field is determined from Ohm's law $E_{||} = \frac{1}{q_e n_e} \frac{\partial p_e}{\partial x}$. We further assume drift only occurs in the parallel direction, that is, $u_\perp = 0$.

The reference quantities are chosen for unity (i.e., $m^* = m_\alpha$ and $q^* = q_\alpha$) and listed in Table D.1. Using the reference quantities in Table D.1, the nondimensional Vlasov-Leonard-Bernstein-Fokker-Planck equation is

$$\frac{\partial \tilde{f}_\alpha}{\partial \tilde{t}} + \tilde{v}_{||} \frac{\partial \tilde{f}_\alpha}{\partial \tilde{x}} + \frac{\tilde{q}_\alpha}{\tilde{m}_\alpha} \tilde{E}_{||} \frac{\partial \tilde{f}_\alpha}{\partial \tilde{v}_{||}} = \tilde{C}_{\alpha\alpha} + \tilde{C}_{\alpha e}, \tag{D.0.5a}$$

$$\tilde{C}_{\alpha\alpha} = \tilde{\nu}_{\alpha\alpha} \tilde{\nabla}_{\tilde{\mathbf{v}}} \cdot \left(\frac{\tilde{T}_\alpha}{\tilde{m}_\alpha} \tilde{\nabla}_{\tilde{\mathbf{v}}} \tilde{f}_\alpha + (\tilde{\mathbf{v}} - \tilde{\mathbf{u}}_\alpha) \tilde{f}_\alpha \right), \tag{D.0.5b}$$

$$\tilde{C}_{\alpha e} = \tilde{\nu}_{\alpha e} \tilde{\nabla}_{\tilde{\mathbf{v}}} \cdot \left(\frac{\tilde{T}_e}{\tilde{m}_\alpha} \tilde{\nabla}_{\tilde{\mathbf{v}}} \tilde{f}_\alpha + (\tilde{\mathbf{v}} - \tilde{\mathbf{u}}_e) \tilde{f}_\alpha \right), \tag{D.0.5c}$$

and the nondimensional fluid-electron energy model is

$$\frac{3}{2} \frac{\partial \tilde{p}_e}{\partial \tilde{t}} + \frac{5}{2} \frac{\partial}{\partial \tilde{x}} \left(\tilde{u}_{e,||} \tilde{p}_e \right) - \tilde{u}_{e,||} \frac{\partial \tilde{p}_e}{\partial \tilde{x}} - \frac{\partial}{\partial \tilde{x}} \left(\tilde{\kappa}_{e,||} \frac{\partial \tilde{T}_e}{\partial \tilde{x}} \right) = 3 \tilde{\nu}_{\alpha e} \tilde{n}_\alpha (\tilde{T}_\alpha - \tilde{T}_e). \tag{D.0.6}$$

The nondimensional collision frequencies and thermal conductivity under quasi-neutrality

are

$$\tilde{\nu}_{\alpha\alpha} = \frac{\tilde{n}_\alpha}{\tilde{T}_\alpha^{3/2}} \tag{D.0.7a}$$

$$\frac{\tilde{\nu}_{\alpha e}}{\sqrt{2}} = \frac{\sqrt{\tilde{m}_e}\tilde{n}_e}{\tilde{T}_e^{3/2}} \tag{D.0.7b}$$

$$\tilde{\kappa}_{e,||} = \frac{3.2}{\sqrt{2}} \frac{\tilde{T}_e^{5/2}}{\sqrt{\tilde{m}_e}} \tag{D.0.7c}$$

Number density	$n^* = n_\alpha$
Temperature	$T^* = T_\alpha$
Mass	$m^* = m_\alpha$
Charge	$q^* = q_\alpha$
Drift velocity	$u^* = \sqrt{\frac{T^*}{m^*}}$
Time	$\tau^* = (\nu^*)^{-1} = \frac{\sqrt{m^*}(T^*)^{3/2}}{\zeta n^*}$
Length	$L^* = u^*\tau^*$
Distribution function	$f^* = \frac{n^*}{(u^*)^3}$
Electric field	$E^* = \frac{T^*}{q^* L^*}$
Thermal conductivity	$\kappa^* = \frac{(T^*)^{5/2}}{\zeta\sqrt{m^*}}$

Table D.1: Reference quantities.

Appendix E

DERIVING THE BALANCE EQUATIONS FOR TOTAL MASS, MOMENTUM AND ENERGY

We first derive equation (4.2.11),

$$2(nU)_\alpha = \frac{3n_\alpha T_\alpha}{m_\alpha} + nu_{||}^2.$$

Proof. If f_α is an arbitrary distribution with number density n_α, bulk velocity $\mathbf{u} = u_{||}\hat{\mathbf{u}}_{||} + 0\hat{\mathbf{u}}_\perp$, and temperature T_α, then $3n_\alpha T_\alpha = m_\alpha \langle f_\alpha, |\mathbf{v} - \mathbf{u}|^2 \rangle$.

$$\begin{aligned}
2(nU)_\alpha &= 2\left\langle f_\alpha, \frac{v_\perp^2 + v_{||}^2}{2} \right\rangle \\
&= \langle f_\alpha, (v_{||} - u_{||})^2 + u_{||}^2 + v_\perp^2 \rangle + \underbrace{\langle f_\alpha, 2u_{||}(v_{||} - u_{||}) \rangle}_{=0 \text{ by symmetry}} \\
&= \langle f_\alpha, (v_{||} - u_{||})^2 + (v_\perp - 0)^2 \rangle + \langle f_\alpha, u_{||}^2 \rangle \\
&= \frac{3n_\alpha T_\alpha}{m_\alpha} + nu_{||}^2.
\end{aligned}$$

□

Deriving the **zeroth-order moment** of equation (4.2.1) given in equations (4.2.9).

Proof. Taking the zeroth-order moment of equation (4.2.1),

$$\frac{\partial n}{\partial t} + \frac{\partial}{\partial x}(nu_{||}) + \frac{q_\alpha}{m_\alpha} E_{||} \underbrace{\left\langle \frac{\partial f_\alpha}{\partial v_{||}}, 1 \right\rangle}_{(*)} = \underbrace{\langle C_{\alpha\alpha}, 1 \rangle + \langle C_{\alpha e}, 1 \rangle}_{(**)}.$$

Term $(*)$ equals zero from integration by parts and assuming decay to zero at infinity. The moments in term $(**)$ equal zero by collisional particle conservation. Integrating over space,

$$0 = \frac{d}{dt}\int_{\Omega_{\mathbf{x}}} n(x,t)dx + \left[nu_{||}\right]_{\partial\Omega_{\mathbf{x}}}.$$

□

Deriving the **first-order moment** of equation (4.2.1) given in equations (4.2.9).

Proof. Taking the first-order moment of equation (4.2.1) in $v_{||}$,

$$\frac{\partial}{\partial t}(nu_{||}) + \frac{\partial S_\alpha}{\partial x} + \frac{q_\alpha}{m_\alpha}E_{||}\underbrace{\left\langle \frac{\partial f_\alpha}{\partial v_{||}}, v_{||}\right\rangle}_{(*)} = \underbrace{\langle C_{\alpha\alpha}, 1\rangle + \langle C_{\alpha e}, 1\rangle}_{(**)}.$$

Term $(*)$ equals $-n$ from integration by parts and assuming decay to zero at infinity. The moments in term $(**)$ equal zero by collisional momentum conservation. Using Ohm's law to express the electric field,

$$\frac{\partial}{\partial t}(nu_{||}) + \frac{\partial S_\alpha}{\partial x} - \frac{q_\alpha}{m_\alpha q_e}\frac{\partial p_e}{\partial x} = 0.$$

Since $p_e = nT_e$ by quasi-neutrality,

$$\frac{\partial}{\partial t}(nu_{||}) + \frac{\partial S_\alpha}{\partial x} - \frac{q_\alpha}{m_\alpha q_e}\frac{\partial}{\partial x}(nT_e) = 0.$$

Integrating over space,

$$0 = \frac{d}{dt}\int_{\Omega_{\mathbf{x}}} (nu_{||})(x,t)dx + \left[S_\alpha - \frac{q_\alpha nT_e}{q_e m_\alpha}\right]_{\partial\Omega_{\mathbf{x}}}.$$

□

Deriving the **second-order moment** of equation (4.2.1) given in equations (4.2.9).

Proof. Taking the second-order moment of equation (4.2.1),

$$\frac{\partial}{\partial t}(nU)_\alpha + \frac{\partial Q_\alpha}{\partial x} + \frac{q_\alpha}{m_\alpha}E_{||}\underbrace{\left\langle \frac{\partial f_\alpha}{\partial v_{||}}, \frac{v_{||}^2 + v_\perp^2}{2} \right\rangle}_{(*)} = \underbrace{\left\langle C_{\alpha\alpha}, \frac{v_{||}^2 + v_\perp^2}{2} \right\rangle}_{(**)} + \left\langle C_{\alpha e}, \frac{v_{||}^2 + v_\perp^2}{2} \right\rangle.$$

Term $(*)$ equals $-nu_{||}$ from integration by parts and assuming decay to zero at infinity. The moment in term $(**)$ equals zero from the energy conservation of like-species Coulomb collisions. However, the second-order moment of the ion-electron Coulomb collision is not zero (or negligible). Using Ohm's law to express the electric field, and equation (4.2.12),

$$\frac{\partial}{\partial t}(nU)_\alpha + \frac{\partial Q_\alpha}{\partial x} - \frac{q_\alpha}{m_\alpha q_e}u_{||}\frac{\partial p_e}{\partial x} = \frac{3\nu_{\alpha e}n}{m_\alpha}(T_e - T_\alpha).$$

Adding the fluid-electron energy equation (4.2.13),

$$\frac{\partial}{\partial t}\left((nU)_\alpha + \frac{3}{2}p_e\right) + \frac{\partial}{\partial x}\left(Q_\alpha + \frac{5}{2}u_{||}p_e\right) - \left(\frac{q_\alpha}{m_\alpha q_e} + 1\right)u_{||}\frac{\partial p_e}{\partial x} - \frac{\partial}{\partial x}\left(\kappa_{e,||}\frac{\partial T_e}{\partial x}\right) = \left(\frac{1}{m_\alpha} - 1\right)3\nu_{\alpha e}n(T_e - T_\alpha).$$

Since we are working with the nondimensionalized model (see Appendix D), $m_\alpha = 1$. This, along with the additional assumption that $q_\alpha = -q_e$ in our applications of interest, reduces the energy balance equation to

$$\frac{\partial}{\partial t}\left((nU)_\alpha + \frac{3}{2}p_e\right) + \frac{\partial}{\partial x}\left(Q_\alpha + \frac{5}{2}u_{||}p_e\right) - \frac{\partial}{\partial x}\left(\kappa_{e,||}\frac{\partial T_e}{\partial x}\right) = 0.$$

Integrating over space,

$$0 = \frac{d}{dt}\int_{\Omega_\mathbf{x}}\left((nU)_\alpha + \frac{3}{2}n_eT_e\right)(x,t)dx + \left[Q_\alpha + \frac{5}{2}u_{||}nT_e - \kappa_{e,||}\frac{\partial T_e}{\partial x}\right]_{\Omega_{\partial\mathbf{x}}}.$$

□

Deriving the **reduced kinetic fluxes** (4.2.7) from kinetic fluxes (4.2.6).

Proof. Assuming f_α is an arbitrary distribution function, the temperature and perpendicular temperature are

$$T_\alpha \doteq \frac{m_\alpha \langle |\mathbf{v}-\mathbf{u}|^2, f_\alpha\rangle}{3\langle 1, f_\alpha\rangle}, \qquad T_{\alpha,\perp} \doteq \frac{m_\alpha \langle v_\perp^2, f_\alpha\rangle}{2\langle 1, f_\alpha\rangle}.$$

Moreover, $T_\alpha = T_{\alpha,\perp} = T_{\alpha,||}$. Thus,

$$\begin{aligned} S_\alpha &= \langle v_{||} f_\alpha, v_{||}\rangle \\ &= \langle v_{||}^2, f_\alpha\rangle \\ &= \langle v_{||}^2 + v_\perp^2, f_\alpha\rangle - \langle v_\perp^2, f_\alpha\rangle \\ &= 2(nU)_\alpha - \frac{2nT_{\alpha,\perp}}{m_\alpha} \\ &= \frac{3nT_\alpha}{m_\alpha} + nu_{||}^2 - \frac{2nT_\alpha}{m_\alpha} \\ &= \frac{nT_\alpha}{m_\alpha} + nu_{||}^2. \end{aligned}$$

Deriving the reduced kinetic flux Q_α falls out of the change of variables $\mathbf{w} = \mathbf{v} - \mathbf{u}$, that is, $w_{||} = v_{||} - u_{||}$ and $w_\perp = v_\perp$. Let $\langle\cdot,\cdot\rangle_{\mathbf{v}}$ denote the L^2 inner product (in cylindrical coordinates with azimuthal symmetry) with respect to $\mathbf{v}$, and let $\langle\cdot,\cdot\rangle_{\mathbf{w}}$ denote the L^2 inner product (in cylindrical coordinates with azimuthal symmetry) with respect to $\mathbf{w}$. First note that

$$v_{||}^2 + v_\perp^2 = v_{||}^2 - u_{||}^2 + u_{||}^2 + v_\perp^2 = (v_{||} - u_{||})^2 + 2(v_{||} - u_{||})u_{||} + u_{||}^2 + v_\perp^2.$$

With this relationship,

$$\begin{aligned} Q_\alpha &= \left\langle v_{||} f_\alpha, \frac{v_{||}^2 + v_\perp^2}{2} \right\rangle_{\mathbf{v}} = \frac{1}{2}\left\langle v_{||} f_\alpha, v_{||}^2 - u_{||}^2 + u_{||}^2 + v_\perp^2 \right\rangle_{\mathbf{v}} \\ &= \frac{1}{2}\left\langle (w_{||} + u_{||}) f_\alpha, w_{||}^2 + 2w_{||}u_{||} + u_{||}^2 + w_\perp^2 \right\rangle_{\mathbf{w}}. \end{aligned}$$

After expanding, all terms with odd powers of $w_{||}$ equal zero by symmetry.

$$\begin{aligned}
Q_\alpha &= \frac{1}{2}u_{||}\langle w_{||}^2, f_\alpha\rangle_{\mathbf{w}} + u_{||}\langle w_{||}^2, f_\alpha\rangle_{\mathbf{w}} + \frac{1}{2}u_{||}^3\langle 1, f_\alpha\rangle_{\mathbf{w}} + \frac{1}{2}u_{||}\langle w_\perp^2, f_\alpha\rangle_{\mathbf{w}} \\
&= \frac{1}{2}u_{||}\langle w_{||}^2 + w_\perp^2, f_\alpha\rangle_{\mathbf{w}} + u_{||}\langle w_{||}^2, f_\alpha\rangle_{\mathbf{w}} + \frac{1}{2}u_{||}^3\langle 1, f_\alpha\rangle_{\mathbf{w}} \\
&= \frac{3nT_\alpha u_{||}}{2m_\alpha} + u_{||}\langle v_{||}^2 - 2u_{||}v_{||} + u_{||}^2, f_\alpha\rangle_{\mathbf{v}} + \frac{nu_{||}^3}{2} \\
&= \frac{3nT_\alpha u_{||}}{2m_\alpha} + u_{||}(S_\alpha - 2nu_{||}^2 + nu_{||}^2) + \frac{nu_{||}^3}{2} \\
&= \frac{3nT_\alpha u_{||}}{2m_\alpha} + u_{||}\left(\frac{nT_\alpha}{m_\alpha} + nu_{||}^2 - nu_{||}^2\right) + \frac{nu_{||}^3}{2} \\
&= u_{||}\left(\frac{1}{2}\left(\frac{3nT_\alpha}{m_\alpha} + nu_{||}^2\right) + \frac{nT_\alpha}{m_\alpha}\right) \\
&= u_{||}\left((nU)_\alpha + \frac{nT_\alpha}{m_\alpha}\right).
\end{aligned}$$

□

If f_α is an arbitrary distribution function, then by the reduced kinetic fluxes the ***nondimensional*** **balance equations for total mass, momentum and energy** are

$$\begin{aligned}
0 &= \frac{d}{dt}\int_{\Omega_{\mathbf{x}}} n(x,t)dx + \left[nu_{||}\right]_{\partial\Omega_{\mathbf{x}}}, \\
0 &= \frac{d}{dt}\int_{\Omega_{\mathbf{x}}} (nu_{||})(x,t)dx + \left[\frac{nT_\alpha}{m_\alpha} + nu_{||}^2 - \frac{q_\alpha nT_e}{q_e m_\alpha}\right]_{\partial\Omega_{\mathbf{x}}}, \\
0 &= \frac{d}{dt}\int_{\Omega_{\mathbf{x}}} \left((nU)_\alpha + \frac{3}{2}n_eT_e\right)(x,t)dx + \left[u_{||}\left((nU)_\alpha + \frac{nT_\alpha}{m_\alpha}\right) + \frac{5}{2}u_{||}nT_e - \frac{3.2T_e^{5/2}}{\sqrt{2m_e}}\frac{\partial T_e}{\partial x}\right]_{\Omega_{\partial\mathbf{x}}}.
\end{aligned}$$

Appendix F

STENGER QUADRATURE NODES AND WEIGHTS

Found in [79, 160], let $z \in \mathbb{C}$ with $\mathfrak{Re}(z) \leq -1$. Then for each $K \in \mathbb{N}$ the quadrature nodes and weights (for $j = -K, ..., K$)

$$h_{st} \coloneqq \pi^2/\sqrt{K} \tag{F.0.1a}$$

$$t_j \coloneqq \log\left(\exp(jh_{st}) + \sqrt{1+\exp(2jh_{st})}\right) \tag{F.0.1b}$$

$$w_j \coloneqq h_{st}/\sqrt{1+\exp(-2jh_{st})} \tag{F.0.1c}$$

satisfy the error estimate

$$\left|\int_0^\infty \exp(tz)dt - \sum_{j=-K}^{K} w_j \exp(t_j z)\right| \leq C_{st}\exp(|\mathfrak{Im}(z)|/\pi)\exp(-\pi\sqrt{2K}), \tag{F.0.2}$$

where C_{st} is a constant independent of z and K.

Appendix G

A QUASI-NEWTON SOLVER FOR THE MACROSCOPIC SYSTEM

For $i = 1, 2, ..., N_x$,

$$\begin{aligned}\left(\mathbf{R}^{(\ell)}_{nu_{||}}\right)_i &= (nu_{||})^{(\ell)}_i - (nu_{||})^k_i + \frac{\Delta t}{\Delta x}\left(\hat{S}^k_{\alpha,i+\frac{1}{2}} - \hat{S}^k_{\alpha,i-\frac{1}{2}}\right) \\ &\quad - \frac{\Delta t}{2\Delta x}\frac{q_\alpha}{m_\alpha q_e}\left(n^{k+1}_{i+1}T^{(\ell)}_{e,i+1} - n^{k+1}_{i-1}T^{(\ell)}_{e,i-1}\right),\end{aligned} \tag{G.0.1a}$$

$$\begin{aligned}\left(\mathbf{R}^{(\ell)}_{(nU)_\alpha}\right)_i &= (nU)^{(\ell)}_{\alpha,i} - (nU)^k_{\alpha,i} + \frac{\Delta t}{\Delta x}\left(\hat{Q}^k_{\alpha,i+\frac{1}{2}} - \hat{Q}^k_{\alpha,i-\frac{1}{2}}\right) \\ &\quad - 3\Delta t\frac{\sqrt{2}\sqrt{m_e}}{m_\alpha^2}\frac{(n^{k+1}_i)^2}{(T^{(\ell)}_{e,i})^{3/2}}\left(T^{(\ell)}_{e,i} - \frac{m_\alpha}{3}\left(2U^{(\ell)}_{\alpha,i} - (u^{(\ell)}_{||,i})^2\right)\right) \\ &\quad - \frac{\Delta t}{2\Delta x}\frac{q_\alpha}{m_\alpha q_e}u^{(\ell)}_{||,i}\left(n^{k+1}_{i+1}T^{(\ell)}_{e,i+1} - n^{k+1}_{i-1}T^{(\ell)}_{e,i-1}\right),\end{aligned} \tag{G.0.1b}$$

$$\begin{aligned}\left(\mathbf{R}^{(\ell)}_{T_e}\right)_i &= n^{k+1}_i T^{(\ell)}_{e,i} - n^k_i T^k_{e,i} + \frac{5\Delta t}{3\Delta x}\left(u^k_{||,i+\frac{1}{2}}\widehat{nT}^k_{e,i+\frac{1}{2}} - u^k_{||,i-\frac{1}{2}}\widehat{nT}^k_{e,i-\frac{1}{2}}\right) \\ &\quad - \frac{\Delta t}{3\Delta x}u^{(\ell)}_{||,i}\left(n^{k+1}_{i+1}T^{(\ell)}_{e,i+1} - n^{k+1}_{i-1}T^{(\ell)}_{e,i-1}\right) \\ &\quad - \frac{2\Delta t}{3\Delta x^2}\left(\kappa^{(\ell)}_{e,||,i+\frac{1}{2}}(T^{(\ell)}_{e,i+1} - T^{(\ell)}_{e,i}) - \kappa^{(\ell)}_{e,||,i-\frac{1}{2}}(T^{(\ell)}_{e,i} - T^{(\ell)}_{e,i-1})\right) \\ &\quad - 2\Delta t\frac{\sqrt{2}\sqrt{m_e}}{m_\alpha}\frac{(n^{k+1}_i)^2}{(T^{(\ell)}_{e,i})^{3/2}}\left(\frac{m_\alpha}{3}\left(2U^{(\ell)}_{\alpha,i} - (u^{(\ell)}_{||,i})^2\right) - T^{(\ell)}_{e,i}\right).\end{aligned} \tag{G.0.1c}$$

$$\left(\mathbf{P}^{(\ell)}_{nu_{||},nu_{||}}\right)_{i,j} = \begin{cases} 1, & j = i, \\ 0, & \text{otherwise.}\end{cases} \tag{G.0.2}$$

$$\left(\mathbf{P}^{(\ell)}_{nu_{||},(nU)_\alpha}\right)_{i,j} = 0, \quad \forall i, j. \tag{G.0.3}$$

$$\left(\mathbf{P}^{(\ell)}_{nu_{||},T_e}\right)_{i,j} = \begin{cases} \dfrac{\Delta t}{2\Delta x}\dfrac{q_\alpha}{m_\alpha q_e} n^{k+1}_{i-1}, & j = i-1, \\ -\dfrac{\Delta t}{2\Delta x}\dfrac{q_\alpha}{m_\alpha q_e} n^{k+1}_{i+1}, & j = i+1, \\ 0, & \text{otherwise.} \end{cases} \tag{G.0.4}$$

$$\left(\mathbf{P}^{(\ell)}_{(nU)_\alpha,nu_{||}}\right)_{i,j} = \begin{cases} -\dfrac{\Delta t}{2\Delta x}\dfrac{q_\alpha}{q_e}\dfrac{1}{n^{k+1}_i}\left(n^{k+1}_{i+1}T^{(\ell)}_{e,i+1} - n^{k+1}_{i-1}T^{(\ell)}_{e,i-1}\right), & j = i, \\ 0, & \text{otherwise.} \end{cases} \tag{G.0.5}$$

$$\left(\mathbf{P}^{(\ell)}_{(nU)_\alpha,(nU)_\alpha}\right)_{i,j} = \begin{cases} 1, & j = i, \\ 0, & \text{otherwise.} \end{cases} \tag{G.0.6}$$

$$\left(\mathbf{P}^{(\ell)}_{(nU)_\alpha,T_e}\right)_{i,j} = \begin{cases} \dfrac{\Delta t}{2\Delta x}\dfrac{q_\alpha}{q_e}(nu_{||})^{(\ell)}_i \dfrac{n^{k+1}_{i-1}}{n^{k+1}_i}, & j = i-1, \\ -\dfrac{\Delta t}{2\Delta x}\dfrac{q_\alpha}{q_e}(nu_{||})^{(\ell)}_i \dfrac{n^{k+1}_{i+1}}{n^{k+1}_i}, & j = i+1, \\ 0, & \text{otherwise.} \end{cases} \tag{G.0.7}$$

$$\left(\mathbf{P}^{(\ell)}_{T_e,nu_{||}}\right)_{i,j} = \begin{cases} -\dfrac{\Delta t}{3\Delta x}\dfrac{1}{n^{k+1}_i}\left(n^{k+1}_{i+1}T^{(\ell)}_{e,i+1} - n^{k+1}_{i-1}T^{(\ell)}_{e,i-1}\right), & j = i, \\ 0, & \text{otherwise.} \end{cases} \tag{G.0.8}$$

$$\left(\mathbf{P}^{(\ell)}_{T_e,(nU)_\alpha}\right)_{i,j} = 0, \quad \forall i,j. \tag{G.0.9}$$

$$\left(\mathbf{P}^{(\ell)}_{T_e,T_e}\right)_{i,j} = \begin{cases} \dfrac{\Delta t}{3\Delta x}(nu_{||})^{(\ell)}_i \dfrac{n^{k+1}_{i-1}}{n^{k+1}_i} + \dfrac{\Delta t}{3\Delta x^2}\dfrac{3.2}{\sqrt{2}\sqrt{m_e}}\left(\dfrac{5}{2}(T^{(\ell)}_{e,i-1})^{3/2}\left(T^{(\ell)}_{e,i} - T^{(\ell)}_{e,i-1}\right)\right. & \\ \qquad \left. - \left((T^{(\ell)}_{e,i-1})^{5/2} + (T^{(\ell)}_{e,i})^{5/2}\right)\right), & j = i-1, \\ n^{k+1}_i - \dfrac{\Delta t}{3\Delta x^2}\dfrac{3.2}{\sqrt{2}\sqrt{m_e}}\left(\dfrac{5}{2}(T^{(\ell)}_{e,i})^{3/2}\left(T^{(\ell)}_{e,i+1} - T^{(\ell)}_{e,i}\right)\right. & \\ \qquad - \left((T^{(\ell)}_{e,i})^{5/2} + (T^{(\ell)}_{e,i+1})^{5/2}\right) - \dfrac{5}{2}(T^{(\ell)}_{e,i})^{3/2}\left(T^{(\ell)}_{e,i} - T^{(\ell)}_{e,i-1}\right) & \\ \qquad \left. - \left((T^{(\ell)}_{e,i-1})^{5/2} + (T^{(\ell)}_{e,i})^{5/2}\right)\right), & j = i, \\ -\dfrac{\Delta t}{3\Delta x}(nu_{||})^{(\ell)}_i \dfrac{n^{k+1}_{i+1}}{n^{k+1}_i} - \dfrac{\Delta t}{3\Delta x^2}\dfrac{3.2}{\sqrt{2}\sqrt{m_e}}\left(\dfrac{5}{2}(T^{(\ell)}_{e,i+1})^{3/2}\left(T^{(\ell)}_{e,i+1} - T^{(\ell)}_{e,i}\right)\right. & \\ \qquad \left. + \left((T^{(\ell)}_{e,i})^{5/2} + (T^{(\ell)}_{e,i+1})^{5/2}\right)\right), & j = i+1, \\ 0, & \text{otherwise.} \end{cases} \tag{G.0.10}$$

Appendix H

PERMISSIONS

Here we state the permissions for the content contained in Chapter 2. The content that references or is contained in Chapter 2, Sections 2.2-2.5 is derived and/or taken from the paper *An Eulerian-Lagrangian Runge-Kutta finite volume (EL-RK-FV) method for solving convection and diffusion equations*, published in the Journal of Computational Physics, **470** (2022), pp. 111589. This includes the portions of the abstract and Chapter 1 that overview the method presented in Chapter 2. The copyrights are owned by Elsevier, and the original publication is correctly cited [133] throughout this dissertation. Below, we attach screenshots from Elsevier's website on their copyright policy. Note the policy statements pertaining to the use of material in an author's thesis or dissertation, and the non-requirement of Elsevier's written permission.

Author rights in Elsevier's proprietary journals	Published open access	Published subscription
Retain patent and trademark rights	√	√
Retain the rights to use their research data freely without any restriction	√	√
Receive proper attribution and credit for their published work	√	√
Re-use their own material in new works without permission or payment (with full acknowledgement of the original article): 1. Extend an article to book length 2. Include an article in a subsequent compilation of their own work 3. Re-use portions, excerpts, and their own figures or tables in other works.	√	√
Use and share their works for scholarly purposes (with full acknowledgement of the original article): 1. In their own classroom teaching. Electronic and physical distribution of copies is permitted 2. If an author is speaking at a conference, they can present the article and distribute copies to the attendees 3. Distribute the article, including by email, to their students and to research colleagues who they know for their personal use 4. Share and publicize the article via Share Links, which offers 50 days' free access for anyone, without signup or registration 5. Include in a thesis or dissertation (provided this is not published commercially) 6. Share copies of their article privately as part of an invitation-only work group on commercial sites with which the publisher has a hosting agreement	√	√

Figure H.1: Elsevier's copyright permission for authors (screenshot 1). https://www.elsevier.com/about/policies/copyright (link here).

Can I use material from my Elsevier journal article within my thesis/dissertation?

As an Elsevier journal author, you have the right to Include the article in a thesis or dissertation (provided that this is not to be published commercially) whether in full or in part, subject to proper acknowledgment; see the Copyright page for more information. No written permission from Elsevier is necessary.

This right extends to the posting of your thesis to your university's repository provided that if you include the published journal article, it is embedded in your thesis and not separately downloadable.

Figure H.2: Elsevier's copyright permission for authors (screenshot 2). https://www.elsevier.com/about/policies/copyright/permissions (link here).