SPARSE GRID DISCONTINUOUS GALERKIN METHODS FOR NONLINEAR OPTICS AND MATHEMATICAL MODELING OF ASYNCHRONOUS DATA FLOW IN PARALLEL COMPUTERS

By

Kai Huang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Applied Mathematics – Doctor of Philosophy

2022

ABSTRACT

SPARSE GRID DISCONTINUOUS GALERKIN METHODS FOR NONLINEAR OPTICS AND MATHEMATICAL MODELING OF ASYNCHRONOUS DATA FLOW IN PARALLEL COMPUTERS

By

Kai Huang

This thesis consists of two parts: the first part discusses the Sparse Grid Discontinuous Galerkin (SGDG) method and its adaptive version, and their applications in Maxwell equations in nonlinear optical media [1, 2]; the second part discusses a Hamilton-Jacobi model of asynchronous data flow in parallel computers, and corresponding numerical simulation using Weighted Essentially Non-Oscillatory (WENO) method.

SGDG method and its adaptive version were developed in recent years [3, 4, 5, 6, 7], to numerically solve different linear or nonlinear PDE problems, reducing degrees of freedom and computational cost. Compared to the previous works, this thesis mainly focuses on fully implicit SGDG method of nonlinear equations, which broadens the applications of the method. To achieve this goal, the existing SGDG package [8] is coupled with nonlinear solvers in PETSc [9]. Numerical simulations of several model equations and physical relevant problems are presented to demonstrate accuracy and robustness of the method.

Presented in second part of the thesis are models of data flow on processors in a high performance computing framework involving computations necessitating inter-processor communications [10]. First comes an ordinary differential model, and its asymptotic limit results in a model which treats the computer as a continuum of processors and data flow as an Eulerian fluid governed by a conservation law. We derive a Hamilton-Jacobi equation associated with this conservation law for which the existence and uniqueness of solutions can be proved. High order WENO interpolation [11, 12], together with strong stability preserving (SSP) method for time discretization, is applied to simulation of the Hamilton-Jacobi model. We then present the results of numerical experiments for both

discrete and continuum models; these show a qualitative agreement between the two and the effect of variations in the computing environment's processing capabilities on the progress of the modeled computation.

ACKNOWLEDGEMENTS

Over the five years study in Michigan State University, I have received tremendous help from a lot of people. First and foremost, I offer my deepest appreciation to my thesis advisor, Professor Yingda Cheng. Not only does she provide me guidance on my thesis, but also she gives me valuable suggestions and generous helps throughout my PhD study. Without her patience and support in every aspect, this thesis would not have been possible.

I would also thank all of my committee members, Professor Daniel Appelo, Professor Andrew Christlieb, and Professor Jianliang Qian, for their kindness and help. I am also indebted to my undergraduate advisor, Professor Mengping Zhang, for introducing me to the research of numerical analysis.

Support from colleagues is always crucial in large research projects. I would thank my colleagues, Dr. Juntao Huang, Professor Zhanjing Tao, Professor Yan Jiang, Professor Wei Guo, Professor Yuan Liu, for their help and advice in several different parts of the thesis and my research, especially to the adaptive SGDG package and WENO method. I would also thank Dr. Richard C. Bernard, Dr. Cory Hauck, Dr. Xianzhu Tang, Dr. Qi Tang, William Sands, for their help in many aspects of my research career.

Besides, I would appreciate my friends, Anqi Chen, Andres Felipe Galindo-Olarte, Eric Cheuk-Wai Yau, Jing Huang, Rui Wang, Yao Li, Runze Su, Jialin Qu, Jian Song, Zhichao Peng, etc. for their companion and kind help in the many years.

Last but not the least, I am eternally grateful to my parents, Haizhuang Huang and Xianhua Liu. Their persistent support and unconditional love encourage me to pursue higher goals in my life, and make me who I am today.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

In this chapter, we briefly introduce several numerical methods, which are the focus of our research projects: Discontinuous Galerkin (DG) Method, its variant: Sparse Grid Discontinuous Galerkin (SGDG) Method, and Weighted Essentially Non-Oscillatory (WENO) Method.

1.1 Discontinuous Galerkin (DG) Method

Discontinuous Galerkin (DG) Method is a specific kind of finite element methods. Comparing to traditional finite element method, basis functions of DG Methods are chosen to be discontinuous on element interfaces, such as piecewise polynomials, which leads to the introduction of numerical flux. By properly choosing numerical flux, the resulting numerical method is stable and convergent.

The DG Method for transport equation was first introduced by Reed and Hill in 1973, and discussed in a report [13] of Los Alamos National Laboratory. The method was to solve the neuron transport equations, which are time-independent linear hyperbolic equations. In late 1980s and early 90s, a major development, namely Runge-Kutta DG Methods, was established by Cockburn and Shu in a series of paper [14, 15, 16, 17, 18], which is a framework to solve nonlinear time-dependent hyperbolic equations, e.g. Euler equations of compressible gas dynamics [19]. Under this framework, the DG discretization is used only for spatial variables, while explicit, nonlinearly stable high order Runge-Kutta methods [20, 21] are used to discretize the time variable. To avoid oscillation of numerical solutions when strong shock occurs, exact or approximate Riemann solvers as interface fluxes and total variation bounded (TVB) nonlinear limiters [22, 23] are introduced from finite volume methods. DG methodology was also generalized to treat viscous terms as well, and thus the DG methods were designed to solve Navier-Stokes

equations [24, 25].

Comparing to classical finite volume and finite difference methods, DG Methods have certain advantages [26]. By choosing polynomials of high degree, the corresponding DG Methods can achieve high order of accuracy; the mass matrix is sparse so the DG Methods are parallelizable. In addition, DG Methods can handle h-p adaptivity; due to the discontinuous basis functions, the continuity restrictions for conforming finite element methods need not be considered, and grid refinement or unrefinement is easy to handle. Adaptivity is of particular interest for hyperbolic systems and can potentially advance computational efficiency [27, 28].

In 1970s, independent research of Galerkin Methods applying to elliptic and parabolic equations emerged, as discussed in [29], which are now called interior penalty (IP) methods. The idea of penalty formulation could be traced back to 1960s, e.g. [30]. In 1971, Nitsche [31] developed the first penalty method, where a penalty term inversely proportional to mesh size was introduced to guarantee stability and optimal order of convergence for smooth exact solution. Instead, Babuska's approach [32] used a more flexible penalty term, with consistency error and sub-optimal convergence rate. The method was further analyzed and generalized to nonlinear elliptic and parabolic problems by Arnold [33], which were summarized in [34]. After all, a unified framework is proposed [29, 35], for both primal formulation inspired by original interior penalty method, and methods inspired by finite volume methods of hyperbolic problems with proper numerical flux. We also refer to textbooks like [36] for further reference.

Since the DG Methods were introduced, they have found rapid applications in such diverse areas as aeroacoustics, electro-magnetism, gas dynamics, granular flows, magnetohydrodynamics, meteorology, modeling of shallow water, oceanography, oil recovery simulation, semiconductor device simulation, transport of contaminant in porous media, turbomachinery, turbulent flows, viscoelastic flows and weather forecasting, among many others. For further discussions of this, we refer to review papers e.g. [26, 27, 37, 28].

For earlier works on DG methods, we refer to the survey paper [26], and other papers in the same Springer volume. The lecture notes [38] is a good reference for many details, as well as the extensive review paper [39]. There are several special journal issues devoted to the DG method [40, 41, 42, 43, 44], which contain many interesting papers on DG method in all aspects including algorithm design, analysis, implementation and applications. There are also a few books and lecture notes [45, 46, 47, 36, 48, 49] on DG methods.

DG methods have also grown to be broadly adopted for electromagnetic simulations in the past two decades. They have been developed and analyzed for time dependent linear models, including Maxwell's equations in free space (e.g., [50, 51, 52]), dispersive media (e.g., [53, 54, 55, 56]), as well as meta-materials (e.g., [57, 58, 59, 60]). However, there exists only limited study for DG methods for nonlinear Maxwell models. For example, in [61, 62], Kerr nonlinearity was investigated, where the entire Maxwell PDE-ODE system was cast as a nonlinear hyperbolic conservation law, for which DG methods have long been known for their success. A relaxed version of the Kerr model, called the Kerr–Debye model, was examined in [63], where a second-order asymptotic-preserving and positivity-preserving DG scheme is designed and analyzed; [64] also devised and analyzed asymptotic-preserving and positivity-preserving methods for the Kerr-Debye-Lorentz model.

1.2 Sparse Grid Discontinuous Galerkin (SGDG) Method

As discussed above, DG methods have overall good performance for different kinds of problems like hyperbolic or elliptic equations, and they admit flexibility in choosing the discretization meshes and the approximation spaces. However, the methods are often considered too costly due to the large degrees of freedom of the approximation space, especially for problems on high dimension, due to curse of dimensionality [65]. Such drawback is more prominent when dealing with high-dimensional equations arising from

real-world applications, such as kinetic simulations, stochastic analysis, and mathematical modeling in finance or statistics.

The sparse grid techniques [66, 67] was introduced by Zenger [68], and later becomes a major tool to break the curse of dimensionality of grid-based approaches. The idea relies on a tensor product hierarchical basis representation, which can reduce the degrees of freedom without compromising too much on accuracy. The fundamentals of sparse grid techniques can further date back to Smolyak [69] for numerical integration, and they are closely related to hyperbolic cross [70, 71], boolean method [72], discrete blending method [73], and splitting extrapolation method [74]. The construction of the scheme is to seek a proper truncation of the tensor product hierarchical basis, which can be formally derived by solving an optimization problem of cost-benefit ratios [75]. Sparse grid techniques have been incorporated in collocation methods for high-dimensional stochastic differential equations, Galerkin finite element methods, finite difference methods, finite volume methods, and spectral methods for high-dimensional PDEs [3].

Employing the similar sparse grid techniques as above, Sparse Grid Discontinuous Galerkin (SGDG) Methods are a specific class of DG Methods. Instead of choosing piecewise polynomials for basis functions as in typical DG Methods, as introduced in [3, 4, 5], SGDG uses basis functions derived from Alpert's multiwavlet basis functions [76, 77] and multiresolution analysis (MRA) [78]. In one dimension, SGDG Methods would be an equivalent reformulation of traditional DG Methods; however for higher dimension case, to break the curse of dimensionality, we use a subset of tensor product approximation space for DG approximation, based on the hierarchial structure of multiwavelet basis functions. SGDG Methods efficiently reduce the number of degrees of freedom (DoF) of the unknowns from $O(h^{-d})$ to $O(h^{-1}|\log_2 h|^{d-1})$ for d-dimensional problems [3], where h is the uniform mesh size in each dimension. Stability and conservation can be maintained, while error is only slightly deteriorated for sufficiently smooth solutions [4]. SGDG Methods have been applied to elliptic equations [3], transport equations [4],

reaction-diffusion equations [79], Helmholtz equation [80], Vlasov–Maxwell equations [81], radioactive transfer equation [82], etc.

Success of SGDG Methods is encouraging, but several improvement comes afterwards. The main bottleneck is that the previous method can only treat some kind of linear equations with given variable coefficients, or coefficients with specified dependence to unknowns. To compute of nonlinear terms in nonlinear equations, interpolatory multiwavelets for MRA quadrature and sparse grid collocation method [7, 83] were developed, where new multiwavelets called interpolatory multiwavelets were associated to interpolation on the hierarchial grid. Besides, since the Alpert's and interpolatory multiwavelet basis functions are global, it is essential to find efficient implementation to make sure computational cost comparable to traditional DG Methods, so a fast algorithm regarding matrix-vector multiplication based on [84, 85] was also introduced into SGDG method.

In addition, if the exact solution is not smooth enough, the standard SGDG Methods might not be a good choice, since it requires fine mesh to capture the non-smooth solution. In order to overcome this, a family of adaptive SGDG Methods for linear transport equations were developed [5], by using fully tensorized basis functions, with hierarchial surplus as refinement or coarsening indicator automatically capturing the local structures. The adaptive method has also been applied to kinetic equations [5], hyperbolic conservation laws [6], wave equations [83, 86], Hamilton-Jacobi equation [87], and nonlinear Schrodinger equation [88].

1.3 Weighted Essentially Non-Oscillatory (WENO) Method

WENO (Weighted Essentially Non-Oscillatory) Methods, together with ENO (Essentially Non-Oscillatory) Methods, are high order accurate finite difference or finite volume schemes designed for solving hyperbolic and convection-diffusion equations with possibly discontinuous solutions or solutions with sharp gradient regions. The key idea of ENO and WENO Methods is an nonlinear adaptive approximation procedure that achieves high

order accuracy on smooth region of function, while resolving shock or other discontinuities sharply without substantial oscillations. Because the main idea of the methods is not necessarily related to PDEs, ENO or WENO Methods have several non-PDE applications, too [89].

High order finite difference and finite volume methods are based on interpolations of discrete data, mostly by using algebraic polynomials. Approximation theory guarantees that a wider interpolation stencil yields higher order accuracy for smooth function. Fixed stencils are usually applied in these methods, and perform well in smooth problems; however, fixed stencil interpolation causes unavoidable oscillation called Gibbs phenomena near discontinuity, which cannot be eliminated by mesh refinement, leading to numerical instability in nonlinear problems involving discontinuities [90]. In applications such as hyperbolic conservation laws, Hamilton-Jacobi Equations, or convection-diffusion equations, etc. the exact solution might contain discontinuities on either the solution itself or its derivative, regardless of smoothness of initial or boundary conditions [89, 11], and fixed stencil interpolation is a less appealing choice in these cases.

ENO schemes were first introduced by Harten, Engquist, Osher and Chakravarthy in 1987 [91] in the finite volume framework, which in the first time successfully attempting to obtain a self-similar, uniformly high order accurate, and essentially non-oscillatory interpolation. In this paper, they used Newton divided differences to determine the local smoothness of the function to be approximated, which indicates the relatively smooth stencil to be chosen among candidate stencils for interpolation. Later, ENO methods in finite difference framework, together with TVD Runge-Kutta time integration, were shown save significant amount of computational cost in higher dimension [92, 93].

Based on ENO, WENO methods were developed using convex combination of all candidates stencils instead of only one stencil as in ENO. The crucial ingredient is choice of the combination coefficients, also called nonlinear weights, to fulfill the following [89]: (1) when the solution is sufficiently smooth, the nonlinear weight should be close to so-called

"linear weight", which guarantees high order accuracy in combined stencil; (2) when some candidate stencils contain discontinuity while others don't, the stencils containing discontinuities should be associated with smaller nonlinear weight.

The first WENO Method [94] was the one dimension case for hyperbolic conservation laws, under finite volume framework, while [95] provided the multidimensional version, and in addition, [96, 97] improved accuracy. [98] offered 1D finite volume formulation based on a staggered grid and Lax-Friedrichs formulation. [99] developed the multidimensional finite difference formulation with improved accuracy, and a general framework in forming a $(2k+1)$-order WENO approximation from a k-th order ENO stencil was established. The fifth order WENO in [99] becomes the most commonly used WENO method. Higher order ($(2k+1)$-th order, with $k = 3, 4, 5, ...$) WENO reconstruction procedures were developed in [100]. WENO improves upon ENO in robustness, better smoothness of fluxes, better steady state convergence, better provable convergence properties, and more efficiency. We refer to lecture notes and review papers [11, 101, 89, 12] for further details.

Among the applications of WENO methods, we consider Hamilton-Jacobi equations

$$\phi_t + H(\phi_{x_1}, ..., \phi_{x_d}) = 0, \quad \phi(x, 0) = \phi^0(x),$$

where H is usually nonlinear but at least Lipschitz continuous. It is widely known that global C^1 solution does not exist for this equation, regardless of smoothness of initial condition. This is a direct implication, at least for one dimension case, from the fact that if we take $u = \phi_x$, the Hamilton-Jacobi equation is equivalent to hyperbolic conservation law of function u. Singularities of u are discontinuities, so u is bounded with bounded variation, and so is derivative of ϕ. The idea of viscosity solution, a specific kind of weak solution, is then introduced to theory of Hamilton-Jacobi equation, to describe the unique physical relevant solution. More details are discussed in e.g. [102, 103, 104].

Because of the relation between Hamilton-Jacobi equation and hyperbolic conservation law, ENO and WENO method applied in conservation law would be similarly ap-

plied to Hamilton-Jacobi equation. High order ENO schemes for solving Hamilton-Jacobi equations were developed in [105] for the second order case and in [106] for the more general cases, based on ENO schemes for solving conservation laws [91, 92, 93]. High order WENO schemes for solving Hamilton-Jacobi equations were developed in [107], based on WENO schemes for solving conservation laws [94, 99]. The framework of WENO schemes for solving Hamilton-Jacobi equations is again similar to that of ENO schemes described in the previous section. The key ingredient in designing a nonlinear weight, as discussed in [107], is similar to that in [99] for conservation laws, namely the smoothness indicator is a scaled sum of the squares of the L^2 norms of the second and higher derivatives of the interpolation polynomial on the target interval. We refer to the lecture notes of Shu [12] for more details of ENO and WENO schemes.

To couple with ENO and WENO spatially semi-discrete methods above, a popular time discretization method to choose is the class of strong stability preserving (SSP), which is also referred to as total variation diminishing (TVD) or high order Runge-Kutta time discretizations; see [92, 108, 109, 110].

1.4 Organization of Thesis

In Chapter 2, we consider the Maxwell equations with nonlinear optical media, where the nonlinearity comes from cubic Kerr effect and Raman scattering [111, 112]. The numerical simulation of such equations is considered to be computationally intensive but substantially more robust and physically relevant compared with Nonlinear Schrodinger Equation (NLSE) models. A semi-discrete energy-stable DG method was applied to such equations in one dimension [1] or higher dimension [2], while fully implicit method or trapezoidal method was used for time discretization. We construct the method under adaptive SGDG framework [5, 6, 83], with L^2 indicator norm on each element for coarsening and refinement. Sparse grid collocation method with interpolatory multiwavelet basis functions are required to deal with nonlinear problems, and its adaptive version

reduces computational cost [7]. Fast wavelet transform is applied to transform between point values or derivatives of numerical solution at interpolation points, and coefficients of hierarchial multiwavelet basis functions; the fast transform computes results dimension by dimension after proper decomposition of one dimension matrix operator into upper and lower triangular part, so computational cost is further reduced [6, 7]. A C++ package is developed by a group of colleagues, for general numerical simulation using SGDG framework [8], incorporating the standard SGDG implementation [3] with Alpert's multiwavelet basis [76], Lagrangian and Hermite interpolatory multiwavelet [7] for nonlinear problems, adaptivity [5], fast matrix-vector multiplication [6], etc. Several time integration method, including IMEX-RK [6] was implemented in the package, and data structures, linear solvers, and multi-thread parallelism in Eigen package [113] of numerical linear algebra is integrated to the SGDG package. In our work, we further apply PETSc package [114] to implement the adaptive SGDG method. PETSc offers a whole setting of advanced data structures, which are designed for multi-core parallel computing; its rich library of nonlinear solvers [115], including a group of matrix-free method e.g. Jacobian-free Newton–Krylov methods [116], is vital for general simulations for nonlinear problems when Jacobian is not attainable. Several examples are made to test accuracy and to show efficiency of the implementation of the presented algorithm.

In Chapter 3, we present a simplified ordinary differential model of data flow on processors in a high performance computing framework, which involves computations necessitating inter-processor communications. The asymptotic limit of this model treats the computer as a continuum of processors and data flow as an Eulerian fluid governed by a conservation law, which implies a Hamilton-Jacobi equation [103, 104] for which the existence and uniqueness of solutions can be proven. We choose WENO method for numerical simulation of the continuum Hamilton-Jacobi model; WENO interpolation uses a convex combination of candidate stencils, and by assigning each stencil a nonlinear weight based on local smoothness of numerical solution on the stencil, WENO interpola-

tion has high order accuracy [11]. Comparing to other numerical methods like artificial viscosity or TVD method, there is no problem-dependent parameter in WENO method, and no accuracy degeneration occurs in smooth region of numerical solution [12]. Thus numerical experiments for the continuum model are calculated using a spatially fifth order WENO interpolation coupling with optimal third order SSP Runge-Kutta time integration [12, 11], and compared with results from discrete model. A qualitative agreement is shown between the simulations on discrete and continuum models, and we investigate the effect of variations in the computing environment's processing capabilities on the progress of the modeled computation. The major contents of this chapter was already published in [10].

CHAPTER 2

APPLICATION OF SGDG METHOD TO MAXWELL'S EQUATION IN NONLINEAR OPTICS

2.1 Maxwell's Equation in Nonlinear Optics

The propagation of electromagnetic waves in general media is modeled by the time-dependent Maxwell's partial differential equations (PDEs), coupled with constitutive laws that describe the response of the media. Particularly, in nonlinear media, the material response depends nonlinearly on the optical field, and many interesting physical phenomena, such as frequency mixing and second/third-harmonic generation have been observed and harnessed for practical applications. We refer to classical textbooks [117, 118, 119] for a more detailed review of the field of nonlinear optics.

When Maxwell's equations are considered to model the electromagnetic waves propagating through a nonlinear optical medium, the medium response is described by constitutive laws that relate the electric field $\mathbf{E}$ and the electric flux density $\mathbf{D}$ through the polarization $\mathbf{P}$ of the medium. Here we focus on a macroscopic phenomenological description of the polarization, which comprises both linear and nonlinear responses. Specifically, the linear response is modeled by a single resonance Lorentz dispersion, while the nonlinear response is cubic and incorporates the instantaneous Kerr effect and the delayed nonlinear Lorentz dispersion called Raman scattering. Within this description, we will follow the auxiliary differential equation (ADE) approach, where the linear and nonlinear Lorentz dispersion is represented through a set of ODEs, describing the time evolution of $\mathbf{P}$ (hence of $\mathbf{D}$) forced by $\mathbf{E}$, appended to Maxwell's equations. An alternative representation is via a recursive convolution method, where $\mathbf{D}$ is computed from $\mathbf{E}$ through a time convolution integral [120, 1].

We begin with the Maxwell's equations, which govern the time evolution of the elec-

tric field $\mathbf{E}$ and magnetic field $\mathbf{H}$ in a non-magnetic nonlinear optical medium, on time domain $(0, T)$ and spatial domain Ω:

$$\partial_t \mathbf{B} + \nabla \times \mathbf{E} = 0 \tag{2.1}$$

$$\partial_t \mathbf{D} + \mathbf{J}_s - \nabla \times \mathbf{H} = 0 \tag{2.2}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.3}$$

$$\nabla \cdot \mathbf{D} = \rho \tag{2.4}$$

along with initial and boundary data in the domain $\Omega \subset R^d,\ d = 1, 2, 3$. The variable $\mathbf{J}_s$ is the source current density, and ρ is the charge density. The electric flux density $\mathbf{D}$ and the magnetic induction $\mathbf{B}$ are related to the electric and magnetic field, respectively, via the constitutive laws

$$\mathbf{D} = \epsilon_0(\epsilon_\infty \mathbf{E} + \mathbf{P}) \tag{2.5}$$

$$\mathbf{B} = \mu_0 \mathbf{H} \tag{2.6}$$

where $\mathbf{P}$ is the polarization. The dielectric parameter is ϵ_0, the electric permittivity of free space, ϵ_∞, the relative electric permittivity in the limit of the infinite frequency, and μ_0, the magnetic permeability of free space. We will assume here that all model parameters are constant, and the material is isotropic.

To model the linear and nonlinear dispersion in the material we use the auxiliary differential equation (ADE) approach as presented in [121, 120]. A thorough discussion of the modeling of Raman and Kerr effects in optical (silica) fibers can be found in [111]. The linear (L) delayed or retarded response of the material to the electromagnetic field is captured in the polarization, $\mathbf{P}$, via a linear single resonance Lorentz response, which, in the form of a second order ODE, is provided as,

$$\frac{\partial^2 \mathbf{P}^L_{Delay}}{\partial t^2} + \frac{1}{\tau}\frac{\partial \mathbf{P}^L_{Delay}}{\partial t} + \omega_0^2 \mathbf{P}^L_{Delay} = \omega_p^2 \mathbf{E},$$

which could be split into first order system

$$\frac{\partial \mathbf{P}^L_{Delay}}{\partial t} = \mathbf{J}, \quad \frac{\partial \mathbf{J}}{\partial t} = -\frac{1}{\tau}\mathbf{J} - \omega_0^2 \mathbf{P}^L_{Delay} + \omega_p^2 \mathbf{E}.$$

Here ω_0 and ω_p are the resonance and plasma frequencies of the medium, respectively, and τ^{-1} is a damping constant. In addition, $\omega_p^2 = (\epsilon_s - \epsilon_0)\omega_0^2$, with ϵ_s as the relative permittivity at zero frequency.

For pulse widths that are sufficiently short (for e.g., shorter than 1 pico-second (ps) for Silica) [122], the nonlinear response has an instantaneous as well as a delayed component. For the nonlinear (NL) response of the medium, we will consider a cubic Kerr-type instantaneous response, and a retarded Raman molecular vibrational response called Raman scattering. The Kerr effect is a phenomenon in which the refractive index of a material changes proportionally to the square of the applied electric field. Raman scattering arises from the electric field induced changes in the internal nuclear vibrations on time scales about 1 to 100 femto-seconds (fs) [112], and is modeled by a nonlinear single resonance Lorentz delayed response. The two nonlinear responses are given as

$$\mathbf{P}_{Kerr}^{NL} = a(1-\theta)\mathbf{E}|\mathbf{E}|^2,$$

and

$$\mathbf{P}_{Delay}^{NL} = a\theta Q\mathbf{E},$$

while the total nonlinear response is $\mathbf{P}^{NL} = \mathbf{P}_{Kerr}^{NL} + \mathbf{P}_{Delay}^{NL}$. Here a is a third order coupling constant, θ parameterizes the relative strength of the instantaneous electronic Kerr and retarded Raman molecular vibrational responses, and Q describes the natural molecular vibrations within the dielectric material that has frequency many orders of magnitude less than the optical wave frequency, responding to the field intensity. The time evolution of Q is given by the following ODE,

$$\frac{\partial^2 Q}{\partial t^2} + \frac{1}{\tau_v}\frac{\partial Q}{\partial t} + \omega_v^2 Q = \omega_v^2|\mathbf{E}|^2,$$

where ω_v is the resonance frequency of the vibration, and τ_v^{-1} a damping constant. This is essentially a model for a simple linear oscillator, but coupled to the nonlinear field intensity $|\mathbf{E}|^2$.

At the end we obtain the following system, on $(0,T)\times\Omega$,

$$\begin{aligned}
\mu_0\partial_t\mathbf{H} &= -\nabla\times\mathbf{E} && \text{(2.7a)}\\
\partial_t\mathbf{D} &= \nabla\times\mathbf{H} && \text{(2.7b)}\\
\partial_t\mathbf{P} &= \mathbf{J} && \text{(2.7c)}\\
\partial_t\mathbf{J} &= -\frac{1}{\tau}\mathbf{J}-\omega_0^2\mathbf{P}+\omega_p^2\mathbf{E} && \text{(2.7d)}\\
\partial_t Q &= \sigma && \text{(2.7e)}\\
\partial_t\sigma &= -\frac{1}{\tau_v}\sigma-\omega_v^2 Q+\omega_v^2|\mathbf{E}|^2 && \text{(2.7f)}
\end{aligned}$$

with constitutive law

$$\mathbf{D}=\epsilon_0(\epsilon_\infty\mathbf{E}+\mathbf{P}+a(1-\theta)|\mathbf{E}|^2\mathbf{E}+a\theta Q\mathbf{E}). \tag{2.8}$$

Note that here $\mathbf{P}$ is essentially $\mathbf{P}^L_{Delay}$, the linear delayed response. As demonstrated in [1, 2], under the assumption of periodic boundary conditions, the energy of the system is

$$\begin{aligned}
\mathcal{E}(t) &= \int_\Omega \frac{\mu_0}{2}|\mathbf{H}|^2+\frac{\epsilon_0\epsilon_\infty}{2}|\mathbf{E}|^2+\frac{\epsilon_0}{2\omega_p^2}|\mathbf{J}|^2+\frac{\epsilon_0\omega_0^2}{2\omega_p^2}|\mathbf{P}|^2+\frac{\epsilon_0 a\theta}{4\omega_v^2}\sigma^2+\frac{\epsilon_0 a\theta}{2}Q|\mathbf{E}|^2\\
&\quad+\frac{3\epsilon_0 a(1-\theta)}{4}|\mathbf{E}|^4+\frac{\epsilon_0 a\theta}{4}Q^2\ d\Omega, && \text{(2.9)}
\end{aligned}$$

satisfying

$$\frac{d}{dt}\mathcal{E}=-\frac{\epsilon_0}{\omega_p^2\tau}\int_\Omega|\mathbf{J}|^2dx-\frac{\epsilon_0 a\theta}{2\omega_v^2\tau_v}\int_\Omega\sigma^2dx\le 0. \tag{2.10}$$

Besides, $\mathcal{E}\ge 0$ if $\theta\in[0,3/4]$.

The one dimensional case is as following:

$$\begin{aligned}
\mu_0\partial_t H &= \partial_x E && \text{(2.11a)}\\
\partial_t D &= \partial_x H && \text{(2.11b)}\\
\partial_t P &= J && \text{(2.11c)}\\
\partial_t J &= -\frac{1}{\tau}J-\omega_0^2 P+\omega_p^2 E && \text{(2.11d)}\\
\partial_t Q &= \sigma && \text{(2.11e)}\\
\partial_t\sigma &= -\frac{1}{\tau_v}\sigma-\omega_v^2 Q+\omega_v^2 E^2 && \text{(2.11f)}\\
D &= \epsilon_0(\epsilon_\infty E+P+a(1-\theta)E^3+a\theta QE) && \text{(2.11g)}
\end{aligned}$$

Note that here we assume uniformity of all the vector fields in the y and z directions. Thus, all derivatives with respect to y and z in the curl and divergence operators are set to zero. All field quantities are represented by a single scalar component. The scalar magnetic field H (hence B) represents the 2nd (or the 3rd) component of the vector magnetic field $\mathbf{H}$, and the scalar electric flux density D (hence E) represents the 3rd (or the 2nd) component of $\mathbf{D}$ (hence $\mathbf{E}$). Gauss's laws (2.3) (2.4) only involve the x derivatives of the first components of $\mathbf{B}$ and $\mathbf{D}$, and therefore they are decoupled from the one-dimensional model and become irrelevant. Numerical simulation later depends on dimensionless version [1]

$$\partial_t H = \partial_x E \tag{2.12a}$$

$$\partial_t D = \partial_x H \tag{2.12b}$$

$$\partial_t P = J \tag{2.12c}$$

$$\partial_t J = -\frac{1}{\tau} J - \omega_0^2 P + \omega_p^2 E \tag{2.12d}$$

$$\partial_t Q = \sigma \tag{2.12e}$$

$$\partial_t \sigma = -\frac{1}{\tau_v} \sigma - \omega_v^2 Q + \omega_v^2 E^2 \tag{2.12f}$$

$$D = \epsilon_\infty E + P + a(1-\theta)E^3 + a\theta Q E \tag{2.12g}$$

2.2 Formulation of Energy Stable Adaptive SGDG Method in One Dimension

In this section, we describe the energy-stable adaptive SGDG method in one dimension. We start from general formulation of SGDG method, and how it is applied to the nonlinear Maxwell equation as we specified above. Adaptive scheme based on SGDG formulation will also be discussed in the end of the section.

2.2.1 SGDG Method and Alpert's Multiwavelets

We first define the one-dimension spatial domain and its SGDG basis functions [3]. Consider $\Omega = [0, 1]$ as our spatial domain, and define a set of nested grids, where the n-th level grid Ω_n consists of 2^n uniform cells

$$I_n^j = (2^{-n}j, 2^{-n}(j+1)], j = 0, 1, \ldots, 2^n - 1$$

for any $n \geq 0$. For notation convenience, we also denote $I_{-1} = [0, 1]$. Define

$$V_n^k = \{v : v \in P^k(I_n^j), j = 0, 1, \ldots, 2^n - 1\},$$

the piecewise polynomial space, of degree at most k, on n-th level grid Ω_n, and we have the nested structure

$$V_0^k \subset V_1^k \subset V_2^k \subset \ldots.$$

Define W_n^k the orthogonal complement of V_{n-1}^k in V_n^k, with respect to L^2 inner product in $[0, 1]$, i.e.

$$V_{n-1}^k \bigoplus W_n^k = V_n^k, \quad V_{n-1}^k \perp W_n^k.$$

Again for notation convenience, let $W_0^k = V_0^k$; then for any natural number N,

$$V_N^k = \bigoplus_{0 \leq n \leq N} W_n^k.$$

As discussed in [76], we define a set of orthonormal basis functions for each W_n^k, $n = 0, \ldots, N$ and they form a basis of V_N^k. The case of grid level $n = 0$ is trivial; the normalized shifted Legendre polynomials in $[0, 1]$ will be proper choice for such purpose. Denote these Legendre polynomials by $v_{i,0}^0$ for $i = 0, \ldots, k$. When $n > 0$, we consider the orthonormal basis of W_1^k; basis of general $W_n^k, n \geq 1$ will turn out to be a scaling version of basis of W_1^k. In particular, orthonormal basis of W_1^k is defined as

$$h_i(x) = \sqrt{2} f_i(2x - 1), i = 0, 1, \ldots, k,$$

where for fixed integer $k \geq 0$, $f_0, f_1, \ldots, f_k$ are a sequence of functions supported on $[-1, 1]$, satisfying

1. f_i on $(0,1)$ is a polynomial with degree k;

2. each f_i extends evenly or oddly to $(-1,0)$, as in the following formula

$$f_i(x) = (-1)^{i+k+1} f_i(-x), \tag{2.13}$$

   for any $x \in (-1,0)$;

3. $f_0, f_1, \ldots, f_k$ are orthonormal i.e.

$$\int_{-1}^{1} f_i(x) f_j(x) dx = \delta_{ij}$$

   for all $0 \leq i, j \leq k$, where δ_{ij} equals to 1 if $i = j$ and 0 otherwise;

4. f_j has vanishing moments

$$\int_{-1}^{1} f_j(x) x^i dx = 0,$$

   with $i = 0, ..., j + k$, for each $0 \leq j \leq k$.

It was shown [76] that we can compute all f_i through a Gram-Schmidt procedure. The particular form of f_i up to $k = 4$ are provided in Table 1 of [76], and for completeness we offer these results here. For simplicity, we only provide definition of f_i on $(0,1)$, and its values on $(-1,0)$ are obtained by proper extension as described in equation (2.13).

- $k = 0$

$$f_0(x) = \sqrt{\frac{1}{2}};$$

- $k = 1$

$$f_0(x) = \sqrt{\frac{3}{2}}(-1 + 2x), \; f_1(x) = \sqrt{\frac{1}{2}}(-2 + 3x);$$

- $k = 2$

$$f_0(x) = \frac{1}{3}\sqrt{\frac{1}{2}}(1 - 24x + 30x^2), \quad f_1(x) = \frac{1}{2}\sqrt{\frac{3}{2}}(3 - 16x + 15x^2),$$

$$f_2(x) = \frac{1}{3}\sqrt{\frac{5}{2}}(4 - 15x + 12x^2);$$

- $k = 3$:

$$f_0(x) = \sqrt{\frac{15}{34}}(1 + 4x - 30x^2 + 28x^3), \quad f_1(x) = \sqrt{\frac{1}{42}}(-4 + 105x - 300x^2 + 210x^3),$$

$$f_2(x) = \frac{1}{2}\sqrt{\frac{35}{34}}(-5+48x-105x^2+64x^3), \quad f_3(x) = \frac{1}{2}\sqrt{\frac{5}{42}}(-16+105x-192x^2+105x^3);$$

- $k = 4$:

$$\begin{aligned}
f_0(x) &= \sqrt{\frac{1}{186}}(1 + 30x + 210x^2 - 840x^3 + 630x^4), \\
f_1(x) &= \frac{1}{2}\sqrt{\frac{1}{38}}(-5 - 144x + 1155x^2 - 2240x^3 + 1260x^4), \\
f_2(x) &= \sqrt{\frac{35}{14694}}(22 - 735x + 3504x^2 - 5460x^3 + 2700x^4), \\
f_3(x) &= \frac{1}{8}\sqrt{\frac{21}{38}}(35 - 512x + 1890x^2 - 2560x^3 + 1155x^4), \\
f_4(x) &= \frac{1}{2}\sqrt{\frac{7}{158}}(32 - 315x + 960x^2 - 1155x^3 + 480x^4).
\end{aligned}$$

These multiwavelet functions retain orthonormal properties of wavelet bases for different hierarchical levels. More precisely, W_n^k with $n \geq 1$ has basis

$$v_{i,n}^j(x) = 2^{(n-1)/2} h_i(2^{n-1}x - j) = 2^{n/2} f_i(2^n x - 2j - 1), \quad i = 0, ..., k,\ j = 0, ..., 2^{n-1} - 1.$$

Each $v_{i,n}^j$ is supported on I_{n-1}^j, with discontinuity on $x = 2^{-n}w,\ w = 2j, 2j+1, 2j+2$. Note that $\{v_{i,n}^j\}_{0\leq i\leq k, n\geq 1}^{0\leq j\leq 2^{n-1}-1} \bigcup \{v_{i,0}^0\}_{0\leq i\leq k}$ is an orthonormal set in $L^2([0,1])$, i.e.

$$\int_0^1 v_{i,n}^j(x) v_{i',n'}^{j'}(x) dx = \delta_{ii'}\delta_{jj'}\delta_{nn'},$$

while $\{v_{i,n}^j\}_{0\leq i\leq k,\ 1\leq n\leq N}^{0\leq j\leq 2^{n-1}-1} \bigcup \{v_{i,0}^0\}_{0\leq i\leq k}$ is an orthonormal basis of V_N^k.

2.2.2 Semi-Discrete DG Method for the Maxwell's Equation

To formulate semi-discrete DG method [1, 27] for system (2.11), we need some more notations. Recall that we use N to represent the finest level of nested grid. Define

$$x_{i+1/2} = \frac{i}{2^N}, \quad i = 0, 1, \ldots, 2^N,$$

the discrete points on $\Omega = [0, 1]$ containing discontinuities for functions in V_N^k. As defined above, $I_N^j = (x_{j-1/2}, x_{j+1/2}]$ for $j = 1, 2, \ldots, 2^N$; $h = 2^{-N}$ is mesh size. We also denote v^+, v^- the right and left limit on discontinuities, and $[v] = v^+ - v^-$ the jump, $\{v\} = (v^+ + v^-)/2$ the average. They can be evaluated at each cell boundary $x_{j\pm1/2}$, denoted by e.g $(v^+)_{j\pm1/2}$. Then on each cell I_N^j, we can formulate the standard semi-discrete DG method of system (2.11) following [27, 1]: find functions $H_h(t,\cdot)$, $D_h(t,\cdot)$, $E_h(t,\cdot)$, $P_h(t,\cdot)$, $J_h(t,\cdot)$, $Q_h(t,\cdot)$, $\sigma_h(t,\cdot)$, on space V_N^k, such that for each $j = 0, 1, \ldots, 2^N - 1$ and each cell I_N^j,

$$\mu_0 \int_{I_N^j} \partial_t H_h \phi dx + \int_{I_N^j} E_h \partial_x \phi dx - (\widehat{E_h}\phi^-)_{j+1/2} + (\widehat{E_h}\phi^+)_{j-1/2} = 0, \ \forall \phi \in V_N^k \tag{2.14a}$$

$$\int_{I_N^j} \partial_t D_h \phi dx + \int_{I_N^j} H_h \partial_x \phi dx - (\widetilde{H_h}\phi^-)_{j+1/2} + (\widetilde{H_h}\phi^+)_{j-1/2} = 0, \ \forall \phi \in V_N^k \tag{2.14b}$$

$$\partial_t P_h = J_h \tag{2.14c}$$

$$\partial_t J_h = -\frac{1}{\tau} J_h - \omega_0^2 P_h + \omega_p^2 E_h \tag{2.14d}$$

$$\partial_t Q_h = \sigma_h \tag{2.14e}$$

$$\int_{I_N^j} \partial_t \sigma_h \phi dx = -\int_{I_N^j} \left(\frac{1}{\tau_v} \sigma_h - \omega_v^2 Q_h + \omega_v^2 E_h^2 \right) \phi dx, \ \forall \phi \in V_N^k \tag{2.14f}$$

with constitutive law

$$\int_{I_N^j} D_h \phi dx = \int_{I_N^j} \epsilon_0 (\epsilon_\infty E_h + P_h + a(1-\theta) E_h^3 + a\theta Q_h E_h) \phi dx, \ \forall \phi \in V_N^k. \tag{2.15}$$

Both terms $\widehat{E_h}$ and $\widetilde{H_h}$ are numerical fluxes. In our numerical experiments, we pick one of the following as our numerical flux [1]:

central flux

$$\widehat{E_h} = \{E_h\}, \quad \widetilde{H_h} = \{H_h\}; \tag{2.16}$$

alternating flux either alternating flux I

$$\widehat{E_h} = E_h^+, \widetilde{H_h} = H_h^- \tag{2.17a}$$

or alternating flux II

$$\widehat{E_h} = E_h^-, \widetilde{H_h} = H_h^+ \tag{2.17b}$$

dissipative flux, or upwind flux, inspired by the upwind flux for the Maxwell system without Kerr, linear Lorentz and Raman effects

$$\widehat{E_h} = \{E_h\} + \frac{1}{2}\sqrt{\frac{\mu_0}{\epsilon_0\epsilon_\infty}}[H_h], \quad \widetilde{H_h} = \{H_h\} + \frac{1}{2}\sqrt{\frac{\epsilon_0\epsilon_\infty}{\mu_0}}[E_h]. \tag{2.18}$$

Recall that under the assumption of periodic boundary condition, energy $\mathcal{E} = \mathcal{E}(t)$ of system (2.11) is

$$\mathcal{E} = \int_\Omega \left(\frac{\mu_0}{2}H^2 + \frac{\epsilon_0\epsilon_\infty}{2}E^2 + \frac{\epsilon_0}{2\omega_p^2}J^2 + \frac{\epsilon_0\omega_0^2}{2\omega_p^2}P^2 \right. \tag{2.19a}$$

$$\left. + \frac{\epsilon_0 a\theta}{4\omega_v^2}\sigma^2 + \frac{\epsilon_0 a\theta}{4}Q^2 + \frac{\epsilon_0 a\theta}{2}QE^2 + \frac{3\epsilon_0 a(1-\theta)}{4}E^4 \right) dx \tag{2.19b}$$

and its derivative

$$\frac{d}{dt}\mathcal{E} = -\frac{\epsilon_0}{\omega_p^2\tau}\int_\Omega J^2 dx - \frac{\epsilon_0 a\theta}{2\omega_v^2\tau_v}\int_\Omega \sigma^2 dx \le 0, \tag{2.20}$$

indicates that energy $\mathcal{E}$ is non-increasing. Besides, energy $\mathcal{E}$ is non-negative if $\theta \in [0, 3/4]$. Similar results could be obtained for semi-discrete DG method:

Theorem 2.2.1 (Semi-Discrete Stability [1]) *Under periodic boundary condition and the three fluxes* (2.16)(2.17)(2.18) *above, the semi-discrete numerical method* (2.14) *and the discrete energy* $\mathcal{E}_h$ *satisfies* $\frac{d}{dt}\mathcal{E}_h \le 0$. *In addition,* $\mathcal{E}_h \ge 0$ *if* $\theta \in [0, 3/4]$. *The discrete energy* $\mathcal{E}_h$ *is*

$$\mathcal{E}_h = \int_\Omega \left(\frac{\mu_0}{2}H_h^2 + \frac{\epsilon_0\epsilon_\infty}{2}E_h^2 + \frac{\epsilon_0}{2\omega_p^2}J_h^2 + \frac{\epsilon_0\omega_0^2}{2\omega_p^2}P_h^2 \right. \tag{2.21a}$$

$$\left. + \frac{\epsilon_0 a\theta}{4\omega_v^2}\sigma_h^2 + \frac{\epsilon_0 a\theta}{4}Q_h^2 + \frac{\epsilon_0 a\theta}{2}Q_h E_h^2 + \frac{3\epsilon_0 a(1-\theta)}{4}E_h^4 \right) dx \tag{2.21b}$$

With proper assumption of regularity of exact solution, using argument of certain L^2 projection or Gauss-Ladau projection, it was proved that the semi-discrete method converges to exact solution, and optimal convergence rate is obtained for alternating and upwind flux.

Theorem 2.2.2 (Error estimates of semi-discrete method [1]) *Under periodic boundary condition, assume the following regularity of exact solution of system* (2.11)

$$E, H, P, Q, J, \sigma \in W^{1,\infty}([0,T]; H^{k+1}(\Omega)),$$

$$E \in W^{1,\infty}([0,T];W^{1,\infty}(\Omega)),\ Q \in W^{1,\infty}([0,T];L^{\infty}(\Omega)),$$

with sufficiently small a and θ, for numerical scheme (2.14)*, we have*

$$||u - u_h|| \leq Ch^r, \quad u = E, H, P, Q, J, \sigma, \tag{2.22}$$

where $r = k$ for central flux (2.16)*, and $r = k + 1$ for alternating* (2.17) *or upwind flux* (2.18)*.*

2.2.3 Energy-Stable DG Method of the Maxwell Equation

On temporal discretization, a main focus is provable energy stability of fully discrete method. This turns out to be a nontrivial task for the nonlinear Maxwell equation. Common choices, such as the second order leap-frog or implicit trapezoidal method, may not yield provable stability results as for the linear models [1], while the main difficulties arise from the nonlinear Kerr and Raman terms. Here we introduce two fully discrete methods: one can be understood as novel modifications of leap-frog or implicit trapezoidal method, and the other is a fully implicit method. These temporal discretizations are of formal second order accuracy.

The leap-frog style method [1] is as following: given functions $u_h^n \in V_N^k$ at time $t = t^n$, with $u = H, D, E, P, J, Q, \sigma$, find $u_h^{n+1} \in V_N^k$ of all u at $t^{n+1} = t^n + \Delta t$, so that for any j and

each cell I_N^j, and any $\phi \in V_N^k$,

$$\mu_0 \int_{I_N^j} \frac{H_h^{n+1/2} - H_h^n}{\Delta t/2} \phi dx + \int_{I_N^j} E_h^n \partial_x \phi dx - (\widehat{E_h^n} \phi^-)_{j+1/2} + (\widehat{E_h^n} \phi^+)_{j-1/2} = 0 \tag{2.23a}$$

$$\int_{I_N^j} \frac{D_h^{n+1} - D_h^n}{\Delta t} \phi dx + \int_{I_N^j} H_h^{n+1/2} \partial_x \phi dx - (\widetilde{H_h^{n+1/2}} \phi^-)_{j+1/2} + (\widetilde{H_h^{n+1/2}} \phi^+)_{j-1/2} = 0 \tag{2.23b}$$

$$\int_{I_N^j} D_h^{n+1} \phi dx = \int_{I_N^j} \epsilon_0 (\epsilon_\infty E_h^{n+1} + P_h^{n+1} + a(1-\theta) Y_h^{n+1} + a\theta Q_h^{n+1} E_h^{n+1}) \phi dx \tag{2.23c}$$

$$\int_{I_N^j} Y_h^{n+1} \phi dx = \int_{I_N^j} \left(Y_h^n + \frac{3}{2} [(E_h^{n+1})^2 + (E_h^n)^2](E_h^{n+1} - E_h^n) \right) \phi dx \tag{2.23d}$$

$$\frac{P_h^{n+1} - P_h^n}{\Delta t} = \frac{J_h^{n+1} + J_h^n}{2} \tag{2.23e}$$

$$\frac{J_h^{n+1} - J_h^n}{\Delta t} = -\frac{1}{\tau} \frac{J_h^{n+1} + J_h^n}{2} - \omega_0^2 \frac{P_h^{n+1} + P_h^n}{2} + \omega_p^2 \frac{E_h^{n+1} + E_h^n}{2} \tag{2.23f}$$

$$\frac{Q_h^{n+1} - Q_h^n}{\Delta t} = \frac{\sigma_h^{n+1} + \sigma_h^n}{2} \tag{2.23g}$$

$$\int_{I_N^j} \frac{\sigma_h^{n+1} - \sigma_h^n}{\Delta t} \phi dx = -\int_{I_N^j} \left(\frac{1}{\tau_v} \frac{\sigma_h^{n+1} + \sigma_h^n}{2} - \omega_v^2 \frac{Q_h^{n+1} + Q_h^n}{2} + \omega_v^2 E_h^n E_h^{n+1} \right) \phi dx \tag{2.23h}$$

$$\mu_0 \int_{I_N^j} \frac{H_h^{n+1} - H_h^{n+1/2}}{\Delta t/2} \phi dx + \int_{I_N^j} E_h^{n+1} \partial_x \phi dx - (\widehat{\widehat{E_h^{n+1}}} \phi^-)_{j+1/2} + (\widehat{\widehat{E_h^{n+1}}} \phi^+)_{j-1/2} = 0 \tag{2.23i}$$

Be aware that the flux $\widehat{\widehat{E^{n+1}}}$ is similar to $\widehat{E^n}$, or the same as $\widehat{E^{n+1}}$, for central and alternating fluxes; for upwind flux, we have

$$\widehat{\widehat{E^{n+1}}} = \{E_h^{n+1}\} + \frac{1}{2} \sqrt{\frac{\mu_0}{\epsilon_0 \epsilon_\infty}} [H_h^{n+1/2}].$$

The fully implicit method [1] is as following, with the same set-up as the leap-frog

method above:

$$\mu_0 \int_{I_N^j} \frac{H_h^{n+1}-H_h^n}{\Delta t}\phi dx + \int_{I_N^j} \frac{E_h^{n+1}+E_h^n}{2}\partial_x \phi dx - (\widehat{\frac{E_h^{n+1}+E_h^n}{2}}\phi^-)_{j+1/2} + (\widehat{\frac{E_h^{n+1}+E_h^n}{2}}\phi^+)_{j-1/2} = 0 \quad (2.24a)$$

$$\int_{I_N^j} \frac{D_h^{n+1}-D_h^n}{\Delta t}\phi dx + \int_{I_N^j} \frac{H_h^{n+1}+H_h^n}{2}\partial_x \phi dx - (\widetilde{\frac{H_h^{n+1}+H_h^n}{2}}\phi^-)_{j+1/2} + (\widetilde{\frac{H_h^{n+1}+H_h^n}{2}}\phi^+)_{j-1/2} = 0 \quad (2.24b)$$

$$\int_{I_N^j} D_h^{n+1}\phi dx = \int_{I_N^j} \epsilon_0(\epsilon_\infty E_h^{n+1} + P_h^{n+1} + a(1-\theta)Y_h^{n+1} + a\theta Q_h^{n+1}E_h^{n+1})\phi dx \quad (2.24c)$$

$$\int_{I_N^j} Y_h^{n+1}\phi dx = \int_{I_N^j} \left(Y_h^n + \frac{3}{2}[(E_h^{n+1})^2 + (E_h^n)^2](E_h^{n+1}-E_h^n)\right)\phi dx \quad (2.24d)$$

$$\frac{P_h^{n+1}-P_h^n}{\Delta t} = \frac{J_h^{n+1}+J_h^n}{2} \quad (2.24e)$$

$$\frac{J_h^{n+1}-J_h^n}{\Delta t} = -\frac{1}{\tau}\frac{J_h^{n+1}+J_h^n}{2} - \omega_0^2\frac{P_h^{n+1}+P_h^n}{2} + \omega_p^2\frac{E_h^{n+1}+E_h^n}{2} \quad (2.24f)$$

$$\frac{Q_h^{n+1}-Q_h^n}{\Delta t} = \frac{\sigma_h^{n+1}+\sigma_h^n}{2} \quad (2.24g)$$

$$\int_{I_N^j} \frac{\sigma_h^{n+1}-\sigma_h^n}{\Delta t}\phi dx = -\int_{I_N^j} \left(\frac{1}{\tau_v}\frac{\sigma_h^{n+1}+\sigma_h^n}{2} - \omega_v^2\frac{Q_h^{n+1}+Q_h^n}{2} + \omega_v^2 E_h^n E_h^{n+1}\right)\phi dx \quad (2.24h)$$

We can establish the energy stability for the resulting fully discrete methods as following.

Theorem 2.2.3 (Fully Discrete Stability [1]) *Under periodic boundary condition and any of the three numerical fluxes, the discrete energy is non-increasing, i.e.* $\mathcal{E}_h^{n+1} \le \mathcal{E}_h^n$. *In addition,* $\mathcal{E}_h^n \ge 0$, *if* $\theta \in [0, 3/4]$ *and CFL condition* $\Delta t \le Ch$ *are satisfied. Here* C *is a constant depending on constants* $\mu_0, \epsilon_0, \epsilon_\infty$ *of the Maxwell equation, polynomial degree* k, *and choice of numerical flux. The discrete energy* $\mathcal{E}_h$ *at* $t = t^n$ *is*

$$\mathcal{E}_h^n = \int_\Omega \left(\frac{\mu_0}{2}(H_h^n)^2 + \frac{\epsilon_0\epsilon_\infty}{2}E_h^2 + \frac{\epsilon_0}{2\omega_p^2}J_h^2 + \frac{\epsilon_0\omega_0^2}{2\omega_p^2}P_h^2 + \frac{\epsilon_0 a\theta}{4\omega_v^2}\sigma_h^2 + \frac{\epsilon_0 a\theta}{4}Q_h^2 + \frac{\epsilon_0 a\theta}{2}Q_h E_h^2 + \frac{3\epsilon_0 a(1-\theta)}{4}E_h^4\right)dx \quad (2.25)$$

for fully implicit method, and

$$\mathcal{E}_h^n = \int_\Omega \left(\frac{\mu_0}{2} H_h^{n+1/2} H_h^{n-1/2} + \frac{\epsilon_0 \epsilon_\infty}{2} E_h^2 + \frac{\epsilon_0}{2\omega_p^2} J_h^2 + \frac{\epsilon_0 \omega_0^2}{2\omega_p^2} P_h^2 + \frac{\epsilon_0 a\theta}{4\omega_v^2} \sigma_h^2 + \frac{\epsilon_0 a\theta}{4} Q_h^2 + \frac{\epsilon_0 a\theta}{2} Q_h E_h^2 + \frac{3\epsilon_0 a(1-\theta)}{4} E_h^4 \right) dx \quad (2.26)$$

for leap-frog method.

2.2.4 Interpolatory Multiwavelets

In addition to Alpert's basis for SGDG space V_N^k, we need interpolatory multiwavelets to obtain a second set of basis, to deal with nonlinear terms, e.g. $f(u_h)$ when $u_h \in V_N^k$ and f is nonlinear [7, 5, 6].

Alpert's multiwavelets and the space W_n^k are constructed, corresponding to the difference of the L^2 projection on adjacent levels. Instead, the sparse grid collocation basis proposed in [7] corresponds to interpolation on nested grids. Denote the set of interpolation points in the interval $I = [0, 1]$ at mesh level 0 by $X_0 = \{x_i\}_{i=0}^P \subset I$, where the number of interpolation points in X_0 is $P + 1$. Then the interpolation points at mesh level $n \geq 1$, X_n, can be obtained in the following manner:

$$X_n = \{x_{i,n}^j := 2^{-n}(x_i + j), i = 0, \ldots, P, j = 0, \ldots, 2^n - 1\}.$$

In order to save computational cost, the points on different level of X_n should be nested, i.e.,

$$X_0 \subset X_1 \subset X_2 \subset X_3 \subset \cdots,$$

which can be achieved by requiring $X_0 \subset X_1$.

Given the interpolation points, we define the basis functions on the 0-th level grid as Lagrange $(K = 0)$ or Hermite $(K \geq 1)$ interpolation polynomials of degree less than or equal to $M := (P + 1)(K + 1) - 1$ which satisfy the property

$$\phi_{i,l}^{(l')}(x_{i'}) = \delta_{ii'}\delta_{ll'}.$$

for $i, i' = 0, ..., P$ and $l, l' = 0, ..., K$. It is easy to see that span$\{\phi_{i,l} : i = 0, \ldots, P,\ l = 0, \ldots, K\} = V_0^M$. The constants P, K, M will be specified later, depending on the Alpert's multiwavelets that the interpolatory multiwavelets relate to and the problem to solve. With the basis function at mesh level 0, we can define basis function at mesh level $n \geq 1$:

$$\phi_{i,l,n}^j := 2^{-nl}\phi_{i,l}(2^n x - j), \quad i = 0, \ldots, P,\ l = 0, \ldots, K,\ j = 0, \ldots, 2^n - 1,$$

which is a complete basis set for V_n^M.

Now we introduce the hierarchical representations based on the nested structure of interpolation points. Define $\tilde{X}_0 := X_0$ and $\tilde{X}_n := X_n \setminus X_{n-1}$ for $n \geq 1$; then we have the decomposition

$$X_n = \bigcup_{i=0}^{n} \tilde{X}_i.$$

Denote the points in $\tilde{X}_1$ by $\tilde{X}_1 = \{\tilde{x}_i\}_{i=0}^P$. Then the points in $\tilde{X}_n$ for $n \geq 1$ can be represented by

$$\tilde{X}_n = \{\tilde{x}_{i,n}^j := 2^{-(n-1)}(\tilde{x}_i + j) \ :\ i = 0, \ldots, P,\ j = 0, \ldots, 2^n - 1\}.$$

For notational convenience, we let $\widetilde{W}_0^M = V_0^M$. The increment function space $\widetilde{W}_n^M$ for $n \geq 1$ is introduced as a function space that satisfies

$$V_n^M = V_{n-1}^M \bigoplus \widetilde{W}_n^M,$$

and is defined through the multiwavelets $\psi_{i,l} \in V_1^M$ that satisfy

$$\psi_{i,l}^{(l')}(x_{i'}) = 0, \quad \psi_{i,l}^{(l')}(\tilde{x}_{i'}) = \delta_{ii'}\delta_{ll'}$$

for $i, i' = 0, ..., P$ and $l, l' = 0, ..., K$. Here the superscript l' denotes the l'-th order derivative. Then $\widetilde{W}_n^M$ is given by

$$\widetilde{W}_n^M = span\{\psi_{i,l,n}^j, \quad i = 0, ..., P,\ l = 0, ..., K,\ j = 0, ..., 2^{n-1} - 1\},$$

where

$$\psi_{i,l,n}^j(x) := 2^{-(n-1)l}\psi_{i,l}(2^{n-1}x - j).$$

For completeness, we list the basis functions used in this thesis in the following, and we focus on Hermite interpolation here [6]; for more possible choices of basis, see [6, 7]. The basis functions in $\widetilde{W}_1$ are piecewise polynomials on $I_l = (0, 1/2)$ and $I_r = (1/2, 1)$. Note that the functions may be discontinuous at the interface $x = 1/2$; thus I_l and I_r are both defined to be open intervals. The basis functions in $\widetilde{W}_1$ here are all supported on one half interval I_l or I_r and vanish on the other half. For simplicity, we will only declare the function on its support. For example, $\psi_0(x)|_{I_r}$ gives the definition of ψ_0 on I_r while vanishing on I_l. The interpolation points are put at the cell interface

$$\widetilde{X}_0 = \{0^+, 1^-\}, \quad \widetilde{X}_1 = \{\left(\frac{1}{2}\right)^+, \left(\frac{1}{2}\right)^-\}$$

Here and below, we use superscripts $+, -$ to emphasize the left and right limits of a function at that point, which is a feature of the discontinuous piecewise polynomial space. The basis functions in $\widetilde{W}_0^3$ and $\widetilde{W}_1^3$ are, when $P = 1, K = 1$,

$$\phi_{0,0}(x) = (x-1)^2(2x+1), \phi_{1,0}(x) = -x^2(2x-3),$$

$$\phi_{0,1}(x) = x(x-1)^2, \phi_{1,1}(x) = x^2(x-1),$$

and

$$\psi_{0,0}(x)|_{I_l} = -4x^2(4x-3), \psi_{1,0}(x)|_{I_r} = 4(x-1)^2(4x-1),$$

$$\psi_{0,1}(x)|_{I_l} = 2x^2(2x-1), \psi_{1,1}(x)|_{I_r} = 2(x-1)^2(2x-1).$$

When $P = 1, K = 2$, the basis functions in $\widetilde{W}_0^5$ and $\widetilde{W}_1^5$ are

$$\phi_{0,0}(x) = -(x-1)^3(6x^2+3x+1), \phi_{0,1}(x) = -x(x-1)^3(3x+1),$$

$$\phi_{0,2}(x) = -\frac{1}{2}x^2(x-1)^3, \phi_{1,0}(x) = x^3(6x^2-15x+10),$$

$$\phi_{1,1}(x) = -x^3(x-1)(3x-4), \phi_{1,2}(x) = \frac{1}{2}x^3(x-1)^2,$$

and

$$\psi_{0,0}(x)|_{I_l} = 16x^3(12x^2-15x+5), \psi_{1,0}(x)|_{I_r} = -16(x-1)^3(12x^2-9x+2),$$

$$\psi_{0,1}(x)|_{I_l} = -8x^3(2x-1)(3x-2), \psi_{1,1}(x)|_{I_r} = -8(x-1)^3(2x-1)(3x-1),$$

$$\psi_{0,2}(x)|_{I_l} = x^3(2x-1)^2, \psi_{1,2}(x)|_{I_r} = -(x-1)^3(2x-1)^2.$$

The construction above has close connection with interpolation operators. For a given function $f(x) \in C^{K+1}(I)$, we define $I_N^{P,K}[f]$ as the standard Hermite interpolation on V_N^M, and we have the representation

$$I_N^{P,K}[f](x) = \sum_{n=0}^{N} \sum_{j=0}^{\max(2^{n-1}-1,0)} \sum_{l=0}^{K} \sum_{i=0}^{P} b_{i,l,n}^j \psi_{i,l,n}^j(x).$$

Clearly, $(I_N^{P,K} - I_{N-1}^{P,K})[f](x)$ is in $\widetilde{W}_N^M$. The algorithm converting between the point values and the derivatives $\{f^{(l)}(x_{i,n}^j)\}$ to hierarchical coefficients $\{b_{i,l,n}^j\}$ is given in [7], and by a standard argument in fast wavelet transform, can be performed in $O(M2^n)$ flops.

2.2.5 SGDG Method with Multiresolution Interpolation in One Dimension

As discussed in section (2.2.3), we derive the energy stable DG method. To corporate it with multiresolution interpolation [6], we need to decide how to numerically calculate terms

$$\int_0^1 f(U_h^{n+1}, U_h^n)\phi dx, \quad \forall \phi \in V_N^k, \tag{2.27}$$

in fully discrete numerical scheme (2.23) and (2.24), where

$$U = (H, D, E, P, J, Q, \sigma)$$

and $f(U_h^{n+1}, U_h^n)$ takes one of the following up to some constants:

$$[(E_h^{n+1})^2 + (E_h^n)^2](E_h^{n+1} - E_h^n), \quad E_h^n E_h^{n+1}, \quad Q_h^{n+1} E_h^{n+1}.$$

Notice that nonlinear terms of the Maxwell equation discussed in the thesis occur as part of integrand of integrals over spatial domain. The fully discrete SGDG schemes with interpolation are to replace nonlinear term in form of (2.27) to

$$\int_0^1 \mathcal{I}\left[f(U_h^{n+1}, U_h^n)\right]\phi dx \tag{2.28}$$

where $\mathcal{I}[\cdot]$ is a multiresolution interpolation operator $\mathcal{I}_N^{P,K}$ as described in previous section, onto finite element space V_N^M with the same multiresolution structure as V_N^k, but of polynomial degree $M = (P+1)(K+1) - 1$, as we discuss above.

To elaborate the implementation of such algorithm regarding treatment to nonlinear terms, assume we know both U_h^{n+1} and U_h^n for an implicit scheme. We first read the (derivative) values of U_h^{n+1} and U_h^n, each component of which is a linear combination of Alpert's basis functions at the chosen interpolation points. We then calculate the (derivative) values of f at interpolation points of interpolatory multiwavelet basis. Last, we transform the (derivative) values back to coefficients of (Alpert's or interpolatory) multiwavelet basis, by using the algorithm introduced in [7]. Such numerical algorithm can be performed through a fast matrix-vector product as in [6, 84]. As stated in [6], we remark that the computational cost does not increase too much compared to the multiresolution DG schemes for linear equations [5]. The cost of the transformation from the (derivative) values to hierarchical coefficients is only linearly dependent on the dimension d [7].

2.3 Formulation of Energy Stable Adaptive SGDG Method in Higher Dimension

2.3.1 Semi-Discrete and Fully-Discrete Energy Stable DG Method

As mentioned before in (2.7) and (2.8), we consider the following Maxwell system for vector-valued $\mathbf{H}, \mathbf{D}, \mathbf{E}, \mathbf{P}, \mathbf{J}$ and scalar Q, σ, on $(0,T) \times \Omega$,

$$\mu_0 \partial_t \mathbf{H} = -\nabla \times \mathbf{E} \tag{2.29a}$$

$$\partial_t \mathbf{D} = \nabla \times \mathbf{H} \tag{2.29b}$$

$$\partial_t \mathbf{P} = \mathbf{J} \tag{2.29c}$$

$$\partial_t \mathbf{J} = -\gamma \mathbf{J} - \omega_0^2 \mathbf{P} + \omega_p^2 \mathbf{E} \tag{2.29d}$$

$$\partial_t Q = \sigma \tag{2.29e}$$

$$\partial_t \sigma = -\gamma_v \sigma - \omega_v^2 Q + \omega_v^2 |\mathbf{E}|^2 \tag{2.29f}$$

with constitutive law

$$\mathbf{D} = \epsilon_0(\epsilon_\infty \mathbf{E} + \mathbf{P} + a(1-\theta)|\mathbf{E}|^2\mathbf{E} + a\theta Q\mathbf{E}). \tag{2.30}$$

Here

$$\gamma = \frac{1}{\tau}, \quad \gamma_v = \frac{1}{\tau_v}.$$

For simplicity, we consider the method only for the two-dimension transverse electric (TE) mode. Extension to the full three-dimension model is straightforward [2]. Thus, we consider the two-dimension system of equations

$$\begin{aligned}
\mu_0 \partial_t H_z &= \partial_y E_x - \partial_x E_y && \text{(2.31a)}\\
\partial_t D_x &= \partial_y H_z && \text{(2.31b)}\\
\partial_t D_y &= -\partial_x H_z && \text{(2.31c)}\\
\mathbf{D} &= \epsilon_0(\epsilon_\infty \mathbf{E} + \mathbf{P} + a(1-\theta)|\mathbf{E}|^2\mathbf{E} + a\theta Q\mathbf{E}) && \text{(2.31d)}\\
\partial_t \mathbf{P} &= \mathbf{J} && \text{(2.31e)}\\
\partial_t \mathbf{J} &= -\gamma \mathbf{J} - \omega_0^2 \mathbf{P} + \omega_p^2 \mathbf{E} && \text{(2.31f)}\\
\partial_t Q &= \sigma && \text{(2.31g)}\\
\partial_t \sigma &= -\gamma_v \sigma - \omega_v^2 Q + \omega_v^2 |\mathbf{E}|^2 && \text{(2.31h)}
\end{aligned}$$

Here, all vector fields have two components polarized in the x-y plane, e.g. $\mathbf{D} = (D_x, D_y)$, $\mathbf{E} = (E_x, E_y)$, $\mathbf{P} = (P_x, P_y)$, and $\mathbf{J} = (J_x, J_y)$, except $\mathbf{H} = H_z$. Q and σ are scalar components.

Assume computational domain, now on x-y plane, is $\Omega = [0,1] \times [0,1]$, and consider the uniform mesh for Sparse Grid DG purpose. Define

$$x_{i+\frac{1}{2}} = \frac{i}{2^N}, \quad y_{j+\frac{1}{2}} = \frac{j}{2^N}, \quad i,j = 0,1,\ldots,2^N.$$

Then $\{x_{i+1/2}\}$ and $\{y_{j+1/2}\}$ divide $[0,1]$ on x and y direction uniformly into 2^N many cells. Let

$$I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}], \quad J_j = [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}], \quad K_{ij} = I_i \times J_j, \quad \forall i,j,$$

and $\{K_{ij}\}_{i,j}$ is a partition or mesh of Ω. Each cell K_{ij} has cell center

$$(x_i, y_j) = (\frac{x_{i-\frac{1}{2}} + x_{i+\frac{1}{2}}}{2}, \frac{y_{j-\frac{1}{2}} + y_{j+\frac{1}{2}}}{2}) = (\frac{i-1/2}{2^N}, \frac{j-1/2}{2^N}),$$

and grid size is $\Delta x = \Delta y = 2^{-N}$.

Regarding the mesh, we define

$$V_h^k = \left\{ v \in L^2(\Omega) : v|_{K_{ij}} \in Q^k(K_{ij}),\ \forall i, j \right\},$$

where $Q^k(K_{ij})$ consists of polynomials with degree up to k in each variable on K_{ij}. Without confusion, V_h^k is also used to represent its vector version in the following. For any function $v \in V_h^k$, we write

$$v(x^{\pm}_{i+\frac{1}{2}}, y) = \lim_{\epsilon \to 0^{\pm}} v(x_{i+\frac{1}{2}} + \epsilon, y), \quad v(x, y^{\pm}_{j+\frac{1}{2}}) = \lim_{\epsilon \to 0^{\pm}} v(x, y_{j+\frac{1}{2}} + \epsilon),$$

which are left or right limit on cell boundary of x or y direction. Average and jump on cell interface $x = x_{i+1/2}$ is

$$\{v\}_{x_{i+\frac{1}{2}}} = \frac{1}{2}\Big(v(x^+_{i+\frac{1}{2}}, y) + v(x^-_{i+\frac{1}{2}}, y)\Big), \quad [v]_{x_{i+\frac{1}{2}}} = v(x^+_{i+\frac{1}{2}}, y) - v(x^-_{i+\frac{1}{2}}, y),$$

and on cell interface $y = y_{j+1/2}$ is

$$\{v\}_{y_{j+\frac{1}{2}}} = \frac{1}{2}\Big(v(x, y^+_{j+\frac{1}{2}}) + v(x, y^-_{j+\frac{1}{2}})\Big), \quad [v]_{y_{j+\frac{1}{2}}} = v(x, y^+_{j+\frac{1}{2}}) - v(x, y^-_{j+\frac{1}{2}}).$$

A typical semi-discrete DG method is as following [2]: find each component of H_{zh}, $\mathbf{D}_h$, $\mathbf{E}_h$, $\mathbf{P}_h$, $\mathbf{J}_h$, Q_h and σ_h in V_h^k, under periodic boundary condition, such that,

$$\mu_0(\partial_t H_{zh}, \phi) + \mathcal{B}_h^{\mathbf{E}}(E_{xh}, E_{yh}, \phi) = 0, \quad \forall \phi \in V_h^k \tag{2.32a}$$

$$(\partial_t D_{xh}, \phi) + \mathcal{B}_{xh}^{H}(H_{zh}, \phi) = 0, \quad \forall \phi \in V_h^k, \tag{2.32b}$$

$$(\partial_t D_{yh}, \phi) + \mathcal{B}_{yh}^{H}(H_{zh}, \phi) = 0, \quad \forall \phi \in V_h^k, \tag{2.32c}$$

$$\mathbf{D}_h = \epsilon_0\Big(\epsilon_\infty \mathbf{E}_h + \mathbf{P}_h + a(1-\theta)\mathcal{I}_h\Big(|\mathbf{E}_h|^2 \mathbf{E}_h\Big) + a\theta \mathcal{I}_h(Q_h \mathbf{E}_h)\Big), \tag{2.32d}$$

$$\partial_t \mathbf{P}_h = \mathbf{J}_h \tag{2.32e}$$

$$\partial_t \mathbf{J}_h = -\gamma \mathbf{J}_h - \omega_0^2 \mathbf{P}_h + \omega_p^2 \mathbf{E}_h \tag{2.32f}$$

$$\partial_t Q_h = \sigma_h \tag{2.32g}$$

$$\partial_t \sigma_h = -\gamma_v \sigma_h - \omega_v^2 Q_h + \omega_v^2 \mathcal{I}_h(|\mathbf{E}_h|^2) \tag{2.32h}$$

where $\mathcal{I}_h$ is interpolation on Gauss-Legendre points, projecting nonlinear terms to DG space, $(\cdot,\cdot)$ the inner product on $\Omega=[0,1]^2$, and

$$\begin{aligned}
\mathcal{B}_h^{\mathbf{E}}(E_{xh},E_{yh},\phi) &= -(E_{yh},\partial_x\phi)+(E_{xh},\partial_y\phi)-\sum_{i=1}^{2^N}\int_0^1 \widehat{E_{yh}}(x_{i+\frac{1}{2}},y)[\phi]_{x_{i+\frac{1}{2}}}dy \\
&\quad +\sum_{j=1}^{2^N}\int_0^1 \widehat{\widehat{E_{xh}}}(x,y_{j+\frac{1}{2}})[\phi]_{y_{j+\frac{1}{2}}}dx, \qquad (2.33a)\\
\mathcal{B}_{xh}^{H}(H_{zh},\phi) &= \sum_{j=1}^{2^N}\int_0^1 \widetilde{\widetilde{H_{zh}}}(x,y_{j+\frac{1}{2}})[\phi]_{y_{j+\frac{1}{2}}}dx+(H_{zh},\partial_y\phi), \qquad (2.33b)\\
\mathcal{B}_{yh}^{H}(H_{zh},\phi) &= -\sum_{i=1}^{2^N}\int_0^1 \widetilde{H_{zh}}(x_{i+\frac{1}{2}},y)[\phi]_{x_{i+\frac{1}{2}}}dy-(H_{zh},\partial_x\phi). \qquad (2.33c)
\end{aligned}$$

We choose the fluxes to be central flux

$$\widehat{E_{yh}}(x_{i+\frac{1}{2}},y)=\{E_{yh}\}_{x_{i+\frac{1}{2}}},\ \widehat{\widehat{E_{xh}}}(x,y_{j+\frac{1}{2}})=\{E_{xh}\}_{y_{j+\frac{1}{2}}}, \qquad (2.34a)$$

$$\widetilde{H_{zh}}(x_{i+\frac{1}{2}},y)=\{H_{zh}\}_{x_{i+\frac{1}{2}}},\ \widetilde{\widetilde{H_{zh}}}(x,y_{j+\frac{1}{2}})=\{H_{zh}\}_{y_{j+\frac{1}{2}}}, \qquad (2.34b)$$

or alternating flux

$$\widehat{E_{yh}}(x_{i+\frac{1}{2}},y)=E_{yh}(x^{\dagger}_{i+\frac{1}{2}},y),\ \widehat{\widehat{E_{xh}}}(x,y_{j+\frac{1}{2}})=E_{xh}(x,y^{\natural}_{j+\frac{1}{2}}), \qquad (2.35a)$$

$$\widetilde{H_{zh}}(x_{i+\frac{1}{2}},y)=H_{zh}(x^{\ddagger}_{i+\frac{1}{2}},y),\ \widetilde{\widetilde{H_{zh}}}(x,y_{j+\frac{1}{2}})=H_{zh}(x,y^{\sharp}_{j+\frac{1}{2}}), \qquad (2.35b)$$

where

$$(\dagger,\ddagger),(\natural,\sharp)=(+,-)\text{ or }(-,+).$$

For the semi-discrete in space methods with the central or alternating numerical fluxes, one can establish an energy relation similar as for the continuous model. Additionally, error estimates can be proved and they are optimal with respect to the approximation property of the discrete space V_h^k when the numerical fluxes are alternating.

Theorem 2.3.1 (Semi-discrete in space energy stability [2]) *Under the assumption of periodic boundary conditions, the semi-discrete in space methods* (2.32)*, with either central flux* (2.34) *or alternating flux* (2.35)*, satisfy*

$$\frac{d\mathcal{E}_h(t)}{dt}=-\frac{\epsilon_0\gamma}{\omega_p^2}\int_\Omega|\mathbf{J}_h|^2d\Omega-\frac{\epsilon_0 a\theta\gamma_v}{2\omega_v^2}\int_\Omega\sigma_h^2d\Omega\le 0, \qquad (2.36)$$

with the discrete energy defined as

$$\begin{aligned}\mathcal{E}_h &= \int_\Omega \Bigg(\frac{\mu_0}{2} H_{zh}^2 + \frac{\epsilon_0 \epsilon_\infty}{2} |\mathbf{E}_h|^2 + \frac{\epsilon_0}{2\omega_p^2} |\mathbf{J}_h|^2 + \frac{\epsilon_0 \omega_0^2}{2\omega_p^2} |\mathbf{P}_h|^2 + \frac{\epsilon_0 a\theta}{4\omega_v^2} \sigma_h^2 \\ &\quad + \frac{\epsilon_0 a\theta}{2} \mathcal{I}_h \left(Q_h |\mathbf{E}_h|^2 \right) + \frac{3\epsilon_0 a(1-\theta)}{4} \mathcal{I}_h \left(|\mathbf{E}_h|^4 \right) + \frac{\epsilon_0 a\theta}{4} Q_h^2 \Bigg) d\Omega. \end{aligned} \tag{2.37}$$

Meanwhile, when $\theta \in [0, 3/4]$, $\mathcal{E}_h \geq 0$.

Theorem 2.3.2 (Semi-discrete in space error estimates [2]) *Let* $T > 0$ *be given. Let* $\kappa_{err}, \rho_{err} \in (0, 1)$ *be arbitrary constants, then under periodic boundary conditions and*

1. $\theta \in \left[0, \dfrac{1}{1 + 3(1 - \rho_{err})^{-2}} \right]$,
2. $a\theta C_k \|Q\|_\infty \leq \epsilon_\infty (1 - \kappa_{err})$, *and*
3. $a \left(12(1-\theta) C_k^2 \|\mathbf{E}\|_\infty \|\partial_t \mathbf{E}\|_\infty + (12 - 11\theta) \dfrac{C_k^2}{\rho_{err}} \|\partial_t \mathbf{E}\|_\infty^2 + 2\theta C_k \|\partial_t Q\|_\infty \right) \leq \epsilon_\infty \kappa_{err}$,

as well as the exact solution being sufficiently smooth, the numerical solution u_h *given by 2.32 with suitable initialization admits the following error estimate at time* T

$$\|u - u_h\|_{L^2(\Omega)} \leq C C(\kappa_{err}, \rho_{err}) h^r, u = H_z, \mathbf{E}, \mathbf{P}, \mathbf{J}, \sigma, Q, \tag{2.38}$$

where $r = k$ *for central flux* (2.34)*, and* $r = k + 1$ *for alternating flux* (2.35)*. Here* C *is a generic constant independent of mesh size* h*, but may depend on* k*, the mesh parameter* δ*, the model parameters, and some Sobolev norm of the exact solutions up to time* T*.*

Following the semi-discrete method, the fully-discrete leap-frog DG scheme is, given H_{zh}^n, $\mathbf{E}_h^n$, $\mathbf{D}_h^n$, $\mathbf{J}_h^n$, $\mathbf{P}_h^n$, σ_h^n and Q_n^h in V_h^k at time t^n, we find H_{zh}^{n+1}, $\mathbf{E}_h^{n+1}$, $\mathbf{D}_h^{n+1}$, $\mathbf{J}_h^{n+1}$, $\mathbf{P}_h^{n+1}$,

σ_h^{n+1} and $Q_{n+1}^h \in V_h^k$ at time $t^{n+1} = t^n + \Delta t$, satisfying

$$\mu_0 \left(\frac{H_{zh}^{n+1/2} - H_{zh}^n}{\Delta t/2}, \phi \right) + \mathcal{B}_h^{\mathbf{E}}(E_{xh}^n, E_{yh}^n, \phi) = 0, \quad \forall \phi \in V_h^k, \tag{2.39a}$$

$$\left(\frac{D_{xh}^{n+1} - D_{xh}^n}{\Delta t}, \phi \right) + \mathcal{B}_{xh}^H(H_{zh}^{n+1/2}, \phi) = 0, \quad \forall \phi \in V_h^k, \tag{2.39b}$$

$$\left(\frac{D_{yh}^{n+1} - D_{yh}^n}{\Delta t}, \phi \right) + \mathcal{B}_{yh}^H(H_{zh}^{n+1/2}, \phi) = 0, \quad \forall \phi \in V_h^k, \tag{2.39c}$$

$$\mathbf{D}_h^{n+1} = \epsilon_0 \left(\epsilon_\infty \mathbf{E}_h^{n+1} + \mathbf{P}_h^{n+1} + a(1-\theta)\mathbf{Y}_h^{n+1} + a\theta \mathcal{I}_h \left(Q_h^{n+1} \mathbf{E}_h^{n+1} \right) \right), \tag{2.39d}$$

$$\begin{aligned} \mathbf{Y}_h^{n+1} = \mathbf{Y}_h^n &+ \mathcal{I}_h \left(\left(|\mathbf{E}_h^{n+1}|^2 + |\mathbf{E}_h^n|^2 - \mathbf{E}_h^{n+1} \cdot \mathbf{E}_h^n \right) \left(\mathbf{E}_h^{n+1} - \mathbf{E}_h^n \right) \right) \\ &+ \frac{1}{2} \mathcal{I}_h \left((\mathbf{E}_h^{n+1} + \mathbf{E}_h^n) \cdot (\mathbf{E}_h^{n+1} - \mathbf{E}_h^n)(\mathbf{E}_h^{n+1} + \mathbf{E}_h^n) \right), \end{aligned} \tag{2.39e}$$

$$\frac{\mathbf{P}_h^{n+1} - \mathbf{P}_h^n}{\Delta t} = \frac{\mathbf{J}_h^{n+1} + \mathbf{J}_h^n}{2}, \tag{2.39f}$$

$$\frac{\mathbf{J}_h^{n+1} - \mathbf{J}_h^n}{\Delta t} + \gamma \frac{\mathbf{J}_h^{n+1} + \mathbf{J}_h^n}{2} + \omega_0^2 \frac{\mathbf{P}_h^{n+1} + \mathbf{P}_h^n}{2} = \omega_p^2 \frac{\mathbf{E}_h^{n+1} + \mathbf{E}_h^n}{2}, \tag{2.39g}$$

$$\frac{Q_h^{n+1} - Q_h^n}{\Delta t} = \frac{\sigma_h^{n+1} + \sigma_h^n}{2}, \tag{2.39h}$$

$$\frac{\sigma_h^{n+1} - \sigma_h^n}{\Delta t} + \gamma_v \frac{\sigma_h^{n+1} + \sigma_h^n}{2} + \omega_v^2 \frac{Q_h^{n+1} + Q_h^n}{2} = \omega_v^2 \mathcal{I}_h (\mathbf{E}_h^{n+1} \cdot \mathbf{E}_h^n), \tag{2.39i}$$

$$\mu_0 \left(\frac{H_{zh}^{n+1} - H_{zh}^{n+1/2}}{\Delta t/2}, \phi \right) + \mathcal{B}_h^{\mathbf{E}}(E_{xh}^{n+1}, E_{yh}^{n+1}, \phi) = 0, \quad \forall \phi \in V_h^k \tag{2.39j}$$

The terms of $\mathcal{B}_h^{\mathbf{E}}$, $\mathcal{B}_{xh}^H$, and $\mathcal{B}_{yh}^H$ are defined in (2.33), with either the central fluxes (2.34) or alternating fluxes in (2.35). Energy stability result for fully discrete scheme follows:

Theorem 2.3.3 (Fully-discrete energy stability [2]) *Under the assumption of periodic boundary conditions, the fully-discrete leap-frog nodal DG schemes* (2.39) *satisfy*

$$\mathcal{E}_h^{n+1} - \mathcal{E}_h^n = -\frac{\epsilon_0 \gamma \Delta t}{4\omega_p^2} \int_\Omega |\mathbf{J}_h^{n+1} + \mathbf{J}_h^n|^2 d\Omega - \frac{\epsilon_0 a \theta \gamma_v \Delta t}{8\omega_v^2} \int_\Omega (\sigma_h^{n+1} + \sigma_h^n)^2 d\Omega \leq 0, \tag{2.40}$$

with the discrete energy $\mathcal{E}_h^n$ defined as

$$\begin{aligned} \mathcal{E}_h^n \;=\; \int_\Omega \Big(&\frac{\mu_0}{2} H_{zh}^{n+1/2} H_{zh}^{n-1/2} + \frac{\epsilon_0 \epsilon_\infty}{2} |\mathbf{E}_h^n|^2 + \frac{\epsilon_0}{2\omega_p^2} |\mathbf{J}_h^n|^2 + \frac{\epsilon_0 \omega_0^2}{2\omega_p^2} |\mathbf{P}_h^n|^2 + \frac{\epsilon_0 a \theta}{4\omega_v^2} (\sigma_h^n)^2 \\ &+ \frac{\epsilon_0 a \theta}{2} \mathcal{I}_h \left(Q_h^n |\mathbf{E}_h^n|^2 \right) + \frac{3\epsilon_0 a(1-\theta)}{4} \mathcal{I}_h \left(|\mathbf{E}_h^n|^4 \right) + \frac{\epsilon_0 a \theta}{4} (Q_h^n)^2 \Big) d\Omega. \end{aligned} \tag{2.41}$$

In addition, $\mathcal{E}_h^n \geq 0$ if $\theta \in [0, 3/4]$ and $\Delta t \leq Ch$ for some constant C.

Similarly, we can formulate fully implicit scheme as in (2.39); however, update of H_{zh} at intermediate time step $n+1/2$ is unnecessary, so in fully implicit scheme, (2.39j) should be removed, while (2.39a),(2.39b),and (2.39c) should be replaced by

$$\mu_0\left(\frac{H_{zh}^{n+1}-H_{zh}^{n}}{\Delta t},\phi\right)+\mathcal{B}_h^{\mathbf{E}}(\frac{E_{xh}^{n+1}+E_{xh}^{n}}{2},\frac{E_{yh}^{n+1}+E_{yh}^{n}}{2},\phi)=0,\quad \forall\phi\in V_h^k \tag{2.42a}$$

$$\left(\frac{D_{xh}^{n+1}-D_{xh}^{n}}{\Delta t},\phi\right)+\mathcal{B}_{xh}^{H}(\frac{H_{zh}^{n+1}+H_{zh}^{n}}{2},\phi)=0,\quad \forall\phi\in V_h^k, \tag{2.42b}$$

$$\left(\frac{D_{yh}^{n+1}-D_{yh}^{n}}{\Delta t},\phi\right)+\mathcal{B}_{yh}^{H}(\frac{H_{zh}^{n+1}+H_{zh}^{n}}{2},\phi)=0,\quad \forall\phi\in V_h^k, \tag{2.42c}$$

All simulations of two dimension in Section 2.6 use fully implicit scheme (2.39d)-(2.39i) and (2.42).

2.3.2 High Dimensional Sparse Grid DG Method and Interpolatory Multiwavelets

Based on one dimensional Sparse Grid DG Method we discussed before, we are ready to define it in higher dimension $d>1$. First we recall some basic notations about multi-indexes. For a multi-index $\alpha=(\alpha_1,\ldots,\alpha_d)\in\mathbb{N}_0^d$, where $\mathbb{N}_0$ denotes the set of nonnegative integers, the l_1 and l_∞ norms are defined as

$$|\alpha|_1=\sum_{i=1}^{d}\alpha_i,\quad |\alpha|_\infty=\max_{1\le i\le d}\alpha_i.$$

The component-wise arithmetic operations are

$$\alpha\cdot\beta=(\alpha_1\beta_1,\ldots,\alpha_d\beta_d),\quad c\cdot\alpha=(c\alpha_1,\ldots,c\alpha_d),\quad 2^\alpha=(2^{\alpha_1},\ldots,2^{\alpha_d}),$$

while and relational operations are defined as

$$\alpha\le\beta\Longleftrightarrow\alpha_i\le\beta_i\ \forall i;\quad \alpha=\beta\Longleftrightarrow\alpha\le\beta\text{ and }\beta\le\alpha;$$

$$\alpha<\beta\Longleftrightarrow\alpha\le\beta\text{ and }\beta\le\alpha.$$

By making use of the multi-index notation, we denote by $\mathbf{l}=(l_1,\ldots,l_d)\in\mathbb{N}_0^d$, the mesh level in a multivariate sense. Define the tensor-product mesh grid

$$\Omega_{\mathbf{l}}=\Omega_{l_1}\bigotimes\Omega_{l_2}\bigotimes\ldots\bigotimes\Omega_{l_d},$$

as tensor product of one dimensional mesh, and the corresponding mesh size

$$\mathbf{h_l} = (h_{l_1}, \ldots, h_{l_d}).$$

For convenience, we also write h_{l_i} as h_i, for $i = 1, 2, \ldots, d$. Based on grid $\Omega_\mathbf{l}$, we denote by

$$I_\mathbf{l}^\mathbf{j} = (h_1 j_1, h_1(j_1+1)) \times (h_2 j_2, h_2(j_2+1)) \times \ldots \times (h_d j_d, h_d(j_d+1))$$

an elementary cell, and

$$\mathbf{V}_\mathbf{l}^k = \{\mathbf{v} : \mathbf{v}(\mathbf{x}) \in Q^k(I_\mathbf{l})^\mathbf{j},\ \mathbf{0} \le \mathbf{j} \le 2^\mathbf{l} - \mathbf{1}\} = V_{l_1,x_1}^k \times V_{l_2,x_2}^k \times \ldots \times V_{l_d,x_d}^k$$

the tensor-product piecewise polynomial space, where $Q^k(I_\mathbf{l}^\mathbf{j})$ denotes the collection of polynomials of degree up to k in each dimension on cell $I_\mathbf{l}^\mathbf{j}$, and V_{l_i,x_i}^k the piecewise k-th order polynomial space of variable x_i, with $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_d)$. If we use equal mesh refinement of size $h_N = 2^{-N}$ in each coordinate direction, the grid and space will be denoted by Ω_N and $\mathbf{V}_N^k$, respectively.

Based on a tensor-product construction, the multi-dimensional increment space can be defined as

$$\mathbf{W}_\mathbf{l}^k = W_{l_1,x_1}^k \times W_{l_2,x_2}^k \times \ldots \times W_{l_d,x_d}^k$$

Therefore, the standard tensor-product piecewise polynomial space on N can be written as

$$\mathbf{V}_N^k = \bigoplus_{|\mathbf{l}|_\infty \le N;\ \mathbf{l} \in \mathbb{N}_0^d} \mathbf{W}_\mathbf{l}^k,$$

while the sparse grid approximation space as discussed in e.g. [3] is

$$\widehat{\mathbf{V}}_N^k = \bigoplus_{|\mathbf{l}|_1 \le N;\ \mathbf{l} \in \mathbb{N}_0^d} \mathbf{W}_\mathbf{l}^k, \tag{2.43}$$

a subset of $\mathbf{V}_N^k$. The dimension of $\widehat{\mathbf{V}}_N^k$ scales as $O((k+1)^d 2^N N^{d-1})$ [3], which is significantly less than that of $\mathbf{V}_N^k$ with exponential dependence on Nd. The approximation results for $\widehat{\mathbf{V}}_N^k$ are discussed in e.g. [3], which has a stronger smoothness requirement than the traditional space $\mathbf{V}_N^k$. However, for adaptive scheme we will discuss later, we

will not require the numerical solution to be in $\widehat{\mathbf{V}}_N^k$, but rather in $\mathbf{V}_N^k$, and we can choose basis functions from $\mathbf{V}_N^k$ to solve the PDE problem in each time step adatpively.

Finally, we define the polynomial basis functions of degree k in multi-dimensions as

$$\mathbf{v}_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}(\mathbf{x}) = \prod_{m=1}^{d} v_{i_m,l_m}^{j_m}(x_m),$$

and for all m,

$$l_m \in \mathbb{N}_0, \quad 0 \le i_m \le k, \quad 0 \le j_m \le \max\{0, 2^{l_m-1} - 1\}.$$

The orthonormality of the basis is established in sections before. Furthermore, the support of $\mathbf{v}_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}$ of all $\mathbf{i}$ are $I_{\mathbf{l}'}^{\mathbf{j}}$, where $\mathbf{l}' = (l_1', \ldots, l_d')$ and $l_m' = \max\{l_m - 1, 0\}$.

Following the same manner, we can construct high-dimensional interpolatory multiwavelets. Let

$$\widetilde{\mathbf{W}}_{\mathbf{l}}^M = \widetilde{W}_{l_1,x_1}^M \times \widetilde{W}_{l_2,x_2}^M \times \ldots \times \widetilde{W}_{l_d,x_d}^M,$$

where $\widetilde{W}_{l_m,x_m}^M,\ m = 1, \ldots, d$ are one-dimensional interpolatory multiwavelet spaces of mesh level l_m and m-th dimension variable x_m, and M here is degree of polynomials of interpolatory multiwavelets. Recall that we might choose M as a larger integer than k. Then

$$\mathbf{V}_N^M = \bigoplus_{|\mathbf{l}|_\infty \le N;\ \mathbf{l} \in \mathbb{N}_0^d} \widetilde{\mathbf{W}}_{\mathbf{l}}^M,$$

while the sparse grid approximation space is

$$\widetilde{\mathbf{V}}_N^M = \bigoplus_{|\mathbf{l}|_1 \le N;\ \mathbf{l} \in \mathbb{N}_0^d} \widetilde{\mathbf{W}}_{\mathbf{l}}^M.$$

2.3.3 Adaptive SGDG Method

In this subsection, we formulate an adaptive multiresolution projection algorithm as discussed in [123] and [5, 6]. The last two references provide full details of adaptive SGDG scheme. Define an accuracy threshold or error threshold $\epsilon > 0$; we will use this threshold throughout this subsection when we demonstrate the adaptive scheme, in both refinement and coarsening procedures. In practice, we choose smaller accuracy threshold for

coarsening than refinement, usually $\delta = \varepsilon/10$, where ε and δ are error threshold for refinement and coarsening, respectively.

The backbone of the algorithm is the fact that each hierarchical basis, of space V_N^k in one dimension or of space $\mathbf{V}_N^k$ in higher dimension, represents some details of a solution or a function on a specific mesh scale, which naturally provides an error indicator for the design of adaptive algorithms. In one dimension, consider $u(x)$ has L^2 projection $\tilde{u}(x)$ on V_N^k supported on $\Omega = [0, 1]$, as linear combination of Alpert's multiwavelet basis functions:

$$\tilde{u}(x) = \sum_{i=0}^{k} \sum_{n=0}^{N} \sum_{j=0}^{\max\{0,2^{n-1}-1\}} u_{i,n}^j v_{i,n}^j(x), \tag{2.44}$$

where

$$u_{i,n}^j = \int_0^1 u(x) v_{i,n}^j(x) dx.$$

It was shown in [4] that

$$\sum_{i=0}^{k} \sum_{j=0}^{\max\{0,2^{n-1}-1\}} |u_{i,n}^j|^2 \leq C 2^{-(q+1)n} |u|_{\mathcal{H}^{q+1}(\Omega)},$$

if $u \in \mathcal{H}^{p+1}(\Omega)$, where seminorm $\mathcal{H}^{p+1}(\Omega)$ is

$$|u|_{\mathcal{H}^{p+1}(\Omega)} = \left|\left| \frac{d^{p+1}u}{dx^{p+1}} \right|\right|_{L^2(\Omega)},$$

and $q = \min\{p, k\}$. The hierarchial coefficients $u_{i,n}^j$ here (also called hierarchial surplus) naturally serves as an indicator for local smoothness of $u(x)$. The main idea of the adaptive algorithm is to choose only those basis functions or elements whose coefficients above a prescribed threshold value ϵ. Here we consider using L^2 norm of u as indicator, and since basis functions $v_{i,n}^j$ are orthnormal, this is equivalent to

$$\left(\sum_{i=0}^{k} |u_{i,n}^j|^2 \right)^{1/2}$$

for each fixed n and fixed $0 \leq j \leq \max\{0, 2^{n-1} - 1\}$. In summary, we would flag an element V_n^j if

$$\left(\sum_{i=0}^{k} |u_{i,n}^j|^2 \right)^{1/2} \geq \epsilon.$$

In higher dimension, projection $\tilde{u}$ of function u on $\mathbf{V}_N^k$ is

$$\tilde{u}(\mathbf{x}) = \sum_{\mathbf{0}\le\mathbf{i}\le\mathbf{k}} \sum_{\mathbf{0}\le\mathbf{l}\le\mathbf{N}} \sum_{\mathbf{0}\le\mathbf{j}\le\max\{\mathbf{0},2^{\mathbf{l}-\mathbf{1}}-\mathbf{1}\}} u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}} \mathbf{v}_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}(x), \tag{2.45}$$

where

$$u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}} = \int_\Omega u(\mathbf{x}) \mathbf{v}_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}(\mathbf{x}) d\mathbf{x}, \tag{2.46}$$

with spatial domain $\Omega = [0,1]^d$ and dimension d. Here multi-index of constant is simple duplicate of the constant in d times, where d is dimension of spatial domain, e.g.

$$\mathbf{1} = (1,\ldots,1), \quad \mathbf{N} = (N,\ldots,N),$$

and $\max$ function here is component-wise. It is also demonstrated in [4, 5] that

$$\left(\sum_{\mathbf{0}\le\mathbf{i}\le\mathbf{k}} \sum_{\mathbf{0}\le\mathbf{j}\le\max\{\mathbf{0},2^{\mathbf{l}-\mathbf{1}}-\mathbf{1}\}} |u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}|^2 \right)^{1/2} \le C 2^{-(q+1)\mathbf{l}} |u|_{\mathcal{H}^{q+1}(\Omega)}, \quad \forall u \in \mathcal{H}^{p+1}(\Omega),$$

similar to the one dimension case. So in higher dimension, an element $V_{\mathbf{l}}^{\mathbf{j}}$ will be flagged as important if

$$\left(\sum_{\mathbf{0}\le\mathbf{i}\le\mathbf{k}} |u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}|^2 \right)^{1/2} \ge \epsilon. \tag{2.47}$$

The adaptive scheme is implemented by hash table as the underlying data structure, which stores active elements of the adaptive scheme, and the numerical solution on these active elements. The implementation details will be discussed in next section. We now introduce the concepts of child, parent and leaf elements [6], for better understanding of the adaptive procedure. Assume $\mathbf{V}_{\mathbf{l}}^{\mathbf{j}}$ and $\mathbf{V}_{\mathbf{l}'}^{\mathbf{j}'}$ are two elements, satisfying $|\mathbf{l}|_\infty, |\mathbf{l}'|_\infty \le N$. Then element $\mathbf{V}_{\mathbf{l}}^{\mathbf{j}}$ ais called child element of $\mathbf{V}_{\mathbf{l}}^{\mathbf{j}}$, if and only if for some $1 \le m \le d$, $\mathbf{l}'_m = \mathbf{l}_m + 1$, where $\mathbf{l}_m$ and $\mathbf{l}'_m$ are m-th component of multi-index $\mathbf{l}$ and $\mathbf{l}'$, respectively. Accordingly, we call $\mathbf{V}_{\mathbf{l}}^{\mathbf{j}}$ a father element of $\mathbf{V}_{\mathbf{l}'}^{\mathbf{j}'}$. If an element does not have any of its child elements in the hash table, then we call it a leaf element.

To describe the time evolution of adaptive scheme [6], consider now the numerical solution u_h on active elements at time step t^n is stored in a hash table H. Information of

leaf elements is stored in a separate leaf table. The time evolution from time step t^n to $t^{n+1} = t^n + \Delta t$ consists of four steps. The first step is the prediction step, which predicts the location on spatial domain Ω where the details of numerical solution u_h become significant at the next time step t^{n+1}, to determine in next step where we should add more elements in order to capture the fine structures. We solve from t^n to t^{n+1} using a cheap solver, e.g. the forward Euler method in time, to obtain predicted solution $u_h^{(p)}$. For spatial discretization in this step, if any nonlinear terms involve, the interpolation operator $\mathcal{I}$ applied here has the same multiresolution structure as determined by the hash table H corresponding to the numerical solution u_h at current time step t^n.

The second step is refinement step based on predicted solution $u_h^{(p)}$. An active element in hash table, satisfying (2.47), is considered as important element. All child elements $\mathbf{V}_{\mathbf{l}}^{\mathbf{j}}$ of an important element will be added to the hash table, to obtain new hash table $H^{(p)}$, and coefficients $u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}$ for $\mathbf{0} \leq \mathbf{i} \leq \mathbf{k}$ associated to newly added multiwavelet basis functions, as in (2.46), are set to be zeros. Leaf table is also updated accordingly. Now in the third step, with numerical solution u_h at time step t^n, and a new hash table $H^{(p)}$, we can solve numerical solution $\tilde{u}_h^{n+1}$ for the next step t^{n+1} before the final coarsening step. Again, the interpolation operator is now determined by the new hash table $H^{(p)}$.

The last step is coarsening step, to remove leaf elements that are not important. We traverse elements in the leaf table, and if a leaf element does not satisfy (2.47), it will be removed from both the leaf table and hash table. Note that the leaf table has to be dynamically updated in this procedure, since the removal of any leaf element may potentially create one or more new leaf elements. For the same reason, coarsening is done in recursive way, until no more leaf elements are removed and created. Now we obtain the new hash table and leaf table, and numerical solution u_h^{n+1} at time step $n+1$.

The last component of the algorithm is an efficient implementation of the hash table. As suggested in [124, 75] and further discussed in [4], the hash table approach is easy to implement, requires little storage overhead, and allows one to conveniently deal with hi-

erarchical multi-index $(\mathbf{l}, \mathbf{j})$ in the implementation. Specifically, by a prescribed hash function that will be discussed in details in 2.4.2, the hierarchical multi-index $(\mathbf{l}, \mathbf{j})$ is mapped to a hash key (an integer), which serves as an address in the hash table. Then, given a hierarchical index, the associated data can be easily stored and retrieved by computing the hash key.

2.4 Adaptive SGDG Package and PETSc Implementation

2.4.1 Overview of Package

Before we take into account full details of the package, we briefly introduce the structure of the SGDG package [8] here, which is developed using C++ language. The SGDG package is developed under the same logic as development of SGDG methods and their adaptive version, together with necessary time integration methods, to solve several different PDEs. Therefore in general, the SGDG pacakge consists significantly of two parts, regarding spatial and time discretization, together with other supporting classes and files.

The most important part is SGDG spatial discretization, including Alpert's multiwavelet basis, interpolatory multiwavelet basis, formulation of bilinear form, representation of numerical solutions as linear combination of basis functions, and fast algorithms for matrix multiplication. Several other features are implemented to support these purposes, including hash table to refer to basis functions, quadrature functions, and transformation of numerical solution under different multiwavelet basis, etc. Besides, several time integration methods are implemented to corporate with spatial SGDG discretization, including several Runge Kutta method or SSP methods [92, 108, 109, 110], and IMEX method [125, 6], with help of data structures and linear solvers in Eigen package [113]; after re-implementing several key functions regarding PETSc [9] data structures, generally fully implicit method with nonlinear terms becomes available in the package. At the end, several examples regarding different equations are written for further research and users in future; several unit tests are developed to verify correctness of certain significant and

complicated C++ classes and functions; makefile is developed to handle compiling and linking procedure to create executable effectively; doxygen [126] is introduced to generate documentation for source codes of the package.

2.4.2 Hash Key of SGDG Elements

For implementation of SGDG method, hash key is a key ingredient. After spatial discretization, each element or cell in SGDG formulation is provided a unique integer as the hash key, and this element could be identified in the later calculations by the hash key.

To describe the details of hash key, we first notice that each element is uniquely identified by its levels and support indexes on each dimension. Consider the SGDG formulation of d dimension and an element $\mathbf{V}_{\mathbf{l}}^{\mathbf{j}}$, then on dimension i, the level of the element is l_i and the support is identified as j_i, $1 \leq i \leq d$, which are i-th component of multi-index $\mathbf{l}$ and $\mathbf{j}$, respectively. Assume there are two elements $\mathbf{V}_{\mathbf{l}}^{\mathbf{j}}$ and $\mathbf{V}_{\mathbf{l}'}^{\mathbf{j}'}$, then element $\mathbf{V}_{\mathbf{l}}^{\mathbf{j}}$ has smaller hash key than $\mathbf{V}_{\mathbf{l}'}^{\mathbf{j}'}$, if and only if

1. $|\mathbf{l}|_1 < |\mathbf{l}'|_1$, i.e.
$$\sum_{i=1}^{d} l_i < \sum_{i=1}^{d} l'_i;$$
or

2. $|\mathbf{l}|_1 = |\mathbf{l}'|_1$, and $l_i = l'_i$ for $i = p+1, p+2, \ldots, d$, but $l_p < l'_p$ for some $1 \leq p \leq d$; or

3. $l_i = l'_i$ for all $1 \leq i \leq d$, and $j_i < j'_i$ for $i = 1, 2, \ldots, p-1$, but $j_p < j'_p$ for some $1 \leq p \leq d$.

It is clear that under such procedure, two elements, whose level vectors have the same l_1 norm, or who have the same level vectors, will have relatively close hash key numbers; elements with smaller l_1 norm of level vector will have smaller hash key numbers. These requirements are suitable for SGDG scheme, since sparse grid approximation space (2.43) collects elements with l_1 norm of level vector bounded by the maximum mesh level N.

2.4.3 SGDG Spatial Discretization

A DGSolution class is defined, in general, to represent the SGDG solutions as linear combination of (Alpert's or interpolatory) multiwavelet basis functions. The class stores several important information, including maximum mesh level $N =$ NMAX, dimension $d =$ DIM, number of components of solutions VEC_NUM, and most importantly, the hash table implemented in a C++ standard container map dg. Certain member functions are implemented, which includes initialization by L^2 projection of initial functions, copy function to copy SGDG solutions among realizations, evaluation functions to evaluate SGDG solutions at different spatial points regarding different basis functions, functions to compute errors, treatments regarding artificial viscosity, etc.

Active elements of SGDG method, together with the numerical solution, as discussed in last section, is stored in hash table, implemented by a standard C++ container map called dg. The key values of the map dg are hash key discussed in 2.4.2; the hash key is mapped to the object of class Element, each of which represents element $\mathbf{V}_\mathbf{l}^\mathbf{j}$, and numerical solution on the Alpert's multiwavelet basis $\{\mathbf{v}_{\mathbf{i},\mathbf{l}}^\mathbf{j}\}_{\mathbf{0}\le\mathbf{i}\le\mathbf{k}}$. On each element or object of Element class, the associated coefficients of numerical solution for Alpert's basis functions are stored in ucoe_alpt, while the associated coefficients for other interpolatory multiwavelet basis are stored in ucoe_intp. Other necessary data, including fucoe_intp for coefficients of $f(u)$ for interpolatory multiwavelet basis with given nonlinear function f and numerical solution u, are also declared and stored in Element class. Some basic information of the element, including the support and discontinuous points of the basis functions, are also stored explicitly for further use in volume integral etc. Besides, to compute the volume integral and cell boundary integral more efficiently, we need to determine whether two basis functions (or two elements) are orthogonal to each other; several functions in the class play such role. For the purpose of adaptive method, we need to add or remove elements dynamically twice on each time step, so we need to store the pointer to children element in hash_ptr_chd and to parent element in hash_ptr_par.

```
class DGSolution
{
public:
    // constructor and destructor of the class

    static int DIM;          // dimension of spatial domain
    static int VEC_NUM;      // number of unknowns

    ...

    // find pointers to elements that are related to current element
    void find_ptr_vol_alpt();
    void find_ptr_flx_alpt();
    void find_ptr_vol_intp();
    void find_ptr_flx_intp();

    // functions regarding error calculations and copying data between vectors

    // key of the map is hash key
    // basis functions and associated data store in object of class Element
    std::unordered_map<int, Element> dg;

    // calculate value of DG solution at a given point
    std::vector<double> val(const std::vector<double> & x, \
                            const std::vector<int> & derivative) const;

    // functions for initialization of DG solution

protected:

    const bool sparse;
    const int NMAX;          /// maximum mesh level

    Hash hash;
    AllBasis<AlptBasis> all_bas;
    AllBasis<LagrBasis> all_bas_Lag;
    AllBasis<HermBasis> all_bas_Her;

    ...
};
```

Algorithm 2.1: DGSolution Class, in DGSolution.h

There are also several classes defined for basis functions, all of which are derived from the base class Basis. Common features of basis functions are defined here, including level of mesh, support of basis and discontinuous points, index indicating which multiwavelet function the current basis function is scaled from, and several functions to compute volume or edge integrals. The derived classes, including AlptBasis, LagrBasis, HermBasis, represent the corresponding Alpert's multiwavelet basis functions and Lagrangian or Hermite interpolatory multiwavelet basis functions. Expression of basis functions and their derivatives are implemented for further usage in volume and edge integral calcula-

```
class Element
{
public:
    Element(const std::vector<int> & level_, const std::vector<int> & suppt_, \
                                        AllBasis<AlptBasis> & all_bas, Hash & hash);
    ~Element() {};

    static int DIM;          ///< dimension of spatial domain
    static int VEC_NUM;      ///< number of unknowns
    static std::vector<std::vector<bool>> is_intp;
                             ///< specify which components need interpolation

    const std::vector<int> level;       ///< mesh level in each dimension
    const std::vector<int> suppt;       ///< support index in each dimension

    /// left and right end point of support in multi-dimension
    std::vector<double> xl;
    std::vector<double> xr;

    std::vector<std::vector<double>> dis_point;     ///< discontinuous points
    std::vector<std::vector<double>> supp_interv;   ///< interval of support

    std::vector<int> order_elem;        ///< order for this element in each dimension
    int hash_key;                       ///< hash key for this element

    static int PMAX_alpt;       //< polynomial degree for Alpert's basis
    static int PMAX_intp;       //< polynomial degree for interpolatory basis

    // order of Alpert's or interpolatory multiwavelet basis in this element, e.g
    std::vector<VecMultiD<int>> order_alpt_basis_in_dg;
    ...

    // coefficients of Alpert's basis, and in fast multiplication e.g.
    std::vector<VecMultiD<double>> ucoe_alpt;
    ...

    // coefficients of interpolatory basis for numerical solution u, e.g.
    std::vector< VecMultiD<double> > ucoe_intp;
    ...

    // coefficients for interpolatory basis of f(u), e.g.
    std::vector< std::vector< VecMultiD<double> > > fucoe_intp;
    ...

    // coefficients of numerical solution on next time step, for fully implicit method
    std::vector< VecMultiD<double> > up_intp_next;
    std::vector< VecMultiD<double> > ucoe_alpt_next;

    // number of total children and parent elements; used in adaptive scheme
    int num_total_chd, num_total_par;

    // maps that store existing children and parents elements; used in adaptive scheme
    std::unordered_map<int,Element*> hash_ptr_chd;
    std::unordered_map<int,Element*> hash_ptr_par;

    ...
};
```

Algorithm 2.2: Element Class, in Element.h

tions.

Previous paragraphs discuss the top-down structure from DGSolution class to basis functions, and these classes are core classes for regular SGDG methods. However, to implement adaptive methods properly, a derived DGAdapt class from DGSolution class is necessary. The main feature added to DGAdapt is functions refine() and coarsen(), which conduct the refinement and coarseness procedure in adaptive method. The basic logic of these two algorithms are to search among elements stored in dg map of DGSolution class, or through leaf table called leaf and leaf_zero_child that store all leaf elements or leaf elements with no active child elements, to determine if an element is important, as discussed in details in Subsection 2.3.3. Note that an element's importance and whether an element should be removed are determined by the element and its active children, so the determination process is local.

The data structures defined above provide a clear picture of how solution is represented in the SGDG algorithm, but we still need certain data structures or classes of matrix to represent related transformations when time integration is applied. Recall that in Basis class, we define a general class for all basis, together with operations on basis functions, especially certain inner products between different basis functions or their derivatives. We then define, accordingly, a class template AllBasis<T>, which collects all basis functions in one dimension, where T represents one of AlptBasis, LagrBasis, HermBasis, specifying which type of basis is collected in the AllBasis<T>; we employ C++ template here to provide a generic class among all kinds of basis functions.

On top of the AllBasis class, we define class OperatorMatrix1D, which provides one dimensional transformation matrix from coefficients of numerical solution. Using the information provided in OperatorMatrix1D, we are able to assemble and store matrixes regarding SGDG spatial discretization in class BilinearForm. BilinearForm supports two sets of data structure: for thread parallelization we could use Eigen package [113] and its matrixes and vectors; for implicit scheme using nonlinear solver in time stepping, we use

```
class DGAdapt :
    public DGSolution
{
public:

    // constructor and destructor of class DGAdapt

    // refine based on reference solution generated by Euler forward
    void refine();

    // coarsen
    void coarsen();

    ...

protected:

    const double eps;      // refinement threshold
    const double eta;      // coarsening threshold

    // leaf element that does not have its all children
    std::unordered_map<int, Element*> leaf;
    // leaf element with no child
    std::unordered_map<int, Element*> leaf_zero_child;

    // after adding or deleting, check no holes in solution
    void check_hole();

    // update leaf table DGAdapt::leaf
    void update_leaf();

    // add a given element into DG solution
    void add_elem(Element & elem);

    // delete a given element into DG solution
    void del_elem(Element & elem);

    ...

private:

    // coarsen based on leaf elements with no child
    void coarsen_no_leaf();

    // update leaf table DGAdapt::leaf_zero_child
    void update_leaf_zero_child();

    ...
};
```

Algorithm 2.3: DGAdapt class, in DGAdapt.h

PETSc package [9] and its data structure and solvers.

2.4.4 Time Integration and PETSc Package

The package uses ODESolver class to deal with time integration, coupling with data structure from Eigen package. ODESolver class stores matrix and solution vectors together at the same time, and provide several functions to operate matrix and vector operations. Several specific time integration methods are defined as derived class of ODESolver, including explicit Strong Stability Preserving (SSP) methods, multistep methods, and IMEX method. Implementations of some of these methods require solving a linear system, in which case linear solver of Eigen package will be used to solve linear system when evolving the solution to next time step. This kind of approach made certain implicit method available in the package, but a fully implicit method involving nonlinearity would not fit into such framework. To deal with this, we introduce PETSc package [9] for nonlinear solvers involved.

In order to illustrate how SGDG package couples with PETSc package, we provide the following linear example. Consider linear PDE

$$U_t - (AU)_x - (BU)_y = 0,\ (t,x,y) \in [0,T] \times [0,1] \times [0,1]$$

with periodic boundary condition on both x and y direction, where $U(t,x,y) = (U_0, U_1, U_2)^T$ is vector-valued with three components, and

$$A = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

An accurate solution of this system is

$$U(t,x,y) = e^{\cos(2\pi(\sqrt{5}t+x+2y))}(\sqrt{5}, 2, -1).$$

The semi-discrete SGDG scheme of this equation is, to find vector-valued solution $U =$

$U(t,x,y)$, for any vector-valued test function V, such that

$$(U_t,V)+(AU,V_x)+\sum_{j=1}^{2^N}\int_0^1\left[C_{01}U(x^+,y)+C_{00}U(x^-,y)\right]\cdot[V]\Big|_{x=x_{j+1/2}}dy$$

$$+\quad(BU,V_y)+\sum_{j=1}^{2^N}\int_0^1\left[C_{11}U(x,y^+)+C_{10}U(x,y^-)\right]\cdot[V]\Big|_{y=y_{j+1/2}}dx=0 \tag{2.49}$$

where $(\cdot,\cdot)$ is inner product on spatial region $[0,1]^2$ of (x,y). For alternating flux

$$\widehat{U_0}=U_0^-,\quad \widehat{U_1}=U_1^+,\quad \widehat{U_2}=U_2^+,$$

we have

$$C_{00}=\begin{bmatrix}0&0&0\\0&0&0\\-1&0&0\end{bmatrix},\ C_{01}=\begin{bmatrix}0&0&-1\\0&0&0\\0&0&0\end{bmatrix},\ C_{10}=\begin{bmatrix}0&0&0\\1&0&0\\0&0&0\end{bmatrix},\ C_{11}=\begin{bmatrix}0&1&0\\0&0&0\\0&0&0\end{bmatrix}.$$

We can rewrite (2.49) as $(U_t,V)+B(U,V)=0$ where $B(\cdot,\cdot)$ is a bilinear form. For fully implicit method in time, consider numerical solutions at time step t^n and $t^{n+1}=t^n+\Delta t$ are U^n and U^{n+1} respectively, then

$$F(U^{n+1};U^n,V):=(U^{n+1}-U^n,V)+\frac{\Delta t}{2}B(U^{n+1}+U^n,V)=0. \tag{2.50}$$

In Algorithm 2.4, we provide an example of solving (2.48), after removing some technical details, to reveal the main structure of the SGDG package. Maximum mesh level NMAX and polynomial degree PMAX of Alpert's multiwavelet basis are provided by command line arguments. Static variables, hash key in class Hash, collection of all Alpert's multiwavelet basis functions in class AllBasis<AlptBasis>, one dimension matrix for linear operations in class OperatorMarix1D<AlptBasis,AlptBasis>, and numerical solution and hash table in class DGSolution, are defined in proper order. Functions with name format assemble_matrix_*_system assemble the matrix corresponding to bilinear form $B(\cdot,\cdot)$, using coefficients matrix A,B, and $C_{00},C_{01},C_{10},C_{11}$, and store the assembled matrix in PETSc matrix mat_petsc of class HyperbolicAlpt, a derived class from class BilinearForm.

```
// include header files

typedef struct AppCtx AppCtx;
struct AppCtx
{
    HyperbolicAlpt* app_HyperbolicAlpt;
    DGSolution*     app_DGSolution;
    double          dt;
};

int main(int argc, char** argv)
{
    PetscErrorCode ierr;
    ierr = PetscInitialize(&argc,&argv,(char*)0,NULL);if (ierr) return ierr;

    int NMAX=6,PMAX=1;
    ierr=PetscOptionsGetInt(NULL,NULL,"-NMAX",&NMAX,NULL);CHKERRQ(ierr);
    ierr=PetscOptionsGetInt(NULL,NULL,"-PMAX",&PMAX,NULL);CHKERRQ(ierr);

    // assign values to static variables
    AlptBasis::PMAX = PMAX;
    Element::DIM = 2;
    Element::VEC_NUM = 3;
    ...

    Hash hash; // hash key
    AllBasis<AlptBasis> all_bas_alpt(NMAX);
    const std::string boundary_type = "period";
    OperatorMatrix1D<AlptBasis, AlptBasis> \
                        oper_matx_alpt(all_bas_alpt, all_bas_alpt, boundary_type);

    DGSolution dg_solu(true, NMAX, NMAX, all_bas_alpt, hash);

    // initialize the DGSolution with an initial function

    HyperbolicAlpt linear(dg_solu, oper_matx_alpt);
    linear.assemble_matrix_vol_system(0,A);
    linear.assemble_matrix_vol_system(1,B);
    linear.assemble_matrix_flx_system(0,1,C01);
    linear.assemble_matrix_flx_system(0,-1,C00);
    linear.assemble_matrix_flx_system(1,1,C11);
    linear.assemble_matrix_flx_system(1,-1,C10);

    double final_time = 1./std::sqrt(5.);
    double cfl = 0.9;

    AppCtx app_sol;
    app_sol.app_HyperbolicAlpt = &linear;
    app_sol.app_DGSolution     = &dg_solu;

    double current_time = 0.;

    // while loop to find numerical solution; see below

    // calculate errors; print solutions

    // Destroy PETSc objects

    return ierr;
}
```

Algorithm 2.4: Example of Linear Equations, in Linear2D.cpp

In the while loop of finding numerical solutions of each time step, as shown in Algorithm 2.5, a Scalable Nonlinear Equations Solvers (SNES) object of PETSc is created for solving general linear and nonlinear system. For linear problem, scalable linear equations solvers (KSP) of PETSc is a better choice, and setting of KSP object is similar to SNES object.

```
while (current_time < final_time)
{
    double dt = cfl / std::pow(2., NMAX);
    dt = std::min( dt , finaltime - curr_time );
    app_sol.dt = dt;

    // Initialize vectors for residual and next step solution
    Vec residual, next_step_solution;
    ...

    SNES snes;
    ierr = SNESCreate(PETSC_COMM_WORLD,&snes);
    ierr = SNESSetFunction(snes,residual,FormFunction,&app_sol);
    ierr = SNESSetFromOptions(snes);

    ierr = SNESSolve(snes,NULL,next_step_solution);

    // Copy next step solution back to DGSolution class

    curr_time += dt;

    // Destroy PETSc objects
}
```

Algorithm 2.5: While Loop to Find Numerical Solution in Algorithm 2.4

As in while loop of Algorithm 2.5, the key functions of SNES nonlinear solver are SNESSetFunction and SNESSolve. SNESSetFunction provides to the nonlinear solver a FormFunction, which compute $F(x)$ if the nonlinear problem to be solved is $F(x) = 0$. In our example, as in Algorithm 2.6, FormFunction computes $F(U^{n+1}; U^n, V)$ with input of numerical solution of next time step U^{n+1}, and numerical solution of current time step U^n and other related information are provided through an application context of class AppCtx. As in Algorithm 2.4, an AppCtx object stores time step dt$= \Delta t$, and the pointer of DGSolution object of SGDG solution, and HyperbolicAlpt object of bilinear form of the system.

Nevertheless, SNESSolve uses iterative method to find solution of nonlinear prob-

```
PetscErrorCode FormFunction(SNES snes,Vec next_u,Vec f,void* ctx)
{
    AppCtx *user = (AppCtx*)ctx;
    PetscErrorCode ierr;

    Vec now_u;
    // Copy current step solution in user->app_DGSolution to now_u
    ...

    Vec sum_, diff_;
    // sum_  = 0.5 * (next_u + now_u) * user->dt
    ...
    // diff_ = next_u - now_u
    ...

    // f = diff_ + mat_petsc * sum_
    ierr = MatMultAdd(user->app_HyperbolicAlpt->mat_petsc, sum_, diff_, f);

    return ierr;
}
```

Algorithm 2.6: Function FormFunction for Nonlinear Solver in Algorithm 2.4

lem, and SNESSetFromOptions sets various SNES parameters, either from command line arguments or by default. A typical set of options provided through command line arguments is like

```
Linear2D -NMAX 6 -PMAX 3 -snes_mf \
        -snes_rtol 1e-12 -snes_atol 1e-12 -snes_stol 1e-12
```

Algorithm 2.7: Command to Run Executable Linear2D

which set maximum mesh level 6 and polynomial degree 3, choose matrix-free method through option -snes_mf, and set tolerance of nonlinear solvers by options –snes_rtol, –snes_atol, and –snes_stol.

Before the end of this section, we remark that although the example provided here is linear, the nonlinear problem can fit into this structure too. In case a FormFunction is provided, matrix-free method can be applied to solve the numerical solution.

2.5 Numerical Simulation in One Dimensional Case

2.5.1 Kink Shape Solutions

The first numerical test we consider, as discussed in [1], where a traveling wave solution was constructed for the instantaneous intensity-dependent Kerr response neglecting the influence of damping, corresponding to the case of $\theta = 0$ and $\tau = \infty$ in (2.11). This yields a simplified system from Equations (2.11) as the following:

$$\partial_t H = \partial_x E \tag{2.51a}$$

$$\partial_t D = \partial_x H \tag{2.51b}$$

$$\partial_t P = 6J \tag{2.51c}$$

$$\partial_t J = 6(-\omega_0^2 P + \omega_p^2 E) \tag{2.51d}$$

$$D = \epsilon_\infty E + P + aE^3 \tag{2.51e}$$

where the only nonlinear term comes from constitutive law of D, representing the cubic Kerr effect. We can find a traveling wave function $E(x,t) = E(\xi)$ with $\xi = 6(x - vt)$, where $E(\xi)$ is comprised of a kink and anti-kink wave, and is solved based on the following ODE of $E(\xi)$ and $\Phi(\xi)$:

$$\frac{dE}{d\xi} = \Phi \tag{2.52a}$$

$$\frac{d\Phi}{d\xi} = \frac{6av^2 E\Phi^2 + (\epsilon_\infty \omega_0^2 + \omega_p^2 - \omega_0^2/v^2)E + a\omega_0^2 E^3}{1 - \epsilon_\infty v^2 - 3av^2 E^2} \tag{2.52b}$$

Here the parameters are

$$\epsilon_\infty = 2.25,\ a = 0.75,\ \omega_0 = 93.627179982222216,\ \omega_p = \sqrt{3}\omega_0,\ v = \frac{0.6545}{\sqrt{\epsilon_\infty}},$$

with condition

$$E(0) = 0, \quad \Phi(0) = 0.24919666777865812.$$

These conditions are carefully chosen, so that the resulting solution of E, Φ are 6-periodic. The overall solution for all variables, based on E, Φ obtained from the ODE, again with

$\xi = 6(x - vt)$, is the following:

$$H(x,t) = -\frac{1}{v}E(\xi), \quad D(x,t) = \frac{1}{v^2}E(\xi) \tag{2.53}$$

$$P(x,t) = (\frac{1}{v^2} - \epsilon_\infty)E(\xi) - aE(\xi)^3 \tag{2.54}$$

$$J(x,t) = (\epsilon_\infty v - \frac{1}{v})\Phi(\xi) + 3avE(\xi)^2\Phi(\xi) \tag{2.55}$$

Note that all H, D, E, P, J have period 1. The function E and J at $t = 0$ are given in Figure 2.1.

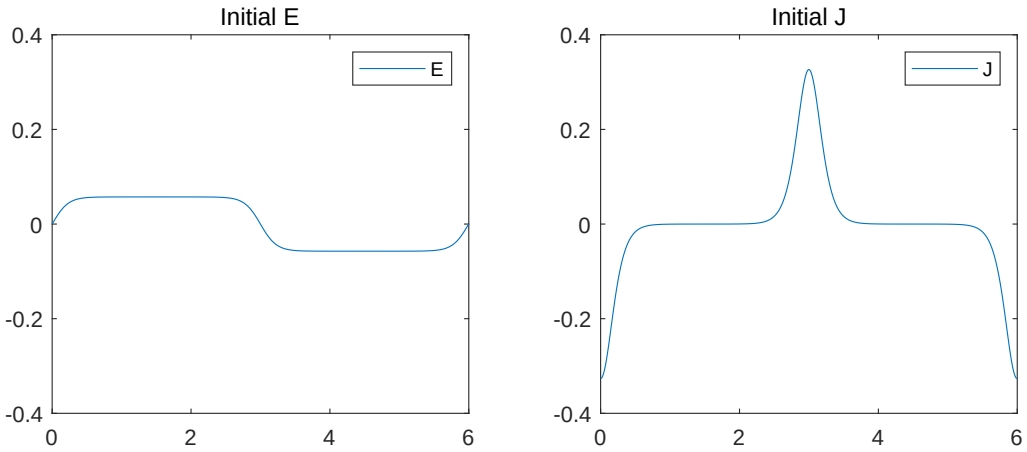


Figure 2.1: Function E and J at $t = 0$

We perform an accuracy test for fully implicit method. We choose the CFL condition to be

$$\Delta t = CFL \times h^{(k+1)/2}$$

where $h = 1/2^N$ is mesh size, and k is degree of piecewise polynomial. CFL constant is set as in Table 2.1. L^1, L^2 and L^∞ errors with order, regarding function E and J, are shown,

k	1	2	3
CFL	1	8	63

Table 2.1: CFL constant for fully implicit method

respectively, in Table 2.2 and 2.3 for $k = 1$, in Table 2.4 and 2.5 for $k = 2$, and in Table 2.6 and 2.7 for $k = 3$. All errors are calculated at final time $T = 1/v$. It is clear that SGDG

scheme with P^k Alpert's multiwavelet basis has accuracy order $k+1$. This is expected since SGDG is the same as traditional DG in one dimension.

N	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
6	7.915e-04		1.506e-03		5.295e-03	
7	2.217e-04	1.836	4.421e-04	1.769	1.708e-03	1.632
8	5.606e-05	1.983	1.143e-04	1.951	4.638e-04	1.881
9	1.399e-05	2.002	2.865e-05	1.996	1.180e-04	1.975
10	3.494e-06	2.001	7.160e-06	2.001	2.948e-05	2.001

Table 2.2: Errors and Orders of Accuracy of E. $k=1, T=1/v$.

N	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
6	8.435e-03		1.623e-02		5.124e-02	
7	2.604e-03	1.696	5.537e-03	1.551	1.982e-02	1.370
8	6.826e-04	1.931	1.514e-03	1.871	5.793e-03	1.775
9	1.717e-04	1.991	3.836e-04	1.980	1.486e-03	1.963
10	4.292e-05	2.000	9.603e-05	1.998	3.729e-04	1.994

Table 2.3: Errors and Orders of Accuracy of J. $k=1, T=1/v$.

N	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
6	8.902e-04		1.718e-03		5.946e-03	
7	1.161e-04	2.939	2.359e-04	2.864	9.532e-04	2.641
8	1.452e-05	3.000	2.967e-05	2.991	1.226e-04	2.959
9	1.816e-06	2.999	3.715e-06	2.998	1.541e-05	2.992
10	2.270e-07	3.000	4.643e-07	3.000	1.926e-06	3.000

Table 2.4: Errors and Orders of Accuracy of E. $k=2, T=1/v$.

N	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
6	1.078e-02		2.051e-02		6.959e-02	
7	1.426e-03	2.919	3.144e-03	2.706	1.200e-02	2.536
8	1.787e-04	2.996	3.992e-04	2.977	1.487e-03	3.013
9	2.236e-05	2.998	5.002e-05	2.997	1.844e-04	3.011
10	2.796e-06	3.000	6.253e-06	3.000	2.305e-05	3.000

Table 2.5: Errors and Orders of Accuracy of J. $k=2, T=1/v$.

N	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
6	8.642e-04		1.665e-03		5.768e-03	
7	5.565e-05	3.957	1.134e-04	3.877	4.617e-04	3.643
8	3.473e-06	4.002	7.104e-06	3.996	2.926e-05	3.980
9	2.171e-07	4.000	4.440e-07	4.000	1.830e-06	3.999
10	1.357e-08	4.000	2.775e-08	4.000	1.144e-07	4.000

Table 2.6: Errors and Orders of Accuracy of E. $k = 3, T = 1/v$.

N	L^1 Error	Order	L^2 Error	Order	L^∞ Error	Order
6	1.058e-02		2.006e-02		6.870e-02	
7	6.831e-04	3.953	1.521e-03	3.721	5.726e-03	3.585
8	4.279e-05	3.997	9.565e-05	3.991	3.510e-04	4.028
9	2.674e-06	4.000	5.980e-06	4.000	2.190e-05	4.003
10	1.688e-07	3.986	3.738e-07	4.000	1.371e-06	3.998

Table 2.7: Errors and Orders of Accuracy of J. $k = 3, T = 1/v$.

Besides, we also use this example to test our adaptive method, and we calculate the convergence rate with respect to the error threshold R_ε and with respect to degree of freedoms R_{DOF} as following

$$R_{DOF} = \frac{\log(e_{l-1}/e_l)}{\log(DOF_l/DOF_{l-1})}, \quad R_\varepsilon = \frac{\log(e_{l-1}/e_l)}{\log(\varepsilon_l/\varepsilon_{l-1})}.$$

where e_l is the standard L^2 error, of either E or J component, with refinement parameter ε_l, and DOF_l is the associated number of active degrees of freedom at final time after last coarsening step.

2.5.2 Soliton Propagation

In this example, we will consider the soliton propagation similar to the setup in [1]. The coefficients in this example (2.12) are chosen as

$$\epsilon_\infty = 2.25, \epsilon_s = 5.25, \beta_1 = 3, \frac{1}{\tau} = 1.168 \times 10^{-5}, \frac{1}{\tau_v} = 29.2/32,$$

$$a = 0.07, \theta = 0.3, \Omega_0 = 12.57, \omega_0 = 5.84, \omega_v = 1.28, \omega_p = \omega_0\sqrt{\beta_1}.$$

ε	DOF	E error	R_{DOF}	R_ε	J error	R_{DOF}	R_ε
5.00e-05	48	5.57e-05			3.46e-03		
2.00e-05	58	3.10e-05	3.09	0.64	2.02e-03	2.86	0.59
1.00e-05	74	1.69e-05	2.50	0.88	1.26e-03	1.92	0.67
5.00e-06	88	9.14e-06	3.54	0.89	7.03e-04	3.38	0.85
2.50e-06	92	5.42e-06	11.73	0.75	3.72e-04	14.31	0.92

Table 2.8: Adaptive Result of P^2 case, using alternating flux

ε	DOF	E error	R_{DOF}	R_ε	J error	R_{DOF}	R_ε
5.00e-05	36	7.84e-05			2.46e-03		
2.00e-05	40	3.42e-05	7.89	0.91	1.15e-03	7.23	0.83
1.00e-05	62	2.32e-05	0.88	0.56	8.14e-04	0.79	0.50
5.00e-06	76	8.76e-06	4.78	1.41	3.40e-04	4.28	1.26
2.50e-06	82	6.21e-06	4.54	0.50	1.79e-04	8.44	0.93

Table 2.9: Adaptive Result of P^2 case, using upwind flux

ε	DOF	E error	R_{DOF}	R_ε	J error	R_{DOF}	R_ε
5.00e-05	24	2.77e-05			1.71e-03		
2.00e-05	32	1.64e-05	1.82	0.57	9.12e-04	2.18	0.69
1.00e-05	40	1.05e-05	2.01	0.65	5.65e-04	2.15	0.69
5.00e-06	52	6.01e-06	2.12	0.80	3.82e-04	1.49	0.56
2.50e-06	58	3.98e-06	3.76	0.59	2.75e-04	3.01	0.47

Table 2.10: Adaptive Result of P^3 case, using alternating flux

ε	DOF	E error	R_{DOF}	R_ε	J error	R_{DOF}	R_ε
5.00e-05	22	3.24e-05			1.25e-03		
2.00e-05	26	1.76e-05	3.63	0.66	4.17e-04	6.55	1.19
1.00e-05	32	8.81e-06	3.35	1.00	2.33e-04	2.80	0.84
5.00e-06	44	6.58e-06	0.92	0.42	2.23e-04	0.14	0.06
2.50e-06	48	4.28e-06	4.94	0.62	1.32e-04	6.00	0.75

Table 2.11: Adaptive Result of P^3 case, using upwind flux

Initially, all fields are zero. The left boundary is injected with an incoming solitary wave, for which the electric field is prescribed as

$$E(x=0,t) = f(t)\cos(\Omega_0 t),$$

where $f(t) = M\,\mathrm{sech}(t-20)$. M is a constant to be specified later. As in [1], the boundary condition of H can be approximated from the linearized dispersion relation. Assuming a space-time harmonic variation $e^{i(wt-kx)}$ of all fields, the exact dispersion relation associ-

ated with the linear parts of the system implies that

$$k = \omega\sqrt{\epsilon_\infty}\sqrt{1 - \frac{\omega_p^2/\epsilon_\infty}{\omega^2 - i\omega/\tau - \omega_0^2}}.$$

The approximate value of H is given by

$$H(x=0,t) = \int_{-\infty}^{\infty} \widehat{H}(\omega) e^{i\omega t} d\omega \simeq \Re\left\{ \sum_{m=0}^{8} \frac{(-i)^m}{m!} \left(\frac{1}{Z}\right)^{(m)} \Big|_{\omega=\Omega_0} f^{(m)}(t) \exp(i\omega_0 t) \right\}.$$

where $\Re$ is real part of complex number, $f^{(m)}(t)$ is the m-th derivative of $f(t)$, and $(\frac{1}{Z})^{(m)}$ is the m-th derivative of $Z = -\omega/k$ with respect to ω, where k is considered as a function of ω as described above. Besides, we consider the absorbing right boundary derived from the linearized system, which is

$$\widehat{E} = \frac{3}{4}E^- - \frac{1}{4\sqrt{\epsilon_\infty}}H^-, \quad \widetilde{H} = \frac{3}{4}H^- - \frac{\sqrt{\epsilon_\infty}}{4}E^-,$$

for central flux and alternating flux, and

$$\widehat{E} = \frac{1}{2}E^- - \frac{1}{2\sqrt{\epsilon_\infty}}H^-, \quad \widetilde{H} = \frac{1}{2}H^- - \frac{\sqrt{\epsilon_\infty}}{2}E^-,$$

for upwind flux; these settings guarantee the energy stability for semi-discrete schemes as desired and discussed in [1].

We choose maximum mesh level to be $N = 11$, and the time step $\Delta t = CFL \times \Delta x = CFL/2^N$. The CFL number is chosen to be $CFL = 0.3$ for both alternating I/II fluxes and upwind flux, to eliminate the error from time stepping and to ensure the convergence to the expected solution. Figure A.1 and A.9 show the adaptive scheme solutions at two different time $T = 40$ or $T = 80$, for different fluxes and different degrees of Alpert's multiwavelet basis of SGDG spaces, while the previous figure is for transient fundamental temporal soliton, and the later for transient second order temporal soliton. We choose refinement error threshold $\varepsilon = 10^{-5}$, and coarsening error threshold $\delta = \varepsilon/10 = 10^{-6}$.

As discussed in [121, 122, 1], a daughter pulse occurs other than the soliton-like pulse, and travels faster than the soliton-like pulse. Among all cases, the daughter pulse has

smaller magnitude and higher frequency, comparing to the soliton-like pulse. The daughter pulse shows up in almost all simulations except in case of upwind flux with polynomial degree 1, since numerical dissipation of upwind flux is stronger than other flux, damping the magnitude of the pulse significantly.

We also plot at time $T = 40$ and $T = 80$, the numerical solutions and active elements of adaptive scheme, for different fluxes and polynomial degree, in Figure A.2, A.3, A.4 for transient fundamental case, and in Figure A.10, A.11, A.12 for transient second order case. Among all cases, the active elements of numerical solution sweep from left boundary to right boundary, following the position of the soliton-like pulse. For any fixed flux, high order method uses fewer active elements on simulation, when the high order and low order method share the same refinement and coarsening accuracy threshold. It is also clear that fewer active elements are in the hash table for adaptive scheme with upwind flux, especially at time $T = 80$ comparing to the other alternating fluxes; again this demonstrates upwind flux is numerically more dissipative than other fluxes, so that it significantly eliminates any oscillations.

To demonstrate the function of refinement threshold ε, we also compute the transient fundamental temporal soliton with refinement threshold $\varepsilon = 10^{-3}$ and coarsening threshold $\delta = 10^{-4}$, in Figure A.5, A.6, A.7, and A.8. Spurious oscillations occur on simulations of alternating flux on the left side of soliton-like pulse; besides, oscillation on alternating flux II case has larger magnitude than that of alternating I case. There is no oscillation on upwind flux case, again due to numerical dissipation of upwind flux. Simulations with upwind flux use fewer active elements than alternating fluxes, which is a similar result as in $\varepsilon = 10^{-5}$ case. Comparing to $\varepsilon = 10^{-5}$ case, when polynomial degree and flux are the same, simulations of $\varepsilon = 10^{-3}$ use fewer elements, especially on the right side of soliton-like pulse, and for this reason, daughter pulse in $\varepsilon = 10^{-5}$ case does not show up in the $\varepsilon = 10^{-3}$ case. Larger thresholds of refinement and coarsening allow larger error on numerical solution, so the numerical scheme becomes less sensitive on detecting the

daughter pulse of small magnitude. Similar to $\varepsilon = 10^{-5}$ case, support of elements sweep from the left to the right following transport of soliton-like pulse.

2.6 Numerical Simulation in Higher Dimensional Case

2.6.1 Accuracy and Convergence Test of Linear Equation

The following equation of unknown $u = u(t, x, y)$, with $(t, x, y) \in [0, T] \times \Omega$ and $\Omega = [0, 1] \times [0, 1]$, is considered in this test

$$u_t + u_x + u_y = 0, \tag{2.56}$$

with periodic boundary condition in both x and y direction. The initial condition is

$$u(0, x, y) = u_0(x, y) = \frac{1}{2 + \sin(2\pi(x - y))}, \tag{2.57}$$

and the accurate solution is

$$u(t, x, y) = u_0(x - t, y - t). \tag{2.58}$$

Spatially we use Sparse Grid DG method with upwind flux, and trapezoidal discretization in time. Since trapezoidal rule is second order in time, we use the following CFL condition:

$$\Delta t \leq \frac{CFL}{\frac{1}{\Delta x^{(k+1)/2}} + \frac{1}{\Delta y^{(k+1)/2}}} \tag{2.59}$$

if spatial discretization is $k + 1$ order. The CFL number is chosen as the following:

k	1	2	3
CFL	0.9	4.9	9.9

Table 2.12: CFL numbers

We obtain the following tables of numerical error and order, for end time $T = 1$. Here N represent the maximum mesh level on both x, y direction, i.e. $\Delta x = \Delta y = \frac{1}{2^N}$. We also put together the table at $T = 0$, i.e. the error of projection of initialization for comparison, and similar order of accuracy is observed.

N	L^1 error	order	L^2 error	order	L^∞ error	order
5	4.57312e-02		5.30394e-02		1.15945e-01	
6	1.51710e-02	1.592	1.98859e-02	1.415	5.50358e-02	1.075
7	5.14680e-03	1.560	7.43171e-03	1.420	2.41551e-02	1.188
8	1.49080e-03	1.788	2.31736e-03	1.681	9.06915e-03	1.413
9	2.73017e-04	2.449	4.83023e-04	2.262	2.87813e-03	1.656

Table 2.13: Errors and orders of accuracy of u, for $k = 1$, at $T = 1$

N	L^1 error	order	L^2 error	order	L^∞ error	order
5	7.89276e-03		1.24346e-02		5.85053e-02	
6	2.32947e-03	1.761	3.63022e-03	1.776	2.05560e-02	1.509
7	8.50218e-04	1.454	1.49729e-03	1.278	1.17706e-02	0.804
8	2.33762e-04	1.863	4.21631e-04	1.828	3.97204e-03	1.567
9	6.76643e-05	1.789	1.24259e-04	1.763	1.27171e-03	1.643

Table 2.14: Errors and orders of accuracy of u, for $k = 1$, at $T = 0$

N	L^1 error	order	L^2 error	order	L^∞ error	order
5	5.07673e-03		6.88257e-03		2.69062e-02	
6	4.97155e-04	3.352	7.57930e-04	3.183	5.93592e-03	2.180
7	1.81276e-04	1.456	2.87770e-04	1.397	2.56533e-03	1.210
8	1.89313e-05	3.259	3.28919e-05	3.129	4.44195e-04	2.530
9	2.79344e-06	2.761	5.14717e-06	2.676	7.21485e-05	2.622

Table 2.15: Errors and orders of accuracy of u, for $k = 2$, at $T = 1$

N	L^1 error	order	L^2 error	order	L^∞ error	order
5	1.26846e-03		2.26795e-03		1.55856e-02	
6	1.53309e-04	3.049	2.74032e-04	3.049	2.37428e-03	2.715
7	4.71975e-05	1.700	9.17983e-05	1.578	1.04093e-03	1.190
8	6.57529e-06	2.844	1.27438e-05	2.849	1.99676e-04	2.382
9	9.84804e-07	2.739	2.00984e-06	2.665	3.13352e-05	2.672

Table 2.16: Errors and orders of accuracy of u, for $k = 2$, at $T = 0$

N	L^1 error	order	L^2 error	order	L^∞ error	order
4	4.45318e-03		5.69848e-03		2.10659e-02	
5	1.11115e-03	2.003	1.46891e-03	1.956	7.11292e-03	1.566
6	1.10792e-04	3.326	1.47087e-04	3.320	8.31930e-04	3.096
7	8.05073e-06	3.783	1.70609e-05	3.108	1.51009e-04	2.462
8	1.20156e-06	2.744	1.80771e-06	3.238	1.11994e-05	3.753

Table 2.17: Errors and orders of accuracy of u, for $k = 3$, at $T = 1$

N	L^1 error	order	L^2 error	order	L^∞ error	order
4	1.04154e-03		1.48986e-03		4.86638e-03	
5	2.18472e-04	2.253	4.12664e-04	1.852	1.88407e-03	1.369
6	1.81491e-05	3.589	3.27455e-05	3.656	3.26242e-04	2.530
7	2.18507e-06	3.054	5.21414e-06	2.651	7.02091e-05	2.216
8	1.71983e-07	3.667	3.57593e-07	3.866	6.91854e-06	3.343

Table 2.18: Errors and orders of accuracy of u, for $k = 3$, at $T = 0$

For the same equation, initial condition, and accurate solution as described above, we also consider the adaptive scheme. In such case we use CFL condition

$$\Delta t \leq \frac{CFL}{\frac{1}{\Delta x} + \frac{1}{\Delta y}},$$

with CFL$= 0.9$. We simulate the case of maximum mesh level $N = 7$ or $N = 8$, at final time $T = 1$, and different refinement parameter ε; coarsen parameter η is set to be $\varepsilon/10$. We also compute convergence rate, R_ε with respect to refinement parameter or error threshold, and R_{DOF} with respect to active degree of freedom at final time, with following formula:

$$R_{DOF} = \frac{\log(e_{l-1}/e_l)}{\log(DOF_l/DOF_{l-1})} \tag{2.60}$$

$$R_\varepsilon = \frac{\log(e_{l-1}/e_l)}{\log(\varepsilon_{l-1}/\varepsilon_l)} \tag{2.61}$$

where e_l is L^2 error, ε_l the corresponding refinement parameter ε, and DOF_l the active degrees of freedom at final time step. These results are shown in Table 2.19.

In the following Figure 2.2, we plot the accurate solution at $T = 1$, i.e.

$$u(t = 1, x, y) = \frac{1}{2 + \sin(2\pi(x - y))},$$

and the distribution of active elements on computational domain, by showing the center of active elements, at final time $T = 1$, for different cases as given in Table 2.19, i.e $k = 1$ for $N = 8$ and $k = 2, 3$ for $N = 7$, with $\varepsilon = 10^{-5}$. The figures show that more active elements are needed in the region where the numerical solution has larger magnitude, which is expected for adaptive scheme.

	ε	DOF	L^2 error	R_{DOF}	R_ε
$k=1, N=8$	1e-03	1216	3.472e-03		
	5e-04	1848	2.671e-03	0.627	0.378
	1e-04	3232	9.350e-04	1.878	0.652
	5e-05	5456	3.508e-04	1.872	1.414
	1e-05	10976	1.628e-04	1.099	0.477
	5e-06	13980	1.286e-04	0.972	0.339
$k=2, N=7$	1e-03	900	1.367e-03		
	5e-04	1080	5.501e-04	4.993	1.313
	1e-04	2268	2.874e-04	0.875	0.403
	5e-05	2880	1.808e-04	1.941	0.669
	1e-05	5400	4.064e-05	2.374	0.927
	5e-06	6930	2.996e-05	1.222	0.440
$k=3, N=7$	1e-03	704	5.249e-04		
	5e-04	1120	4.092e-04	0.536	0.359
	1e-04	1856	8.728e-05	3.059	0.960
	5e-05	2112	4.854e-05	4.540	0.846
	1e-05	3968	1.858e-05	1.523	0.597
	5e-06	4800	6.058e-06	5.888	1.617

Table 2.19: L^2 error and order for adaptive scheme, for $T = 1$

2.6.2 Accuracy and Convergence Test for nonlinear Maxwell Equation

Here we consider the nondimensionlized form of system (2.31), i.e.

$$\partial_t H_z = \partial_y E_x - \partial_x E_y \tag{2.62a}$$

$$\partial_t D_x = \partial_y H_z \tag{2.62b}$$

$$\partial_t D_y = -\partial_x H_z \tag{2.62c}$$

$$\mathbf{D} = \epsilon_\infty \mathbf{E} + \mathbf{P} + a(1-\theta)|\mathbf{E}|^2\mathbf{E} + a\theta Q\mathbf{E} \tag{2.62d}$$

$$\partial_t \mathbf{P} = \mathbf{J} \tag{2.62e}$$

$$\partial_t \mathbf{J} = -\gamma \mathbf{J} - \omega_0^2 \mathbf{P} + \omega_p^2 \mathbf{E} \tag{2.62f}$$

$$\partial_t Q = \sigma \tag{2.62g}$$

$$\partial_t \sigma = -\gamma_v \sigma - \omega_v^2 Q + \omega_v^2 |\mathbf{E}|^2 \tag{2.62h}$$

on spatial domain $\Omega = [0,1]^2$ and time domain $[0,T]$, and manufactured solution with periodic boundary condition as specified below, to demonstrate accuracy and convergence

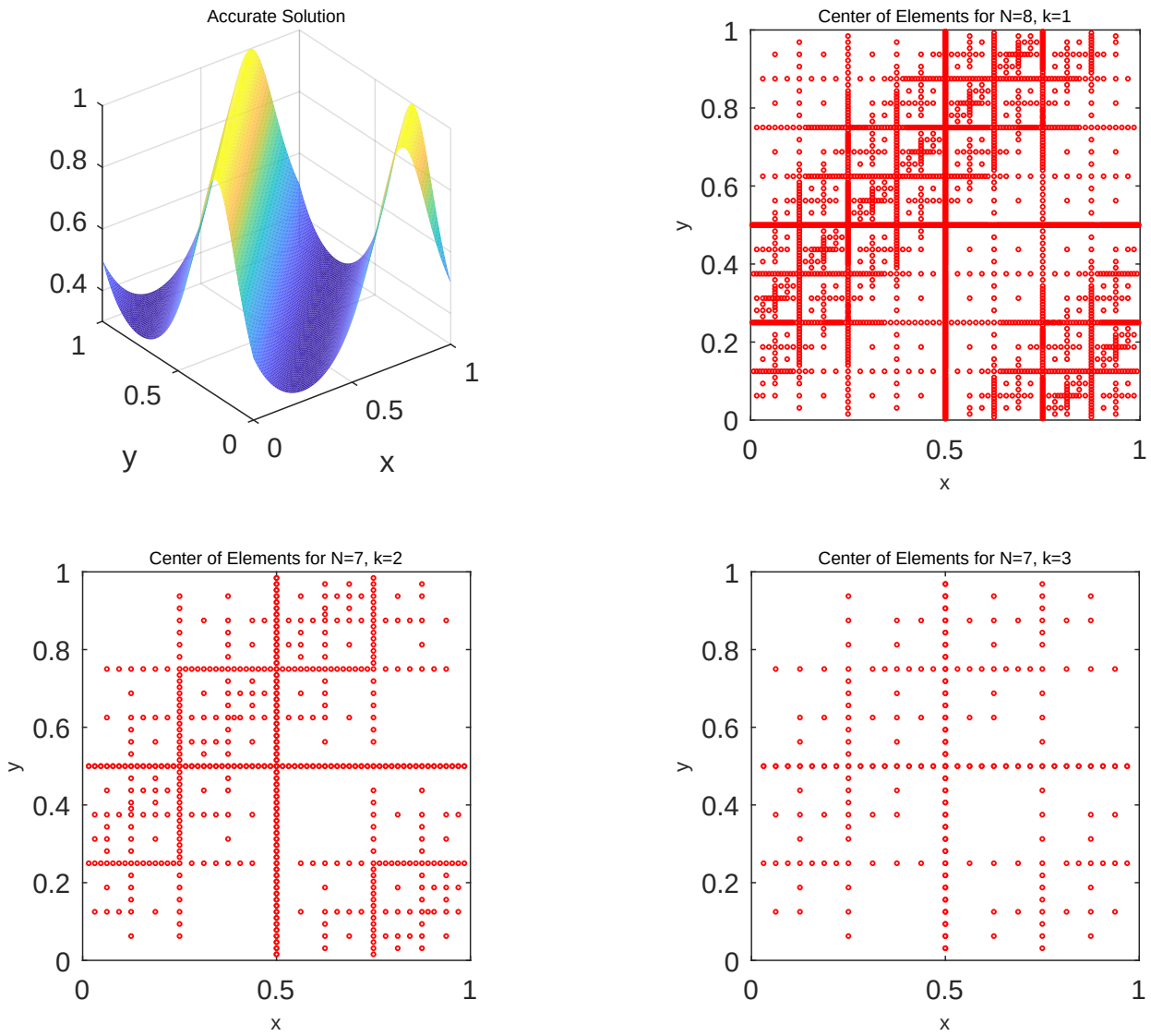


Figure 2.2: Accurate solution, and active elements at $T = 1$ for $\varepsilon = 10^{-5}$; red circles are center of active elements.

of SGDG method. We use following functions as manufactured solution:

$$H_z(t, x, y) = \exp(\cos(w(t + \alpha x + \beta y))/q), \tag{2.63}$$

and

$$\mathbf{P} = \mathbf{D} = \mathbf{E} = (\beta, -\alpha)H_z, \quad \mathbf{J} = \partial_t \mathbf{P}, \quad Q = H_z,\ \sigma = \partial_t H_z. \tag{2.64}$$

We also define parameters as

$$\epsilon_\infty = 1,\ a = 0.01,\ \theta = 0.125,\ \gamma_0 = \gamma_v = 0.05,\ \omega_0 = \omega_p = \omega_v = 1, \tag{2.65}$$

so equations (2.62d), (2.62f), and (2.62h) needs to add extra source for balance of equations. However, the PDE part of system (2.62) is satisfied, whenever $\alpha^2 + \beta^2 = 1$. In our example here, to impose periodic boundary condition, we choose

$$w = 2\pi\sqrt{5}, \quad \alpha = \frac{1}{\sqrt{5}}, \quad \beta = \frac{2}{\sqrt{5}}, \quad q = 15. \tag{2.66}$$

To match the high order accuracy of spatial and time discretization, we first set

$$\Delta t = \mathrm{CFL} \times \left(\frac{1}{2\left(\frac{1}{\Delta x} + \frac{1}{\Delta y}\right)} \right)^{(k+1)/2}, \quad \Delta x = \Delta y = \frac{1}{2^N}, \tag{2.67}$$

where N is the maximum of mesh level, k is degree of Alpert's multiwavelet functions, and we pick the CFL number as following:

$$\mathrm{CFL} = \begin{cases} 0.3 & k = 1 \\ 1 & k = 2 \\ 2 & k = 3 \end{cases}$$

We compute the errors as in Table 2.20 at final time $T = 0.25$, using Alternating Flux I. We can observe reduction of order of convergence from $k + 1$ of regular DG method to $k + 1/2$ of SGDG method by $1/2$, given the property of SGDG method as discussed in e.g. [4], Theorem 3.4. This verifies the accuracy of the program.

N	H_z				E_x				E_y			
	L^2		L^∞		L^2		L^∞		L^2		L^∞	
P^1												
6	8.08e-04		2.89e-03		1.41e-03		3.97e-03		1.61e-03		5.05e-03	
7	2.71e-04	1.58	1.13e-03	1.35	5.19e-04	1.44	2.16e-03	0.88	6.11e-04	1.40	2.42e-03	1.06
8	5.99e-05	2.18	3.64e-04	1.63	2.10e-04	1.30	9.14e-04	1.24	2.33e-04	1.39	1.10e-03	1.14
9	2.16e-05	1.47	1.03e-04	1.82	7.72e-05	1.44	3.55e-04	1.36	8.19e-05	1.51	4.08e-04	1.43
10	6.01e-06	1.85	3.67e-05	1.49	2.78e-05	1.47	1.49e-04	1.25	2.88e-05	1.51	1.63e-04	1.33
P^2												
N	H_z				E_x				E_y			
	L^2		L^∞		L^2		L^∞		L^2		L^∞	
5	5.95e-04		1.27e-03		4.34e-04		1.09e-03		2.56e-04		8.53e-04	
6	8.23e-05	2.85	2.16e-04	2.55	7.09e-05	2.61	2.92e-04	1.90	5.13e-05	2.32	2.16e-04	1.98
7	1.07e-05	2.94	2.81e-05	2.94	1.09e-05	2.70	4.50e-05	2.70	8.34e-06	2.62	3.84e-05	2.49
8	1.34e-06	3.00	4.83e-06	2.54	1.69e-06	2.69	7.98e-06	2.50	1.45e-06	2.52	8.08e-06	2.25
9	1.69e-07	2.99	5.10e-07	3.24	2.84e-07	2.57	1.48e-06	2.43	2.59e-07	2.49	1.39e-06	2.54
P^3												
N	H_z				E_x				E_y			
	L^2		L^∞		L^2		L^∞		L^2		L^∞	
5	8.97e-04		1.39e-03		5.98e-04		9.08e-04		3.00e-04		5.14e-04	
6	5.67e-05	3.98	9.02e-05	3.94	3.78e-05	3.98	6.23e-05	3.86	1.90e-05	3.98	4.43e-05	3.54
7	3.65e-06	3.96	5.92e-06	3.93	2.44e-06	3.96	4.52e-06	3.79	1.23e-06	3.95	3.68e-06	3.59
8	2.28e-07	4.00	3.98e-07	3.90	1.53e-07	3.99	3.63e-07	3.64	7.79e-08	3.98	3.24e-07	3.50
9	1.43e-08	3.99	2.56e-08	3.96	9.64e-09	3.99	3.38e-08	3.43	5.06e-09	3.94	2.80e-08	3.53

Table 2.20: L^2 and L^∞ errors of H_z, E_x, E_y for Alternating Flux I case

2.6.3 Spatial Optical Soliton Propagation

In this subsection, we present the result of the adaptive SGDG schemes to simulate physically relevant problem of spatial optical soliton propagation. We start from the setup of the example in the dimensional form, and show the non-dimensionalized form afterwards based on which the actual simulation is conducted.

We first consider the spatial optical soliton propagation in realistic glasses, characterized by a three-pole Sellmeier linear dispersion, an instantaneous Kerr nonlinearity and a dispersive Raman nonlinearity as discussed in [2, 127]. The non-dimensionalized form of the equations is the following:

$$\begin{aligned}
\mu_0 \partial_t H_z &= \partial_y E_x - \partial_x E_y && \text{(2.68a)}\\
\partial_t D_x &= \partial_y H_z && \text{(2.68b)}\\
\partial_t D_y &= -\partial_x H_z && \text{(2.68c)}\\
\mathbf{D} &= \epsilon_0 \left[\epsilon_\infty \mathbf{E} + b \sum_{s=1}^{3} \mathbf{P}_s + a(1-\theta)|\mathbf{E}|^2 \mathbf{E} + a\theta Q \mathbf{E} \right] && \text{(2.68d)}\\
\partial_t \mathbf{P}_s &= \mathbf{J}_s, \quad s = 1,2,3, && \text{(2.68e)}\\
\partial_t \mathbf{J}_s &= -\gamma \mathbf{J}_s - \omega_{0s}^2 \mathbf{P}_s + \omega_{ps}^2 \mathbf{E}, \quad s = 1,2,3, && \text{(2.68f)}\\
\partial_t Q &= \sigma && \text{(2.68g)}\\
\partial_t \sigma &= -\gamma_v \sigma - \omega_v^2 Q + \omega_v^2 |\mathbf{E}|^2 && \text{(2.68h)}
\end{aligned}$$

with parameters

$$\omega_{01} = 2.7537 \times 10^{16} rad/s, \;\; \omega_{02} = 1.6205 \times 10^{16} rad/s, \;\; \omega_{03} = 1.9034 \times 10^{14} rad/s,$$

$$a = 1.89 \times 10^{-22} m^2/V^2, \quad \gamma_v = \frac{2}{\tau_2}, \;\; \omega_v = \sqrt{\frac{\tau_1^2 + \tau_2^2}{\tau_1^2 \tau_2^2}}, \;\; \tau_1 = 12.2 fs, \;\; \tau_2 = 32.2 fs,$$

and

$$\beta_1 = 0.69617, \;\; \beta_2 = 0.40794, \;\; \beta_3 = 0.89748, \;\; \omega_{ps} = \sqrt{\beta_{ps}}\omega_{0s}, \;\; \gamma_s = 0, \;\; s = 1,2,3,$$

$$\epsilon_\infty = 1, \;\; b = 1, \;\; \theta = 0.3.$$

The physical domain is $[0, 38]$ for x and $[-3, 3]$ for y, in unit of μm; however, to reduce the numerical oscillations from domain boundary, we set the computational domain to be larger, i.e. $[0, 60] \times [-4, 4]$ instead. On the left boundary $x = 0$, time-dependent hard source is injected to the magnetic field H_z, namely,

$$H_z(x = 0, y, t) = H_0 \sin(\omega_c t)\text{sech}(y/w), \tag{2.69}$$

where, $\omega_c = 4.35 \times 10^{15}$ rad/s is the carrier frequency, and $w = 667\ nm,\ H_0 = 4.77 \times 10^7\ A/m$ are the width and the magnitude of the incident wave, respectively.

To better understand expected behaviour of our numerical tests, recall that in uniform glasses, the Maxwell's equations (2.68) with the nonlinear constitutive laws reduce to the nonlinear Schrödinger equation (NLSE) under paraxial assumption [2]. The pulse provided by the hard source above has predictable propagation, as indicated by the accurate solution of NLSE. In fact, the normalized NLSE of $u = u(\zeta, \xi)$, i.e.

$$i\frac{\partial u}{\partial \zeta} = \frac{1}{2}\frac{\partial^2 u}{\partial^2 \xi} + |u|^2 u, \quad \zeta \in (0, +\infty),\ \xi \in (-\infty, +\infty), \tag{2.70}$$

with initial condition $u(0, \xi) = g(\xi),\ \xi \in (-\infty, +\infty)$ admits bright soliton solutions. Specifically, regarding our numerical tests, we considered

$$g(\xi) = g_1(\xi) := \eta\text{sech}\left(\eta(\xi - \xi_0)\right)\exp(-i\Lambda\xi - i\phi)$$

as initial condition, and the solution is given by

$$u_1(\zeta, \xi) = \eta\text{sech}\left(\eta(\xi - \xi_0 - \Lambda\zeta)\right)\exp(-i\Lambda\xi - i\phi + i \cdot \frac{\Lambda^2 - \eta^2}{2}),$$

which is called fundamental soliton; η, Λ, ϕ, ξ_0 are parameters. When $\Lambda = 0$, $|u_1(\zeta, \xi)|$ remains constant for all ζ, and one can see that the fundamental soliton propagates in the dispersive and weakly nonlinear medium without changing its amplitude, width or shape. To compare with following case, consider $\xi_0 = 0$ and $\eta = 1$. Instead, when

$$g(\xi) = g_2(\xi) := 2\text{sech}(\xi),$$

the solution becomes

$$u_2(\zeta,\xi) := 4\exp(-i\zeta/2)\frac{\cosh(3\xi)+3\exp(-4i\zeta)\cosh(\xi)}{\cosh(4\xi)+4\cosh(2\xi)+3\cos(4\zeta)}.$$

This solution is called second-order soliton, as the result of the interactions between two fundamental solitons. We present the magnitude of u_1 and u_2 in the following Figure 2.3.

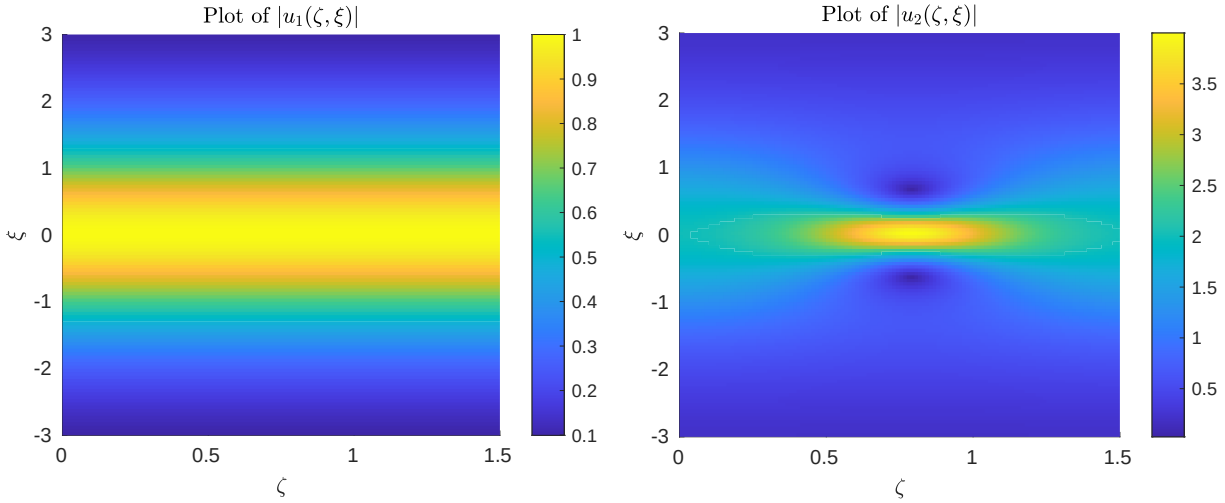


Figure 2.3: Magnitude $|u_1(\zeta,\xi)|$ of Fundamental Soliton and $|u_2(\zeta,\xi)|$ of Second Order Soliton, of Solutions of NLSE (2.70).

Before we simulate the fundamental and second order solitons, we need a final step to non-dimensionalize the Maxwell equation (2.68). Following the procedure desribed in [1], we choose reference space scale $x_0 = 8\mu m$, reference time scale $t_0 = x_0/c = 2.6685\times10^{-14}s$ with c the light speed, and reference value of H_z is $H_0 = 4.77\times10^7 A/m$. If reference value of electric field is E_0, then we have rescaled fields and constants defined as follows:

$$(\mathbf{H}/E_0)\sqrt{\mu_0/\epsilon_0}\to\mathbf{H},\ \mathbf{D}/(\epsilon_0E_0)\to\mathbf{D},\ \mathbf{P}/E_0\to\mathbf{P},$$

$$(\mathbf{J}/E_0)t_0\to\mathbf{J},\ \mathbf{E}/E_0\to\mathbf{E},\ Q/E_0^2\to Q,\ (\sigma/E_0^2)t_0\to\sigma,$$

$$aE_0^2\to a,\ \omega_*t_0\to\omega_*,\ *=0s,ps,v,\ s=1,2,3,\ \gamma_vt_0\to\gamma_v.$$

Note that here we use same notations for scaled and original variables. From the scaling of $\mathbf{H} = H_z$, we have $H_0 = E_0\sqrt{\epsilon_0/\mu_0} = E_0\epsilon_0c$, so $E_0 = 1.797\times10^{10}N/(A\cdot s)$.

Spatial computational domain is $[0, 60] \times [-4, 4]$ in unit of μm, or $[x_a, x_b] \times [y_a, y_b] = [0, 7.5] \times [0, 1]$, after nondimensionalization on both x, y direction and shifting on y direction by 0.5, on x-y plane, and computational domain is large enough in both x and y direction to avoid non-physical oscillation. Initially, all the fields are set to be zero. To make the best use of boundary condition provided as $H_z(x = 0, y, t)$ in (2.69), we consider the adaptive SGDG schemes with alternating numerical fluxes (2.35). Note that in these two cases, we do not need information of $\mathbf{E}(x = 0, y, t)$, since the flux regarding $\mathbf{E}$ on $x = 0$ along x direction is chosen to be right limit on $x = 0$, whose value is evaluated within the computational domain. We refer the interested readers to [1] for numerical boundary treatments suitable for other alternating fluxes. For simplicity we will only present the simulation results by the Alternating I numerical flux. And except for the injected boundary on $x = 0$, the other three edges of boundary are absorbing boundaries based on linearized system on characteristic direction, after neglecting the nonlinear effects of Kerr type and Raman scattering and the linear delayed response. The dimensionless form of these numerical flux follows as

1. at right boundary $x = x_b$ of x direction, $(H_z + \sqrt{\epsilon_\infty} E_y)^+ = (H_z + \sqrt{\epsilon_\infty} E_y)^-$ and $(H_z - \sqrt{\epsilon_\infty} E_y)^+ = 0$;

2. at left boundary $y = y_a$ of y direction, $(H_z + \sqrt{\epsilon_\infty} E_x)^+ = (H_z + \sqrt{\epsilon_\infty} E_x)^-$ and $(H_z - \sqrt{\epsilon_\infty} E_x)^- = 0$;

3. at left boundary $y = y_b$ of y direction, $(H_z - \sqrt{\epsilon_\infty} E_x)^+ = (H_z - \sqrt{\epsilon_\infty} E_x)^-$ and $(H_z + \sqrt{\epsilon_\infty} E_x)^+ = 0$.

In our simulation, we use P^2 Alpert's multiwavlet as basis of numerical solution, coupling with P^3 interpolatory multiwavelet to deal with nonlinearity. The interpolatory multiwavelet basis is chosen to be one degree higher than Alpert's, in order to eliminate numerical error of interpolation. Maximum mesh level is 11, i.e. the most refined Δx

and Δy might be $1/2^{11}$. Error thresholds for refinement and coarsening are $\varepsilon = 10^{-3}$ and $\eta = 10^{-4}$, respectively. The relative tolerance of nonlinear solver of PETSc is set to be 10^{-9}.

We consider first the case of fundamental soliton, where the non-dimensionalized form of initial condition is

$$H_z(x = 0, y, t) = \sin(\omega_c t)\mathrm{sech}(y/w),$$

with rescaled $\omega_c = 116.08$ and $w = 0.083375$. We plot the solutions at different time, and the centers of corresponding active elements for adaptive scheme in Figure 2.4. Note that in this case, the soliton during the propagation maintains magnitude and width as in first picture of Figure 2.3, while the error threshold allows error of specific magnitude to enter into the soliton solution. The distribution of active elements also provide important information of the soliton: more active elements accumulate around $y = 0.5$, where the magnitude of the magnetic field reaches its maximum, and fewer active elements support on region away from $y = 0.5$, since the soliton solution almost vanishes when y is close to its boundary. Along x direction, active elements and their support move along x direction as the soliton propagates. The adaptive scheme generates active elements that properly follow the propagation of the solution.

Next, we present several simulation results of non-dimensionalized initial condition

$$H_z(x = 0, y, t) = 2\sin(\omega_c t)\mathrm{sech}(y/w)$$

with same ω_c and w as in fundamental soliton case; see Figure 2.5. We focus on the region when x is smaller than $1/3$. This solution has changing magnitude and width, which is different from fundamental soliton. As predicted in NLSE (2.70) and Figure 2.3, this is the case when two fundamental solitons produce the second order soliton, with focusing and defocusing effect. Again the support of active elements match the region with larger $|H_z|$ magnitude, while at the region close to boundary of y direction or of large x, the solution almost vanishes and fewer elements are needed for good resolution of the solution.

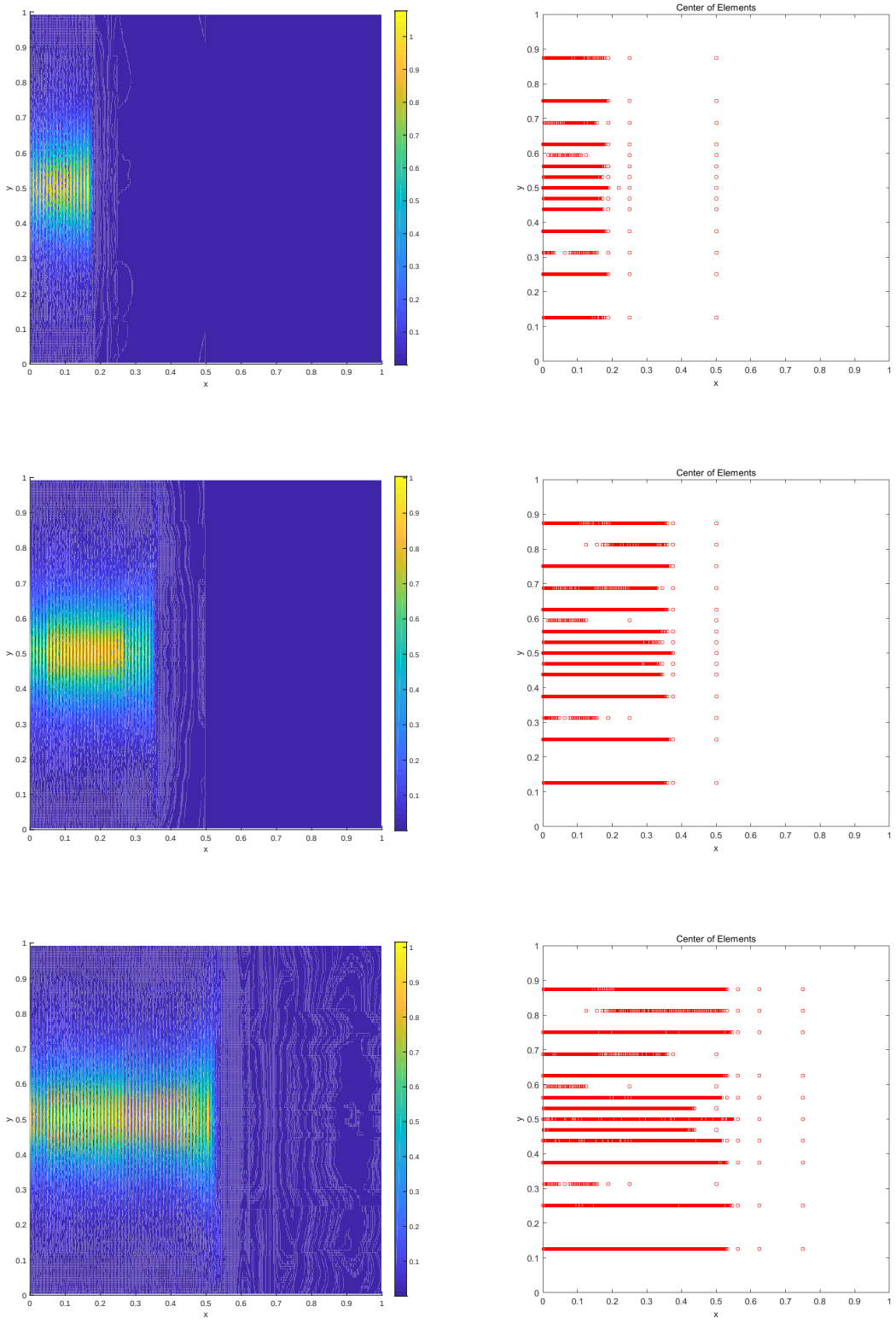


Figure 2.4: Comparison for fundamental soliton between absolute value $|H_z|$ of magnetic field (left column) and center of active elements (right column) at different time. First row: $T = 2$; Second row: $T = 4$; Third row: $T = 6$.

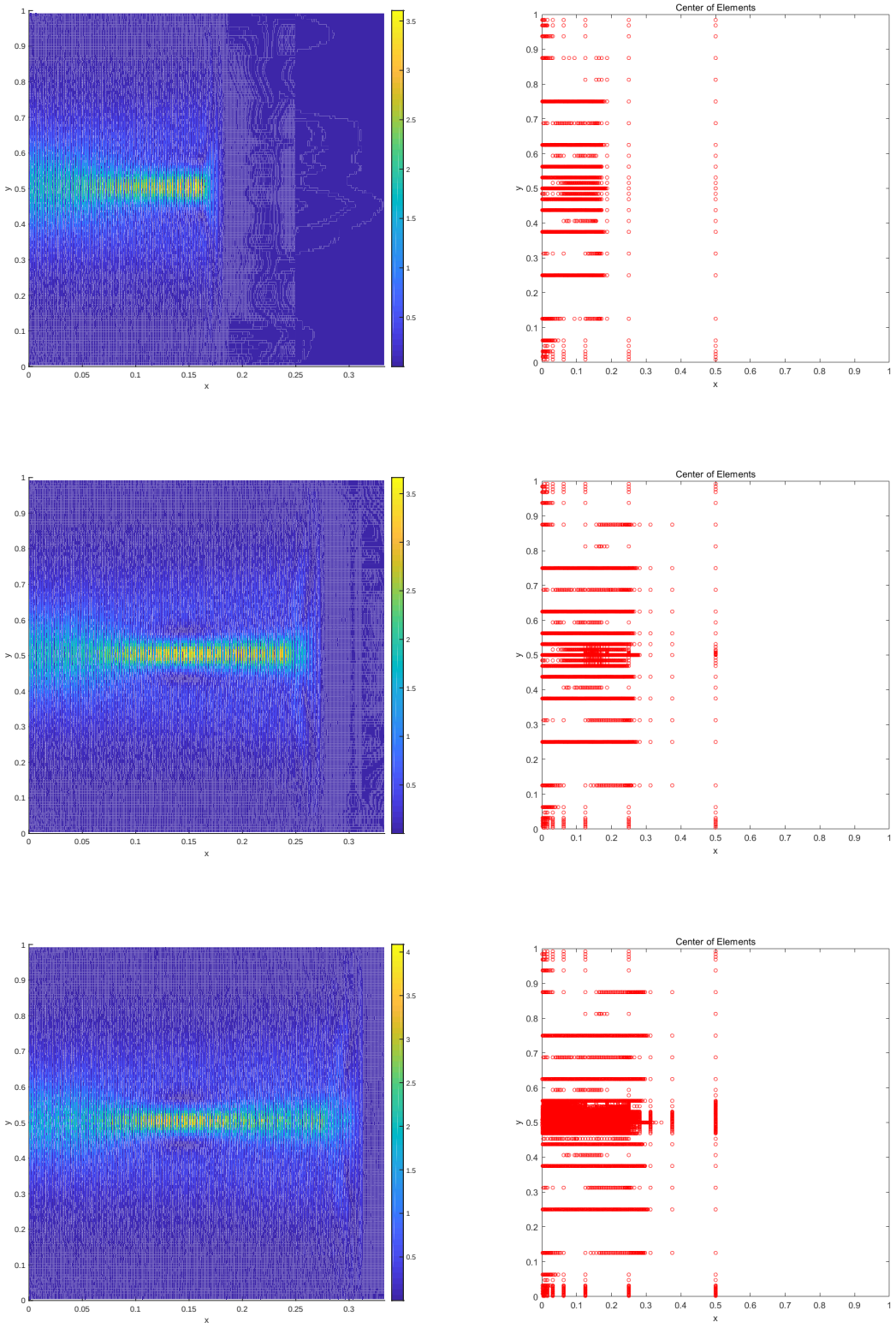


Figure 2.5: Comparison for second order soliton between absolute value $|H_z|$ of magnetic field (left column) and center of active elements (right column) at different time. First row: $T = 1.976$; Second row: $T = 3.024$; Third row: $T = 3.459$.

CHAPTER 3

APPLICATION OF WENO METHOD IN MATHEMATICAL MODEL OF ASYNCHRONOUS DATA FLOW IN PARALLEL COMPUTERS

3.1 Models of Extreme Scale Computers

It has been well-established that current and future generations of extreme scale computers have achieved and, for the foreseeable future, are expected to achieve increases in performance via greater levels of parallelism at multiple levels — e.g., within the processors as well as increasing the number of processors and nodes —as opposed to increases in clock speeds, which are expected to remain relatively flat. Additionally, extremely concurrent codes, involving dynamic parallelism and greater degrees of asynchronous parallel executions, are increasingly needed to leverage this large scale parallellism [128, 129].

As machine improvements depend on increasingly complex architectures and as additional constraints on system development and planning (such as power consumption [129]) arise, a need for predictive, quantitative models of computational performance will grow greater. Previously developed modeling tools such as LogP [130] result in easily evaluated models which can prove difficult to extend and modify. Alternatively, PRAM models have been used as abstractions of codes; these however have scalability issues due to the complexity of simulating them [131]. Other modern tools [132, 133] are similarly still limited to fine-grain simulations of at most a few dozen nodes, again due to their computational complexity during simulations.

Core counts are now in the hundreds of thousands and millions on machines in the TOP500 list of supercomputers; node counts consistently are in the thousands. Such numbers mean that fine-grained simulation tools (such as those listed above) are incapable of describing large-scale phenomena. Alternative approaches have been proposed to address these issues: miniapp codes can mimic key features of the performance of exascale

codes with a much smaller codebase [134]. Aspen, a framework for performance modeling [135, 136], uses a domain specific language which encodes both abstracted features of machines hardware and specific software applications to provide coarse-grained simulations. However, these suffer from the need to develop specialized simulation codes which can be problem dependent, resulting in possibly labor-intensive tools. A workflow modeling apporach, Pegasus, has been developed to model workflows using a graph-theoretic perspective to detect and manage anamolies in the computing environment [137].

We propose developing a macroscopic model of extreme scale computers which views such computing environments in a continuum framework. Such a model has several potential benefits: in addition to being computationally tractable, it will open up the possibility of using the theoretical tools of partial differential equations to understand and control the performance of high-performance computing systems. Specifically, our goal is to derive a fluid-limit model of data flow — which can be described by a partial differential equation — from a simplified deterministic model of data processing and flow in an extreme scale computer with interprocessor communications and asynchronous executions. Fluid models, beyond their obvious utility in physical systems, have been used to model flows in networks, such as vehicular traffic flows [138], supply chains [139], and gas networks [140, 141]. In particular, as discussed in [142] and [139], such fluid models lie at the end of a hierarchy of models which begin with microscopic or discrete models. That is, similar to the derivation of physical fluid laws from many body physics, one may derive continuum-level flow equations from discrete-level models of the dynamics of agent interactions. With such a model, standard numerical simulation tools and analytical methods may be brought to bear for studying large-scale phenomena in extreme-scale computing.

We begin in section 3.2 with a microscopic model of a network of processors performing a multi-stage computational task which necessitates inter-processor communications. In section 3.3, we derive a formal asymptotic limit of this agent-based model as the scale

of the system increases, resulting in an Eulerian fluid flow model. Along with the resulting nonlinear conservation law, we present a related Hamilton-Jacobi equation and establish the existence of solutions [103, 104] in section 3.3. In section 3.4, we illustrate the numerical methods we use to simulate discrete and continuum model, especially WENO interpolation for solving Hamilton-Jacobi equation [12, 11, 107, 99]. Finally in section 3.5, we present the results of numerical experiments, to show agreement between the microscopic and fluid models and then illustrate the behavior of solutions under heterogeneous computing layouts.

This chapter is based on published paper [10].

3.2 The Discrete Model

In this section, we introduce the microscopic model, which is based on a highly simplified, deterministic, semi-discrete ordinary differential equation (ODE). Imagine the computer as a network of processors $\{P_i\}_{i=1}^{i^{\max}}$ that are arranged in a one-dimensional, periodic lattice. The computer is assigned a computational job involving a sequence of $k^{\max}$ tasks which are identical in the sense that each one takes the same effort to complete. This computational job is divided by distributing data within the many processors. Denote by $q_{i,k}(t)$ the amount of data in P_i that sits in stage k at time t.

The dynamics of $q_{i,k}$ are given by a conservation law of the form

$$\dot{q}_{i,k}(t) = F_{i,k-1}(t) - F_{i,k}(t), \quad k = 1, \ldots, k^{\max}, \quad i = 1, \ldots i^{\max}, \tag{3.1}$$

where $F_{i,k}$ ($i = 1, \ldots, i^{\max}$, $k = 1, \ldots, k^{\max}-1$) is the rate of data moving in processor i from stage k to $k+1$, referred to as the *throughput*. At the first stage $k = 1$, $F_{i,0}$ ($i = 1, \ldots, i^{\max}$) is the rate of data being loaded into processor i to be processed, referred to as the *inflow*, and at the final stage, $F_{i,k^{\max}}$ ($i = 1, \ldots, i^{\max}$) is the rate of data completing the final stage of the job, referred to as the *outflow*. Equation (3.1) implies that the data in each processor is neither created or destroyed, only moved in and out of the processor or in between

stages; that is,

$$\frac{d}{dt}\left(\sum_{k=1}^{k^{\max}} q_{i,k}\right) = F_{i,0} - F_{i,k^{\max}}. \tag{3.2}$$

Another fundamental quantity of interest in the discrete model is $Q_{i,k}(t)$, which is defined as the amount of data at time t that has gone through the first $k-1$ stages of P_i. For each $t \geq 0$,

$$Q_{i,k}(t) = \left(\sum_{j=k}^{k^{\max}} q_{i,j}(t)\right) + \int_0^t F_{i,k^{\max}}(s)ds. \tag{3.3}$$

To determine the form of $F_{i,k}$, consider the case of with or without throttling. Without throttling, assume data can move in one processor P_i between tasks in the rate of a given *maximum throughput* $a_i \geq 0$, which is independent of the task number k. If we consider throttling, $F_{i,k}(t)$ does not reach the maximum a_i for two reasons considered here: self-throttling and neighbour-throttling.

1. **Self-throttling:** Given an amount of data $q_{i,k}$ to be processed at stage k in processor

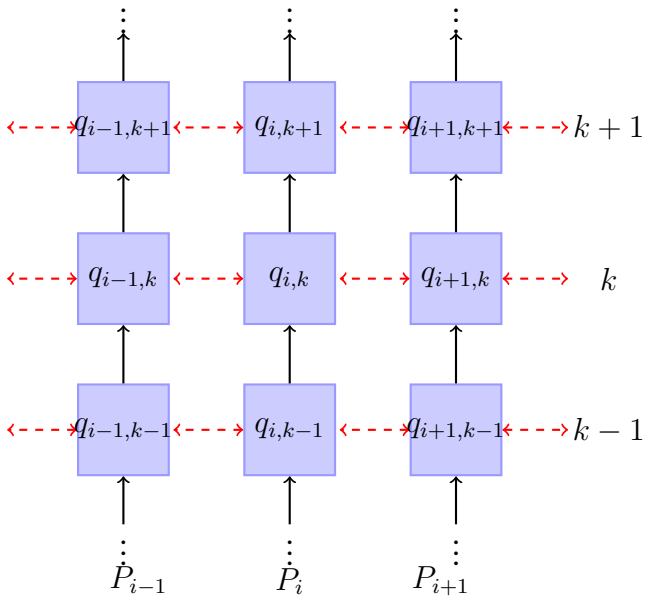


Figure 3.1: Schematic of network of processors. Dashed lines denote inter-processor communications

P_i, we define the self-throttling function

$$v_1(q_{i,k}; q_*) = \max\left\{0, \min\left\{1, \frac{q_{i,k}}{q_*}\right\}\right\}, \tag{3.4}$$

in the form of roof-line model. If no data is available to be processed, then $F_{i,k} = 0$; if the amount of data to be processed drops below a certain threshold $q_* > 0$, then P_i cannot maintain the throughput a_i and the throughput is reduced.

2. **Neighbor throttling:** As the computational task is not entirely parallel across processors, P_i requires sufficient information from its neighbors to perform task k at full throughput. The neighbor throttling function v_2 models this dependence. It gives the amount of available data on P_i at stage k

$$v_2(q_{i,k}, \Delta_{i+1,k}, \Delta_{i-1,k}; \beta) = \min\left\{q_{i,k}, \frac{1}{\beta}\max\{\Delta_{i+1,k}, 0\}, \frac{1}{\beta}\max\{\Delta_{i-1,k}, 0\}\right\}. \tag{3.5}$$

Here $\Delta_{i\pm1,k}$ denotes the data on the right/left neighbor which is available to be used by P_i to process $q_{i,k}$. The parameter $\beta \in (0, 1]$ allows for the possibility that computations do not rely in a one-to-one fashion upon the availability of data from neighbors. If $\Delta_{i\pm1,k} = 0$ the processing of data stops due to the absence of a necessary component of the computational task and so $F_{i,k} = 0$. Alternatively, if both $\Delta_{i+1,k}$ and $\Delta_{i-1,k}$ exceed $\beta q_{i,k}$, then P_i has sufficient data from its neighbors to process $q_{i,k}$ and no throttling occurs.

The data from the left/right neighbor which is available for processing at stage k is given by

$$\Delta_{i\pm1,k} = Q_{i\pm1,k} - Q_{i,k+1} = Q_{i\pm1,k} - (Q_{i,k} - q_{i,k}). \tag{3.6}$$

The data on each neighbor must have completed the same stage for it to be available; additionally, this data is not reused on P_i for the same stage. This means that the data available to be used from the neighbors can be written as above and so the amount of data available to be processed on P_i at stage k is given by

$$v_2\big(q_{i,k}, Q_{i+1,k} - Q_{i,k} + q_{i,k}, Q_{i-1,k} - Q_{i,k} + q_{i,k}; \beta\big). \tag{3.7}$$

The throughput $F_{i,k}$ is a composition of the throttling functions v_1 and v_2:

$$F_{i,k} = a_i v_1\Big(v_2\big(q_{i,k}, Q_{i+1,k} - Q_{i,k} + q_{i,k}, Q_{i-1,k} - Q_{i,k} + q_{i,k}; \beta\big); q_*\Big). \tag{3.8}$$

At first glance, this definition of $F_{i,k}$ appears circular since it depends on $Q_{i,k}$, which in turn depends on $F_{i,k^{\max}}$. However, as a consequence of the conservation law (3.2),

$$\int_0^t F_{i,k^{\max}}(s)ds = \int_0^t F_{i,0}(s)ds + \sum_{j=1}^{k^{\max}} q_{i,j}(0) - \sum_{j=1}^{k^{\max}} q_{i,j}(t) \tag{3.9}$$

Thus to complete the model, we need only prescribe initial data $q_{i,k}(0)$ and the inflow $F_{i,0}$. To prescribe the inflow, we specify $q_{i,0}$ and then let $F_{i,0}$ be evaluated according to (3.8).

Theorem 3.2.1 ([10], Proposition 2.1) *The system* (3.1) *with (i) throughput $F_{i,k}$ defined in* (3.8) *for $i = 1, \dots, i^{\max}$ and $k = 1, \dots, k^{\max}$; (ii) prescribed initial data $q_{i,k}(0)$ for $i = 1, \dots, i^{\max}$ and $k = 1, \dots, k^{\max}$; and (iii) prescribed inflow data $F_{i,0}$ for $i = 1, \dots, i^{\max}$ and $t \geq 0$ has a unique solution for all $t \geq 0$. Moreover, if $q_{i,k}(0) \geq 0$ for all $i = 1, \dots, i^{\max}$ and $k = 1, \dots, k^{\max}$, then $q_{i,k}(t) \geq 0$ for all $t \geq 0$ and $i = 1, \dots, i^{\max}$ and $k = 1, \dots, k^{\max}$.*

3.3 The Continuum Model

In this section, we consider a formally accurate continuum model in the limit as number of processors $i^{\max}$ and number of tasks $k^{\max}$ tend to infinity. So far to our knowledge, there is no immediate conclusion of existence and uniqueness for such model, so a Hamilton-Jacobi equation is demonstrated at the end of this section. The extensive theory of viscosity solutions of Hamilton-Jacobi equation leads to promising result of existence and uniqueness of the model. Higher dimension model can be derived in the similar manner; for details, see [10] Section 3.3.

For given $i^{\max}, k^{\max}$, we define the quantities:

$$\delta := (k^{\max})^{-1}, \quad \varepsilon := (i^{\max})^{-1}, \quad \eta := \frac{\varepsilon}{\delta} = \frac{k^{\max}}{i^{\max}}. \tag{3.10}$$

Here, δ is the fraction of the work done in each stage and ε is the average amount of data in a processor. In following paragraphs, we provide a brief discussion of derivation of the continuum model, by $i^{\max}$ and $k^{\max}$ tending to infinity. We assume, in taking this limit, that the job performed by the computer is fixed – that is, the total amount of work does not change.

To derive such a continuum model, we first express the ODE (3.1) in terms of the following $O(1)$ quantities:

$$r_* := \frac{q_*}{\varepsilon\delta}\,, \quad r_{i,k} := \frac{q_{i,k}}{\varepsilon\delta}\,, \quad R_{i,k} := \frac{1}{\varepsilon}Q_{i,k}\,, \quad D^{\pm}_{i,k} := \pm\frac{R_{i\pm1,k} - R_{i,k}}{\varepsilon}\,, \quad \alpha_i := \frac{a_i}{\varepsilon}\,. \tag{3.11}$$

Define the rescaled throttling functions

$$w_1(r, r_*) = \max\left\{0, \min\left\{1, \frac{r}{r_*}\right\}\right\}. \tag{3.12a}$$

$$w_2\left(r, D^-, D^+; \eta, \beta\right) = \min\left\{r, \frac{1}{\beta}\max\{\eta D^+ + r, 0\}, \frac{1}{\beta}\max\{\eta D^- + r, 0\}\right\} \tag{3.12b}$$

and the composite function

$$w\left(r, D^-, D^+; r_*, \alpha, \eta, \beta\right) := \alpha w_1\left(w_2(r, D^-, D^+; \eta, \beta); r_*\right). \tag{3.13}$$

The dynamics in (3.8) can now be re-expressed in terms of the $O(1)$ quantities in (3.11), thereby obtaining an evolution formula for $r_{i,k}$:

$$\dot{r}_{i,k}(t) = \frac{f_{i,k-1}(t) - f_{i,k}(t)}{\delta} \tag{3.14}$$

$$f_{i,k}(t) = w\left(r_{i,k}(t), -D^-_{i,k}(t), D^+_{i,k}(t); r_*, \alpha_i, \eta, \beta\right) \tag{3.15}$$

for $i = 1, \ldots, i^{\max}$, $k = 0, \ldots, k^{\max}$, and $r_{i,0}$ is prescribed for $i = 1, \ldots, i^{\max}$.

The next step is to interpret (3.14) as a conservative finite-difference formula for a sufficiently smooth function $\rho = \rho(x, y, t)$, defined on $[0, 1) \times [0, 1] \times [0, \infty)$, such that

$$\rho(x_i, z_k, t) = r_{i,k}(t), \tag{3.16}$$

on grid points

$$x_i = (i - 0.5)\,\varepsilon \quad \text{and} \quad z_k = k\delta, \tag{3.17}$$

for $i = 1, \ldots, i^{\max}$ and $k = 0, \ldots, k^{\max}$. We also let $\alpha = \alpha(x)$ be a continuous function such that $\alpha(x_i) = \alpha_i$. Let the function $\phi = \phi(x, z, t)$ interpolate the fluxes on the same grid points:

$$\phi(x_i, z_k, t) = f_{i,k}(t), \tag{3.18}$$

for $i = 1, \ldots, i^{\max}, k = 1, \ldots, k^{\max}, t \geq 0$, and

$$P(x, z, t) = \int_z^1 \rho(x, \xi, t) d\xi + \int_0^t \phi(x, 1, s) ds, \tag{3.19}$$

and we find that both

$$\Phi^{(1)}(\rho(x_i, z_k, t), \partial_x P(x_i, z_k, t), \partial_x^2 P(x_i, z_k, t); r_*, a, \eta, \beta) \tag{3.20}$$

and

$$\Phi^{(0)}(\rho(x_i, z_k, t), \partial_x P(x_i, z_k, t); r_*, a, \eta, \beta) \tag{3.21}$$

approximate

$$w(r_{i,k}(t), D_{i,k}^-(t), D_{i,k}^+(t); r_*, \alpha_i, \eta, \beta), \tag{3.22}$$

when $0 \ll \varepsilon, \delta \ll 1$, where

$$\Phi^{(0)}\left(\rho, \partial_x P; r_*, \alpha, \eta, \beta\right) = w\left(\rho, -\partial_x P, \partial_x P; r_*, \alpha, \eta, \beta\right) \tag{3.23a}$$

$$\Phi^{(1)}\left(\rho, \partial_x P, \partial_x^2 P; r_*, \alpha, \eta, \beta\right) = w\left(\rho, -\partial_x P + \frac{\varepsilon}{2}\partial_x^2 P, \partial_x P + \frac{\varepsilon}{2}\partial_x^2 P; r_*, \alpha, \eta, \beta\right). \tag{3.23b}$$

Thus for $0 \ll \varepsilon, \delta \ll 1$, with $\eta \in (0, \infty)$ fixed, the weak form of (3.14) is formally consistent with the continuum model

$$\partial_t \rho + \partial_z \Phi^{(\ell)}(\rho, \partial_x P, \partial_x^2 P; r_*, a, \eta, \beta) = 0, \qquad (x, z, t) \in \mathbb{T}^1 \times (0, 1) \times (0, \infty), \tag{3.24a}$$

$$\rho(x, 0, t) = \rho_{\mathrm{bc}}(x, t), \qquad (x, t) \in \mathbb{T}^1 \times (0, \infty), \tag{3.24b}$$

$$\rho(x, z, 0) = \rho_0(x, z), \qquad (x, z) \in \mathbb{T}^1 \times (0, 1) \tag{3.24c}$$

where

$$P(x, z, t) = \int_z^1 \rho(x, \xi, t) d\xi + \int_0^t \phi^{(\ell)}(x, 1, s) ds, \tag{3.25a}$$

$$\phi^{(\ell)}(x, z, t) = \Phi^{(\ell)}\left(\rho(x, z, t), \partial_x P(x, z, t), \partial_x^2 P(x, z, t); r_*, \alpha, \eta, \beta\right), \tag{3.25b}$$

and $\Phi^{(\ell)}$, $\ell \in \{0, 1\}$, is given in (3.23). For the sake of compactness, we have slightly abused notation in (3.24a), as the definition of $\Phi^{(0)}$ is independent of $\partial_x^2 P$. Additionally, we have identified $[0, 1)$ with the one-dimensional torus $\mathbb{T}^1$ in order to reflect the periodic layout of the processors.

As in the discrete case, it may appear that the model in (3.24) is circular due to the definition of P in (3.25a). However, as with F in (3.9), $\Phi^{(\ell)}$ can be unwrapped, this time using the conservation law (3.24a); that is

$$\int_0^t \phi^{(\ell)}(x, 1, s)ds = \int_0^t \phi(x, 0, s)ds + \int_0^1 \rho_0(x, \xi)d\xi - \int_0^1 \rho(x, \xi, t)d\xi \tag{3.26}$$

Thus the continuum model is complete once initial condition ρ_0 and inflow condition $\phi_{\mathrm{bc}} := \phi(\cdot, 0, \cdot)$ are specified. In practice, ρ_{bc} is prescribed and then ϕ_{bc} is evaluated using (3.25b) and (3.23).

Furthermore, integrating (3.24a) with respect to z gives

$$\partial_t \int_z^1 \rho(x, \xi, t)d\xi + \Phi^{(\ell)}(x, 1, t) - \Phi^{(\ell)}(x, z, t) = 0 \tag{3.27}$$

Meanwhile, differentiating (3.25a) gives

$$\partial_t P(x, z, t) = \partial_t \int_z^1 \rho(x, \xi, t)d\xi + \Phi^{(\ell)}(x, 1, t) \tag{3.28}$$

Combining (3.27) and (3.28) and using the fact that $\rho = -\partial_z P$ gives a closed Hamilton-Jacobi equation for P with initial and boundary conditions that are derived by applying the definition of P in (3.25a) to (3.24c) and (3.24b), respectively. The complete model is, for some $T > 0$,

$$\partial_t P - \Phi^{(\ell)}(-\partial_z P, \partial_x P, \partial_x^2 P; r_*, \alpha, \eta, \beta) = 0, \quad (x, z, t) \in \mathbb{T}^1 \times (0, 1) \times (0, T), \tag{3.29a}$$

$$P(x, 0, t) - \int_0^1 \rho_0(x, \xi)d\xi - \int_0^t \phi_{\mathrm{bc}}(x, s)ds = 0, \quad (x, t) \in \mathbb{T}^1 \times (0, T), \tag{3.29b}$$

$$P(x, z, 0) - \int_z^1 \rho_0(x, \xi)d\xi = 0, \quad (x, z) \in \mathbb{T}^1 \times (0, 1), \tag{3.29c}$$

where (3.29b) is derived by integrating (3.24b) over $z \in (0, 1)$ and applying (3.26).

The existence and uniqueness of viscosity solution follows:

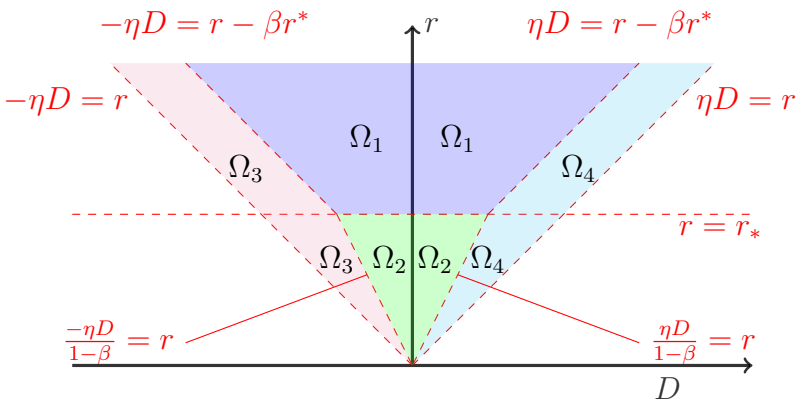


Figure 3.2: Flux $\Phi^{(0)}$ defined in (3.30)

Theorem 3.3.1 ([10], Theorem 3.1) *Assume that α and ρ_0 are (i) non-negative, (ii) uniformly Lipschitz in their arguments, and (iii) periodic in x (that is, $\alpha(0) = \alpha(1)$ and $\rho_{\text{bc}}(0,t) = \rho_{\text{bc}}(1,t)$ for all $t \in [0,T]$). Further, assume that there is an M where $\int_0^T \phi_{\text{bc}}(x,s)ds \leq M$ for all $x \in \mathbb{T}^1$. Then there exists a unique, continuous, viscosity solution (in the sense of [103]) to* (3.29).

We use the flux function $\Phi^{(0)}$ for all of the numerical simulations in Section 3.5. This function is a piecewise constant that can be expressed in the following form:

$$\Phi^{(0)}(r, D; r_*, \alpha, \beta) = \begin{cases} \alpha & (r,D) \in \Omega_1, \\ \dfrac{\alpha r}{r_*} & (r,D) \in \Omega_2, \\ \dfrac{\alpha(r+\eta D)}{\beta r_*} & (r,D) \in \Omega_3, \\ \dfrac{\alpha(r-\eta D)}{\beta r_*} & (r,D) \in \Omega_4, \end{cases} \tag{3.30}$$

where the subdomains Ω_i are depicted in Figure 3.2.

3.4 Numerical Methods for Simulations

In this section, we perform numerical simulations of the one dimensional processor system in order to (i) test the ability of the macroscopic model to approximate the discrete model when ε and δ are small and (ii) explore how model parameters affect the model output. Problem data is specified in terms of continuum models quantities. These quantities are translated back to discrete model quantities in order to implement ODE simulations.

3.4.1 ODE Implementation

The explicit two-step Adams-Bashforth (Section III of [143]) is used to simulate the discrete model formed by (3.1), (3.3), and (3.8). Given η, values $i^{\max}$ and $k^{\max}$ are chosen so that $k^{\max}/i^{\max} = \eta$ as in (3.10). We then compute a solution to the discrete model as follows. Using (3.11) and (3.16), we convert $r_*, a, \rho_0, \rho_{bc}$ to their discrete counterparts:

$$q_* = \varepsilon\delta r_*, \qquad q_{i,k}(0) = \varepsilon\delta\rho_0(x_i, z_k), \qquad q_{i,0}(t) = \rho_{\mathrm{bc}}(x_i, t), \quad a_i = \varepsilon\alpha(x_i). \tag{3.31}$$

This discrete model data is used to set the time step:

$$\Delta t = \frac{q_*}{2\big(\max_i a_i\big)\sqrt{i^{\max}k^{\max}}}. \tag{3.32}$$

The outflow at $F_{i,k^{\max}}$ is tracked and accumulated over time in order to compute $Q_{i,k}$ from (3.3). At the final time T, the result of the explicit time stepping is converted back, via the formula in (3.11), i.e., $r_{i,k}(T) = (\varepsilon\delta)^{-1}q_{i,k}(T)$. In order to compare this against solutions to the continuum model (see below), we use these point-wise values to generate a piecewise constant function r over the cells $C_{i,k} = (x_i - .5\varepsilon, x_i + .5\varepsilon) \times (z_k, z_k + \delta)$:

$$r(x, z) = \sum_{i,k} r_{i,k}\chi_{C_{i,k}}(x, z). \tag{3.33}$$

3.4.2 Hamilton-Jacobi Implementation

The Hamilton Jacobi equation (3.29) is solved numerically using a fifth-order WENO interpolation in x and z and the optimal third-order SSP Runge-Kutta method for time integration. Details of these algorithms can be found in Sections 3.2 and 6, respectively, of [11]. Once a numerical solution for P is computed, we again use WENO interpolation to approximate ρ via the relation $\rho(x, z, t) = -\partial_z P(x, z, t)$.

To condense the notation, let $\sigma = \partial_x P$, $\tau = \partial_z P$ and $\upsilon = \partial_{xx} P$. Then for fixed r_*, α, η, β, and ℓ, let $H(\sigma, \tau, \upsilon) = -\Phi^{(\ell)}(-\tau, \sigma, \upsilon; r_*, a, \eta, \beta)$. The numerical solution for P is computed

on a grid $\{x_n, z_m\}$ where

$$x_n = n\Delta x, \qquad n = 1, 2, \ldots, N, \qquad \Delta x = N^{-1}, \tag{3.34}$$

$$z_m = m\Delta z, \qquad m = 1, 2, \ldots, M, \qquad \Delta z = M^{-1}. \tag{3.35}$$

The semi-discrete method for the grid function $P_{n,m}(t) \approx P(x_n, z_m, t)$ is

$$\frac{d}{dt}P_{n,m}(t) = -\hat{H}(\sigma^-_{n,m}, \sigma^+_{n,m}, \tau^-_{n,m}, \tau^+_{n,m}; \upsilon_{n,m}), \tag{3.36}$$

where the numerical approximations $\sigma^{\pm}_{n,m} \approx \sigma(x^{\pm}_n, z_m)$ and $\tau^{\pm}_{n,m} \approx \tau(x_n, z^{\pm}_m)$ are obtained via WENO interpolation and $\upsilon_{n,m} \approx \upsilon(x_n, z_m)$ is computed by central difference. The numerical flux function $\hat{H}$, based on the global Lax-Friedrichs flux:

$$\hat{H}(\sigma^-, \sigma^+, \tau^-, \tau^+; \upsilon) = H\left(\frac{\sigma^- + \sigma^+}{2}, \frac{\tau^- + \tau^+}{2}, \upsilon\right) - \frac{1}{2}\lambda^x(\sigma^+ - \sigma^-) - \frac{1}{2}\lambda^z(\tau^+ - \tau^-), \tag{3.37}$$

where

$$\lambda^x = \max_{\sigma,\tau}|H_\sigma| = \frac{\alpha\eta}{\beta r_*}, \quad \lambda^z = \max_{\sigma,\tau}|H_\tau| = \frac{\alpha}{\beta r_*}. \tag{3.38}$$

The time step for the SSP integrator is given by

$$\Delta t\left(\frac{\lambda^x}{\Delta x} + \frac{\lambda^z}{\Delta z}\right) \leq 0.6. \tag{3.39}$$

3.4.3 WENO Interpolation

Here we demonstrate how to compute $\sigma^{\pm}$ and $\tau^{\pm}$ using fifth order WENO interpolation [11]. For simplicity, we only consider $\sigma^{\pm}_{i,j}$ for fixed $i,\ j$, which approximate $\partial_x P(x, z_j)$ from the left or right limit of $x = x_i$, and we consider $\sigma^-_{i,j}$ first. For fixed i, j, each of stencils

$$X_0 = \{x_{i-3}, x_{i-2}, x_{i-1}, x_i\},\ X_1 = \{x_{i-2}, x_{i-1}, x_i, x_{i+1}\},\ X_2 = \{x_{i-1}, x_i, x_{i+1}, x_{i+2}\},$$

and function values of $P(\cdot, z_j)$ evaluated on the stencil, we can find a Lagrangian interpolating polynomial Q_k of degree 3, $k = 0, 1, 2$, i.e.

$$Q_k(x_{i-3+k+r}) = P(x_{i-3+k+r}, z_j), \quad r = 0, 1, 2, 3.$$

Then

$$\sigma_{i,j}^{-,k} := \frac{d}{dx} Q_k(x)\Big|_{x=x_i}$$

approximate $\partial_x P(x_i^-, z_j)$; note that $\sigma_{i,j}^{-,k}, k = 0, 1, 2$ are the three possible interpolations of third order ENO procedure. Let

$$\Delta_x^+ P_{i,j} := P_{i+1,j} - P_{i,j} = P(x_{i+1}, z_j) - P(x_i, z_j)$$

denote the standard forward difference operator; here we omit variable t of function P for convenience. Direct calculations imply

$$\sigma_{i,j}^{-,0} = \frac{1}{3}\frac{\Delta_x^+ P_{i-3,j}}{\Delta x} - \frac{7}{6}\frac{\Delta_x^+ P_{i-2,j}}{\Delta x}\frac{\Delta_x^+ P_{i-1,j}}{\Delta x} \tag{3.40}$$

$$\sigma_{i,j}^{-,1} = \frac{-1}{6}\frac{\Delta_x^+ P_{i-2,j}}{\Delta x} + \frac{5}{6}\frac{\Delta_x^+ P_{i-1,j}}{\Delta x} + \frac{1}{3}\frac{\Delta_x^+ P_{i,j}}{\Delta x} \tag{3.41}$$

$$\sigma_{i,j}^{-,2} = \frac{1}{3}\frac{\Delta_x^+ P_{i-1,j}}{\Delta x} + \frac{5}{6}\frac{\Delta_x^+ P_{i,j}}{\Delta x} - \frac{1}{6}\frac{\Delta_x^+ P_{i+1,j}}{\Delta x}. \tag{3.42}$$

WENO procedure uses a convex combination of $\sigma_{i,j}^{-,k},\ k = 0, 1, 2$ as final approximation of $\sigma_{i,j}^-$, i.e.

$$\sigma_{i,j}^- = w_0 \sigma_{i,j}^{-,0} + w_1 \sigma_{i,j}^{-,1} + w_2 \sigma_{i,j}^{-,2}, \tag{3.43}$$

where $w_0, w_1, w_2 \geq 0$ are nonlinear weights whose sum is 1. The key ingredient of WENO interpolation is to choose these nonlinear weights in the following way:

1. In smooth regions, w_0, w_1, w_2 should be very close to the optimal linear weights:

   $$w_0 = 0.1 + O(\Delta x^2),\ \ w_1 = 0.6 + O(\Delta x^2),\ \ w_2 = 0.3 + O(\Delta x^2),$$

   which makes $\sigma_{i,j}^-$ defined by (3.43) fifth order accurate on stencil $\{x_{i-3}, x_{i-2}, \ldots, x_{i+2}\}$ in approximating $\partial_x P(x_i, z_j)$ in smooth regions of $P(\cdot, z_j)$;

2. When stencil X_k contains discontinuity in the x derivative of P, the corresponding weight w_k should be close to zero, so that stencil on non-smooth region has little contribution to $\sigma_{i,j}^-$. The choice of weight in [107] satisfies $w_k = O(\Delta x^4)$ in this case.

To find such nonlinear weight, a smoothness indicator is introduced to measure how smooth the function being interpolated is inside the interpolation stencil. As illustrated in [107], the smoothness indicator is a scaled sum of the squares of the L^2 norms of the second and higher derivatives of the interpolation polynomial on the target interval, i.e.

$$IS_0 = 13(a-b)^2 + 3(a-3b)^2, \tag{3.44}$$

$$IS_1 = 13(b-c)^2 + 3(b+c)^2, \tag{3.45}$$

$$IS_2 = 13(c-d)^2 + 3(3c-d)^2, \tag{3.46}$$

where

$$a = \frac{\Delta_x^2 P_{i-2,j}}{\Delta x^2}, \quad b = \frac{\Delta_x^2 P_{i-1,j}}{\Delta x^2}, \quad c = \frac{\Delta_x^2 P_{i,j}}{\Delta x^2}, \quad d = \frac{\Delta_x^2 P_{i+1,j}}{\Delta x^2}, \tag{3.47}$$

and Δ_x^2 represents the central difference, i.e.

$$\Delta_x^2 P_{i,j} = P_{i+1,j} - 2P_{i,j} + P_{i-1,j}.$$

Then the nonlinear weights are

$$w_k = \frac{\widetilde{w}_k}{\widetilde{w}_0 + \widetilde{w}_1 + \widetilde{w}_2}, \quad k = 0, 1, 2,$$

where we usually choose $\varepsilon = 10^{-6}$, and

$$\widetilde{w}_0 = \frac{1}{(\varepsilon + IS_0)^2}, \quad \widetilde{w}_1 = \frac{6}{(\varepsilon + IS_1)^2}, \quad \widetilde{w}_2 = \frac{3}{(\varepsilon + IS_2)^2}.$$

We obtain the fifth order WENO approximation as

$$\sigma_{i,j}^- = 12\left(-\frac{\Delta_x^+ P_{i-2,j}}{\Delta x} + 7\frac{\Delta_x^+ P_{i-1,j}}{\Delta x} + 7\frac{\Delta_x^+ P_{i,j}}{\Delta x} - \frac{\Delta_x^+ P_{i+1,j}}{\Delta x}\right) - \Phi^{WENO}(a,b,c,d),$$

with

$$\Phi^{WENO}(a,b,c,d) := \frac{1}{3}w_0(a-2b+c) + \frac{1}{6}\left(w_2 - \frac{1}{2}\right)(b-2c+d),$$

and symmetrically, the approximation to the right derivative is

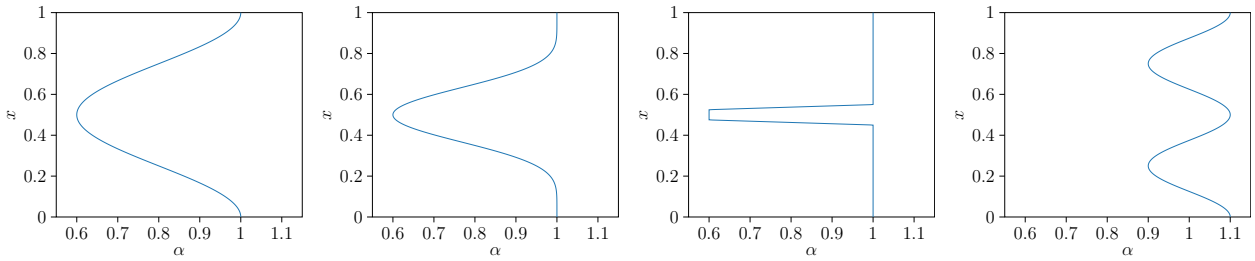
$$\sigma_{i,j}^+ = 12\left(-\frac{\Delta_x^+ P_{i-2,j}}{\Delta x} + 7\frac{\Delta_x^+ P_{i-1,j}}{\Delta x} + 7\frac{\Delta_x^+ P_{i,j}}{\Delta x} - \frac{\Delta_x^+ P_{i+1,j}}{\Delta x}\right) - \Phi^{WENO}(e,d,c,b),$$

with a, b, c, d in (3.47) and

$$e = \frac{\Delta_x^2 P_{i+2,j}}{\Delta x^2}.$$

3.5 Numerical Experiments

We perform a sequence of exploratory experiments below, modifying the parameters η and β, as well as the throughput function α. In all cases, α, ρ_0, and ρ_{bc} are periodic with respect to x and the parameter $r_* = 1$. Results are presented as two-dimensional color maps or line-outs in the z direction. In all figures, the horizontal axis corresponds to the z-axis. Profiles of α for each experiment are depicted in fig. 3.3.



(a) Example 1 (b) Examples 2-3 (c) Example 4 (d) Example 5

Figure 3.3: Profiles of the processor speed α used in the numerical experiments. The non-standard orientation of the graphs is set to match the axes in the numerical results that follow.

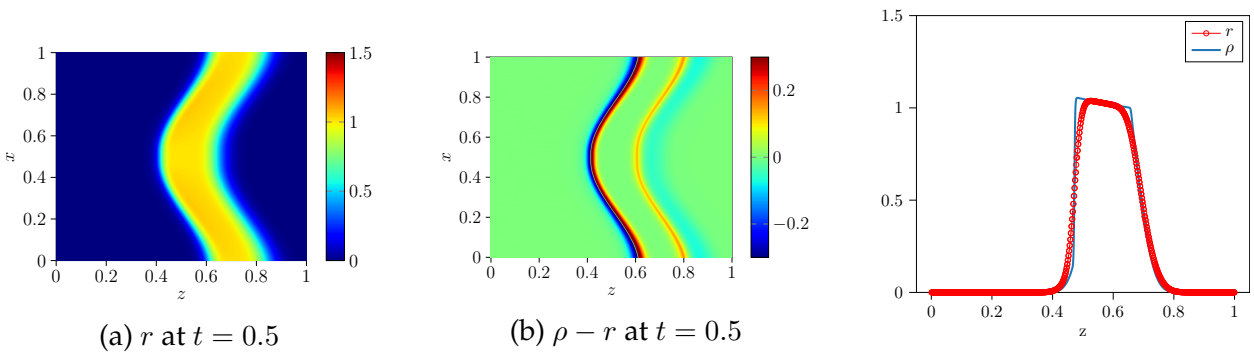
Example 3.5.1 (Agreement between models) *The purpose of this example is to demonstrate that the macroscopic model approximates the microscopic model when ε and δ are sufficiently small. We set $\beta = 1$ and consider $\eta \in \{0.2, 1, 5\}$. The initial condition, boundary condition, and processor speed are given by*

$$\rho_0(x,z) = 1.5\,(\sin(2\pi z))^6\,\chi_{[0,0.5]}(z), \quad \rho_{\mathrm{bc}}(x,t) = 0, \quad \alpha(x) = 1 - 0.4(\sin(\pi x))^2, \tag{3.48}$$

respectively. Both models are simulated up to a final time $t = 0.5$.

For this example, the Hamilton-Jacobi simulation is performed with a 1000×1000 mesh and a time step chosen according to (3.39) *in order to generate a highly resolved numerical solution of the macroscopic model. For the microscopic model, we use $i^{\max} = 1000$ and $k^{\max} = 200$ when $\eta = 0.2$, $i^{\max} = k^{\max} = 500$ when $\eta = 1$, and $i^{\max} = 200$ and $k^{\max} = 1000$ when $\eta = 5$. These solutions to the microscopic model are then used to obtain the piecewise-constant function r on the 1000×1000 mesh from the Hamilton-Jacobi simulation.*

Numerical results for $\eta = 0.2$, $\eta = 1.0$, and $\eta = 5.0$ are shown in fig. B.1, fig. B.2, and fig. B.3, respectively. While the results demonstrate general qualitative agreement between the models, discrepancies develop over time, especially for smaller values of η; see figs. B.1i and B.1l. For the worst case scenario ($\eta = 0.2$),we note that the $k^{\max}$ is relatively small, suggesting that the asymptotic analysis is less relevant. Indeed, in this case the microscopic model displays a diffusive behavior similar to that encountered with under-resolved advective numerical schemes. In response to this error, we increase the size of the discrete model by a factor of 2.5 (giving $i^{\max} = 2500$ and $k^{\max} = 500$), at which point the discrepancy between models decreases noticeably; see the first row of fig. 3.4. It is possible that an increase in the order of the approximation of the model with respect to z-advection (similar to that used to obtain $\Phi^{(0)}$) may be needed to avoid such cases for low η models with relatively low $k^{\max}$. Such a modification remains open.



(a) r at $t = 0.5$ (b) $\rho - r$ at $t = 0.5$ (c) ρ,r at $(x, t) = (0.3, 0.5)$

Figure 3.4: Comparison of the discrete model with $(i^{\max}, k^{\max}) = (2500, 500)$ and the continuum model for $\eta = 0.2$ at time $t = 0.5$. As expected, the discrete model shows better agreement with the continuum model than the previous version with only $(i^{\max}, k^{\max}) = (1000, 200)$ processors and stages; cf. fig. B.1

For the remaining examples, the Hamilton-Jacobi simulations are performed on a coarser mesh of 100×100.

Example 3.5.2 (Variations in η) *In this example, we examine the effect of η on solutions to the macroscopic model while $\beta = 1.0$ is fixed. The initial condition, boundary condition, and processor*

speed are given by

$$\rho_0(x,z) = 1.5\chi_{z\leq 0.2}(x,z) \quad \rho_{\text{bc}}(x,t) = 0, \quad \alpha(x) = 1 - 0.4(\sin(\pi x))^6, \tag{3.49}$$

respectively. It is expected that the slower processor speed around $x = 0.5$ will slow down neighboring processors due to neighbor-based throttling, encoded in the definition of w_2 in (3.12b)*. Moreover, the effect should become more global in x as η increases, since larger values of η correspond to a larger number of stages per processor. Indeed as the stages increase, interactions between neighbors begin to have a cumulative global effect. This trend can be observed by comparing results across the first three rows of fig. B.4 and in the line-outs in the final row.*

Example 3.5.3 (Variations in β) *In this example, we examine the effect of β on solutions to the macroscopic model, while holding $\eta = 1.0$ fixed. The initial condition, boundary condition, and processor speed are again given by* (3.49)*.*

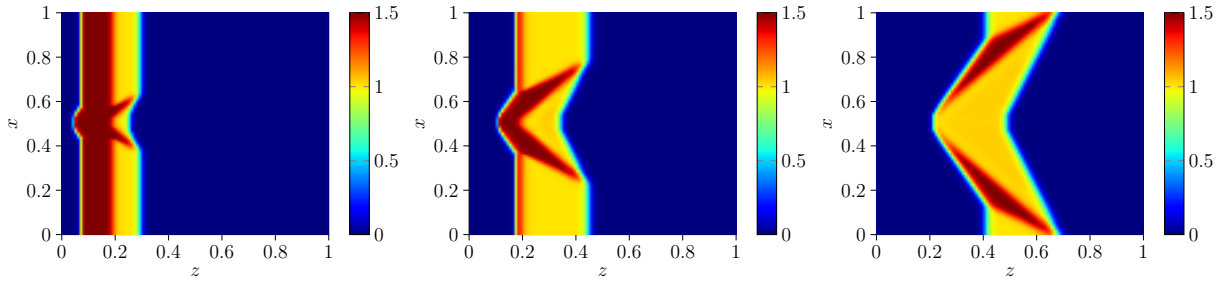
Based on the definition of the function w_2 in (3.12b)*, the expectation is that smaller values of β will lead to reduced throttling effects. Such behavior is confirmed by the numerical results in fig. B.5.*

Example 3.5.4 (Highly localized slowdown) *In this example, we explore the effects of a highly localized slowdown in processor speed when $\eta = \beta = 1$. The initial and boundary conditions are given in* (3.49)*, while the processor speed is given by $\alpha(x) = 1 - 0.4c(x)$, where*

$$c(x) = \begin{cases} 0 & |x - .5| > .05 \\ 40x - 18 & x \in [.45, .475] \\ -40x + 22 & x \in [.525, .55] \\ 1 & |x - .5| < .025 \end{cases}. \tag{3.50}$$

In particular, $\alpha \neq 1$ only on the interval $(0.45, 0.55)$. Simulation results from this example are shown in fig. 3.5. At early times, slower processors in the center of the x domain prohibit neighboring processors from moving data to later stages of the calculation (i.e. along the z-direction).

The result is a buildup of data in the neighboring processors. As time progresses, the build-up of data spreads as throttled processors near the initial slowdown around $x = 0.5$ begin to effect neighbors further away. Eventually these buildups dissipate as the slower processors begin catch up with their throttled neighbors.



(a) ρ at $t = 0.1$ (b) ρ at $t = 0.25$ (c) ρ at $t = 0.5$

Figure 3.5: The effect of a highly localized slowdown on ρ

Example 3.5.5 (Long-term behavior) *In previous examples, we have observed that under some conditions, solutions eventually resemble a traveling profile of the form*

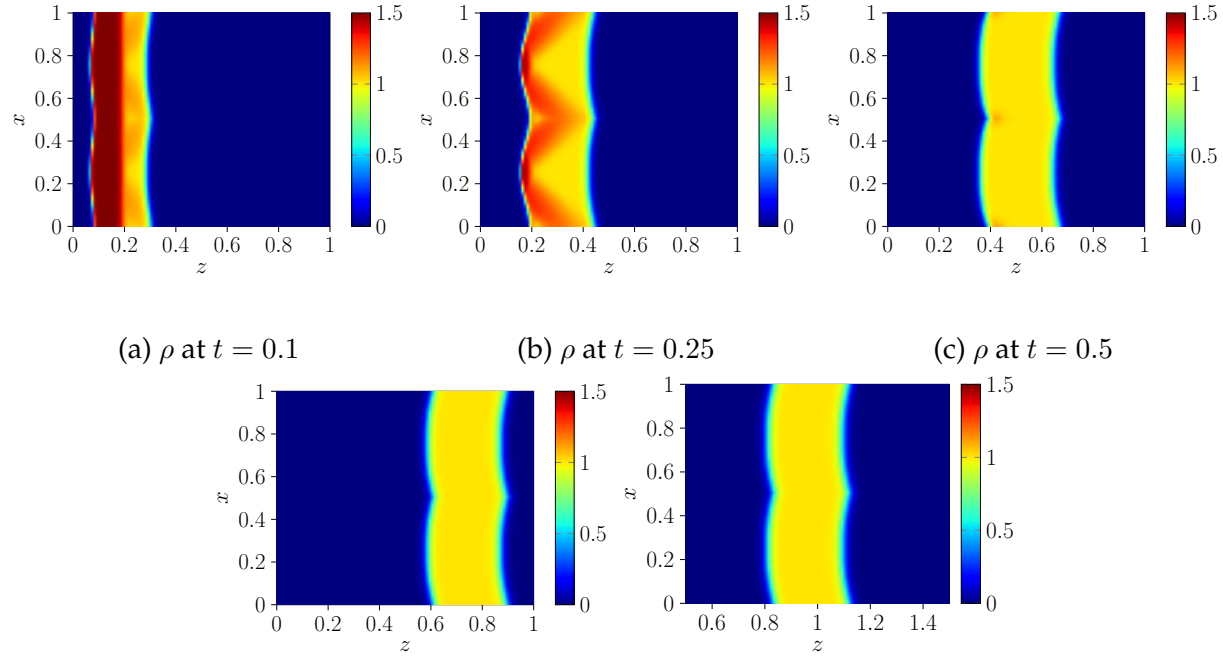
$$\rho_*(x, z, t) = \chi_{[\zeta_0(x), \zeta_1(x)]}(z - st), \tag{3.51}$$

where s is a positive constant and the profiles ζ_0 and ζ_1 are constant in time and satisfy $\zeta_0(x) < \zeta_1(x)$ for all $x \in [0, 1)$. Our intuition is that for a wide range of conditions, traveling profiles of this type will arise after sufficiently long times, if the z domain is extended to $(0, \infty)$. Moreover the shape of ζ_1 and ζ_2 is closely related to the initial data and the shape of α. A more systematic study of such profiles in special case can be found in [144]. Rather than make a precise conjecture at this point, we instead provide an example which further demonstrates our intuition. Initial and boundary conditions are given in (3.49)*. Because the domain in z is limited, we introduce relatively small variations in α, which allow the system to settle faster:*

$$\alpha(x) = 1 + 0.1 \cos(4\pi x). \tag{3.52}$$

Simulation results for this example are presented in fig. 3.6. When $t = 0.5$, the solution has nearly settled to a profile of the form (3.51)*, with cusps that appear where the waves caused by*

throttling meet, at $x = 0.5$ and at the periodic boundary. We see then at $t = 0.75$ that this profile is maintained, with cusps at the same location, and again at $t = 1$ (after extending the z domain). In particular the solution has the periodicity of α.



(a) ρ at $t = 0.1$ (b) ρ at $t = 0.25$ (c) ρ at $t = 0.5$

(d) ρ at $t = 0.75$ (e) ρ at $t = 1$

Figure 3.6: The effect of small variation in processor speed on ρ. After sufficient time, a profile emerges with the periodicity of α.

CHAPTER 4

CONCLUSIONS

In the thesis, we discuss several numerical methods, their applications, and implementations. For Sparse Grid Discontinuous Galerkin (SGDG) method, we focus on its adaptive version [3, 4, 5, 6, 7, 8], with Alpert's multiwavelet as basis for DG space, interpolatory multiwavelet and collocation method to treat nonlinear terms, fast wavelet transform dimension by dimension to reduce computational cost, and nonlinear solver in PETSc [9] for implicit time stepping. We apply this method on Maxwell equation in one or two dimension in nonlinear optical media [120, 121, 111, 122]. An energy stable Discontinuous Galerkin method [1, 2] is employed for this problem, and in our work, coupling with adaptive SGDG scheme package [8] using PETSc [9]. Numerical experiments of model problems show expected accuracy and convergence rate, and adaptive scheme with proper choice of error threshold can choose a set of active elements in hash table to represent numerical solutions, without significant degeneration of numerical accuracy or generating non-physical numerical solutions or oscillations. In future work, data structure of parallel computation in PETSc, e.g. DMPlex to deal with unstructured mesh, and IS and PetscSection for indexing, can provide a solution to compute numerical solution of SGDG method in core parallelism. The usage of PETSc also provides a solution to SGDG method with fully implicit time stepping, broadening the availability of SGDG method. We can also explore different standard for refinement and coarsening step of adaptive method, to better specify active elements in adaptive scheme.

For Weighted Essentially Non-Oscillatory (WENO) method, we consider the fifth order WENO interpolation, which maintains high order accuracy on smooth region, while any singularity does not significantly affect the numerical interpolation [107, 12]. WENO interpolation and strong stability preserving (SSP) time stepping provide promising fully discrete numerical method to solve Hamilton-Jacobi equation [11]. This method is ap-

plied to solve the Hamilton-Jacobi continuum model of extreme scale parallel computers, which is derived from a discrete model by taking limits on number of processors and tasks. Numerical experiments of both models show that this continuum model can capture the asymptotic behaviour of the discrete model. Additionally, these experiments provide an initial understanding of solutions' dependence on parameters associated with the parallelism of the modelled computation as well as the effects heterogeneities in processing capacity [10]. In future work, we can explore control strategies for α that can alleviate bottlenecks caused by local slowdowns in the processor speed. We can also extend the model to allow for more complicated interactions, including stochastic effects, and explore strategies for optimal communication. Finally, the parameters of the model can be taken from processor components of a real supercomputer, and we can determine if the macroscopic model predicts the real global behaviour of the supercomputer.

APPENDICES

APPENDIX A

NUMERICAL SIMULATION OF 1D SOLITON PROPAGATION

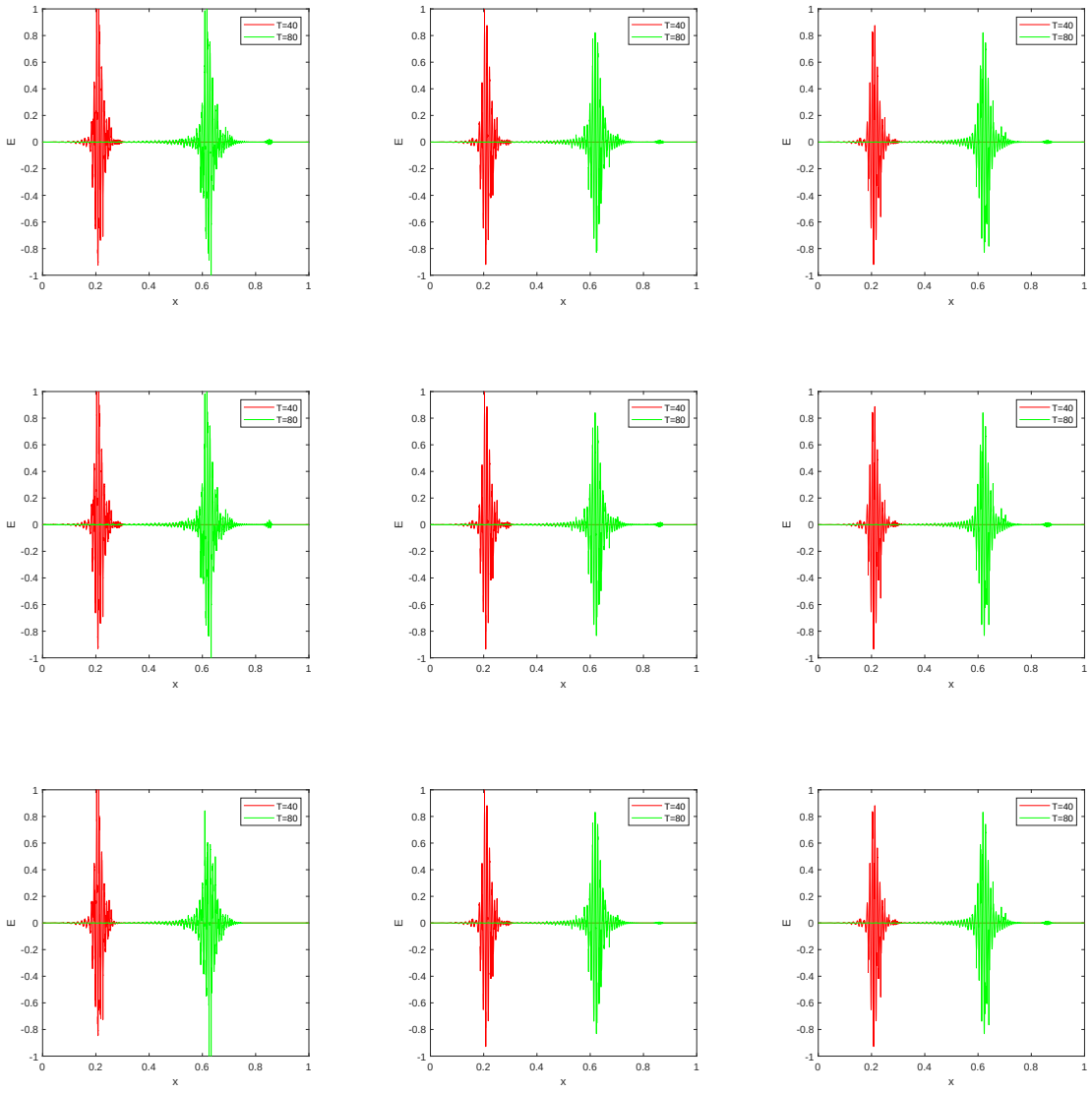
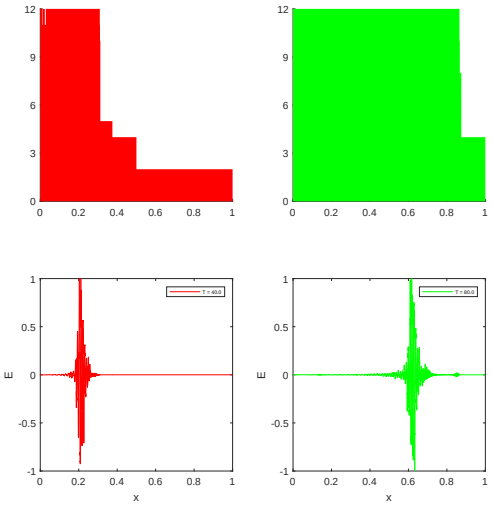
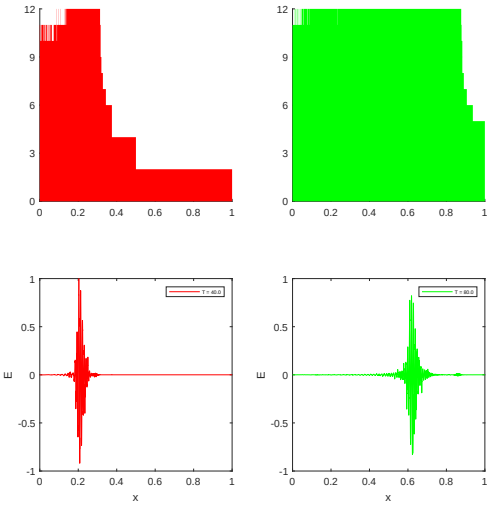


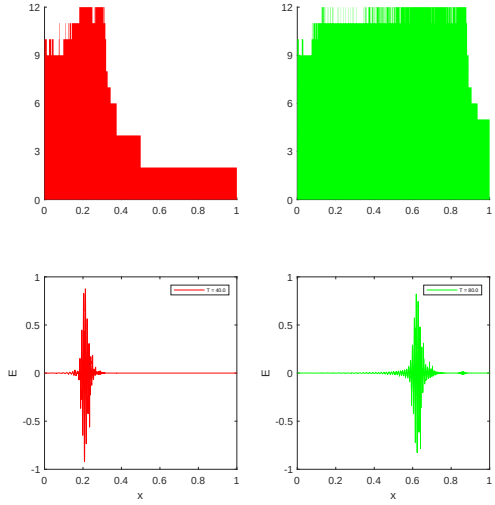
Figure A.1: Transient fundamental ($M = 1$) temporal soliton propagation with the adaptive Sparse Grid DG fully implicit scheme, in Subsection 2.5.2. Maximum mesh level $N = 11$. Refinement threshold $\varepsilon = 10^{-5}$, and coarsen threshold is $\eta = 10^{-6}$. First column: piecewise polynomial degree $k = 1$; second column: $k = 2$; third column: $k = 3$. First row: alternating flux I; second row: alternating flux II; third row: upwind flux. Solutions at time $T = 40$ and $T = 80$ are plotted on each case.



(a) $k = 1$, Alternating flux I

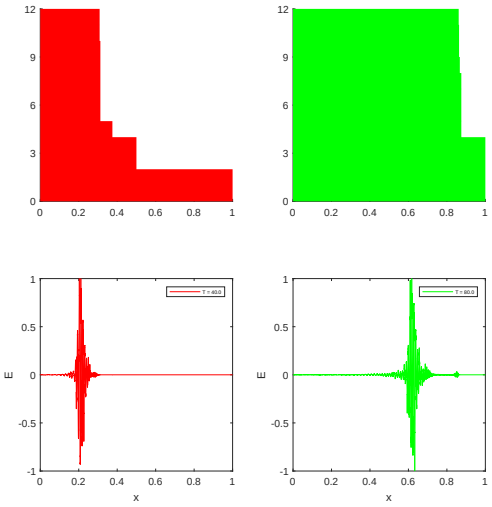


(b) $k = 2$, Alternating flux I

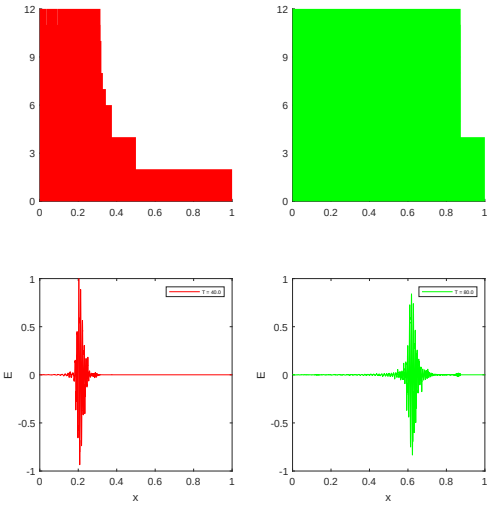


(c) $k = 3$, Alternating flux I

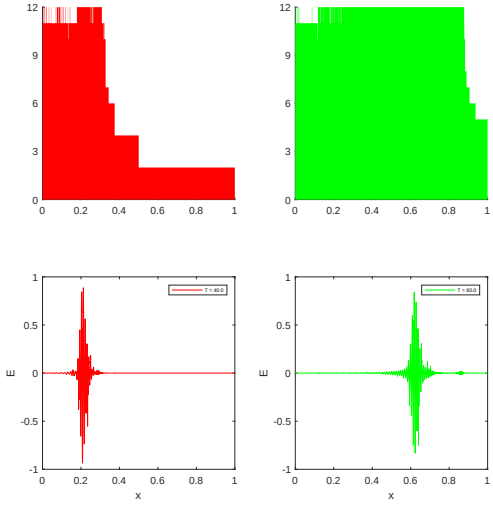
Figure A.2: Comparison of active elements and electric field solutions, at $T = 40$ and $T = 80$, for $k = 1, 2, 3$, of alternating flux I case, as in Figure A.1.



(a) $k = 1$, Alternating flux II

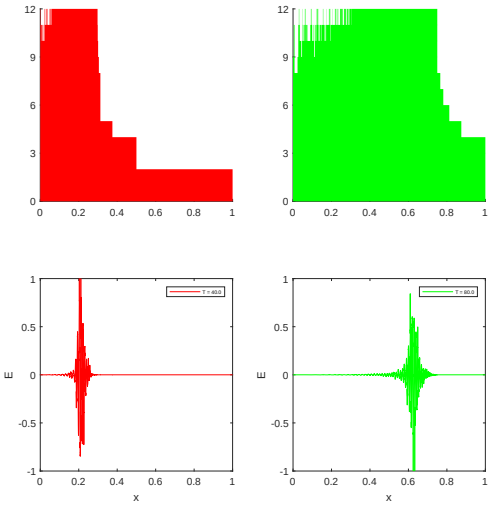


(b) $k = 2$, Alternating flux II

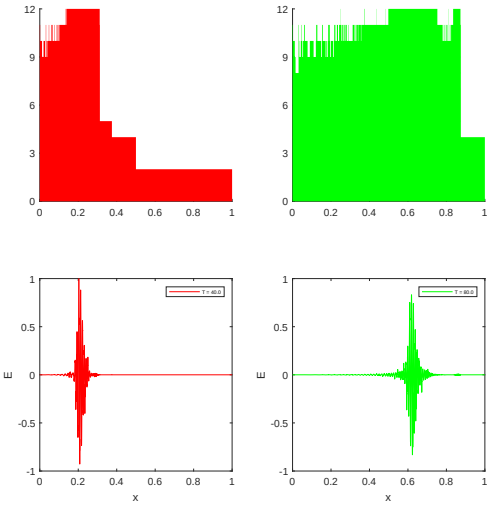


(c) $k = 3$, Alternating flux II

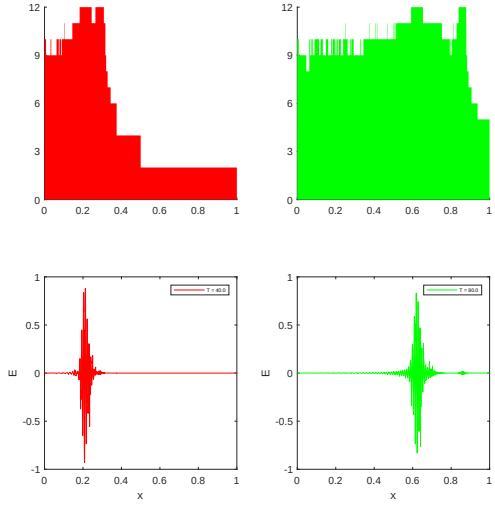
Figure A.3: Comparison of active elements and electric field solutions, at $T = 40$ and $T = 80$, for $k = 1, 2, 3$, of alternating flux II case, as in Figure A.1.



(a) $k = 1$, Upwind flux



(b) $k = 2$, Upwind flux



(c) $k = 3$, Upwind flux

Figure A.4: Comparison of active elements and electric field solutions, at $T = 40$ and $T = 80$, for $k = 1, 2, 3$, of upwind flux case, as in Figure A.1.

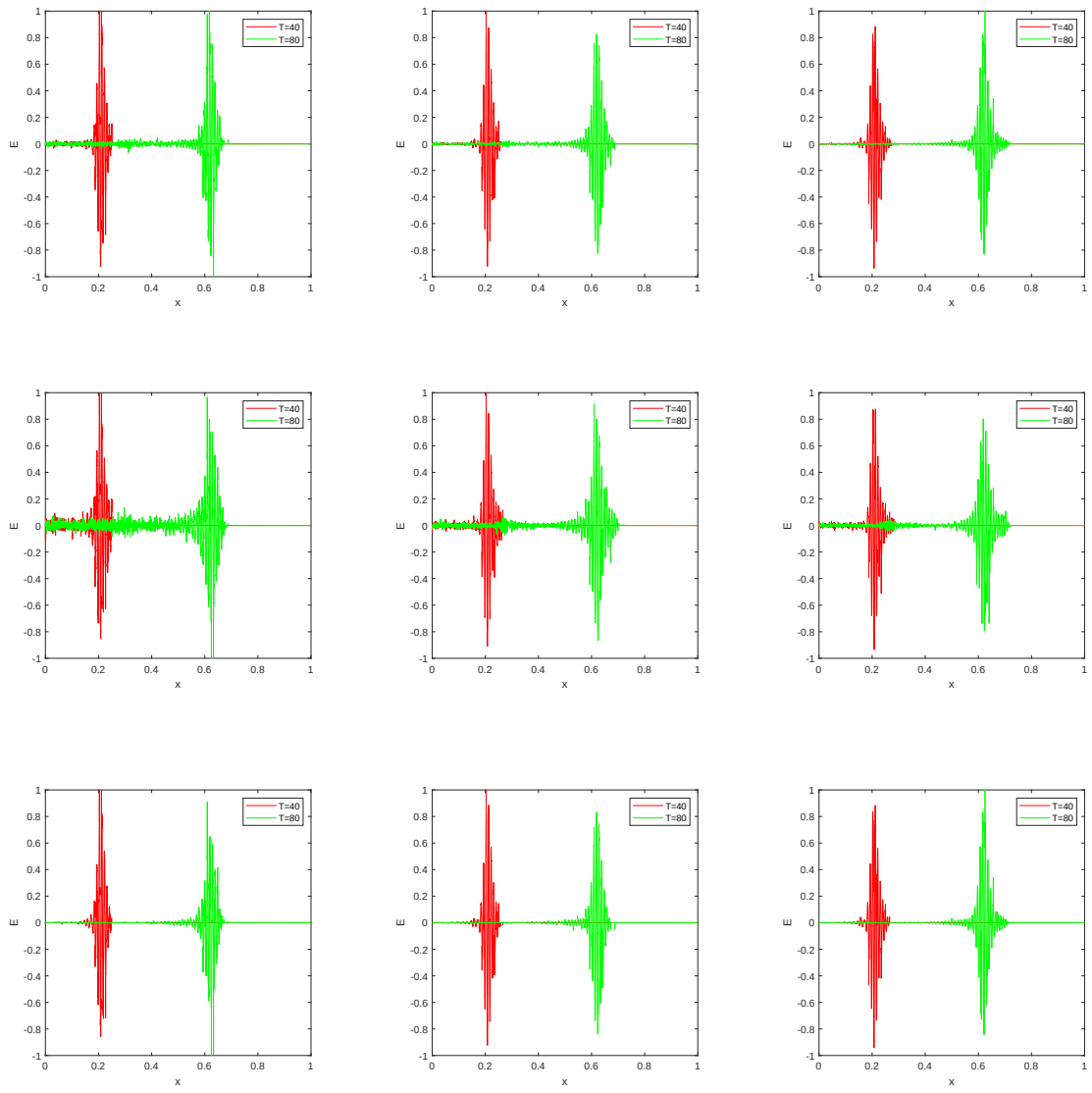
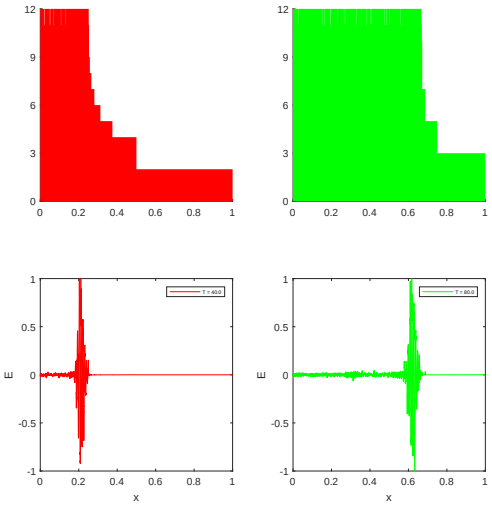
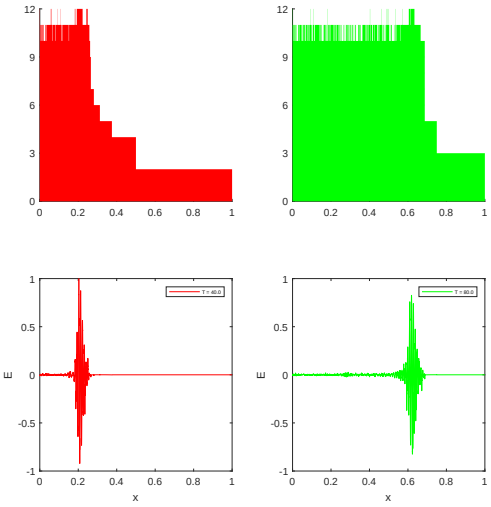


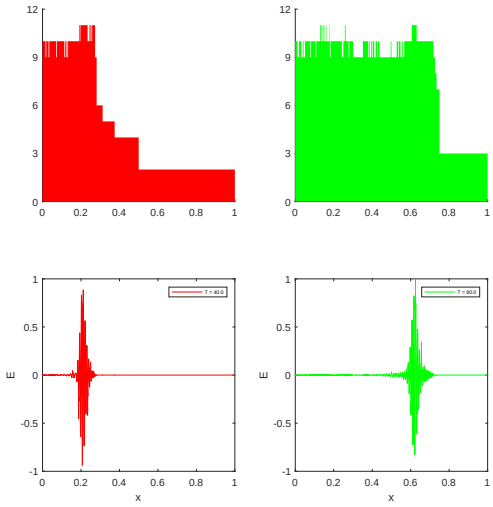
Figure A.5: Transient fundamental ($M = 1$) temporal soliton propagation with the adaptive Sparse Grid DG fully implicit scheme, in Subsection 2.5.2. Maximum mesh level $N = 11$. Refinement threshold $\varepsilon = 10^{-3}$, and coarsen threshold is $\eta = 10^{-4}$. First column: piecewise polynomial degree $k = 1$; second column: $k = 2$; third column: $k = 3$. First row: alternating flux I; second row: alternating flux II; third row: upwind flux. Solutions at time $T = 40$ and $T = 80$ are plotted on each case.



(a) $k = 1$, Alternating flux I

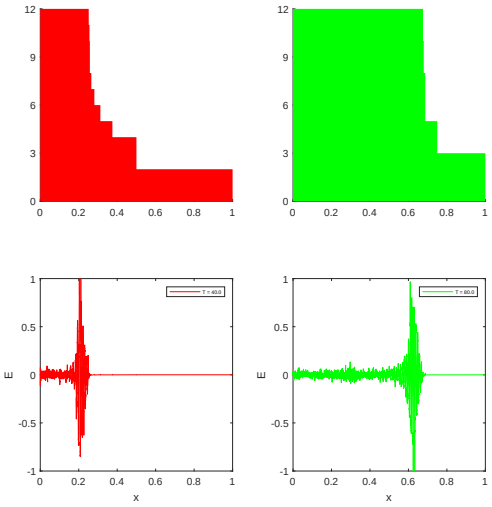


(b) $k = 2$, Alternating flux I

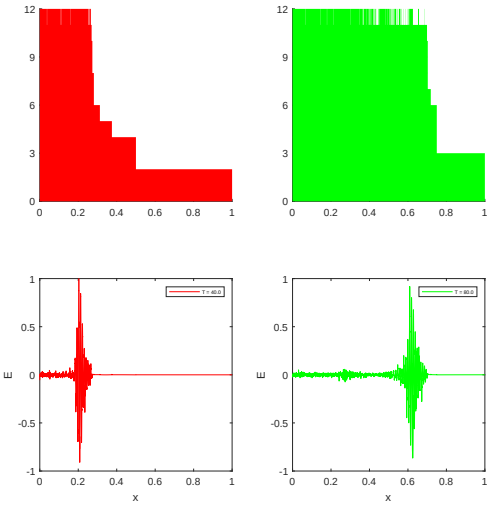


(c) $k = 3$, Alternating flux I

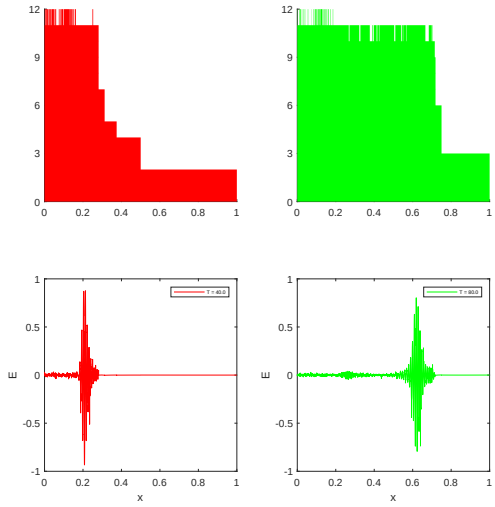
Figure A.6: Comparison of active elements and electric field solutions, at $T = 40$ and $T = 80$, for $k = 1, 2, 3$, of alternating flux I case, as in Figure A.5.



(a) $k = 1$, Alternating flux II

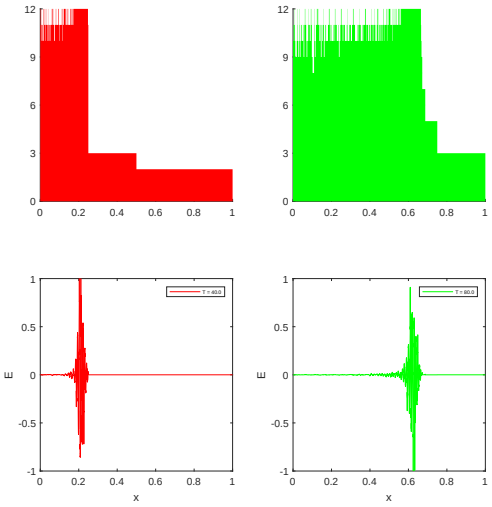


(b) $k = 2$, Alternating flux II

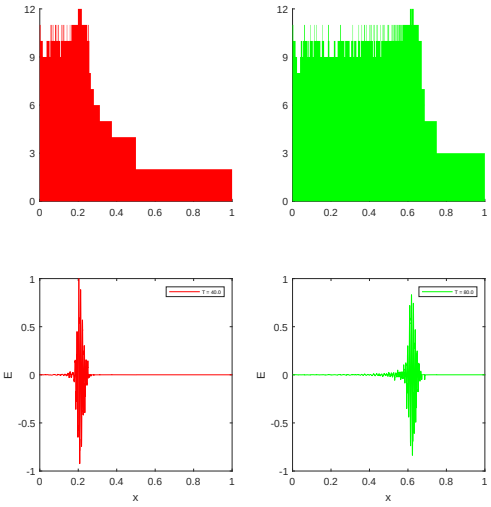


(c) $k = 3$, Alternating flux II

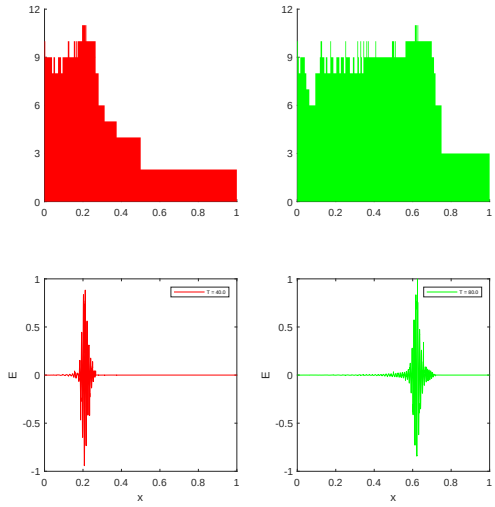
Figure A.7: Comparison of active elements and electric field solutions, at $T = 40$ and $T = 80$, for $k = 1, 2, 3$, of alternating flux II case, as in Figure A.5.



(a) $k = 1$, Upwind flux



(b) $k = 2$, Upwind flux



(c) $k = 3$, Upwind flux

Figure A.8: Comparison of active elements and electric field solutions, at $T = 40$ and $T = 80$, for $k = 1, 2, 3$, of upwind flux case, as in Figure A.5.

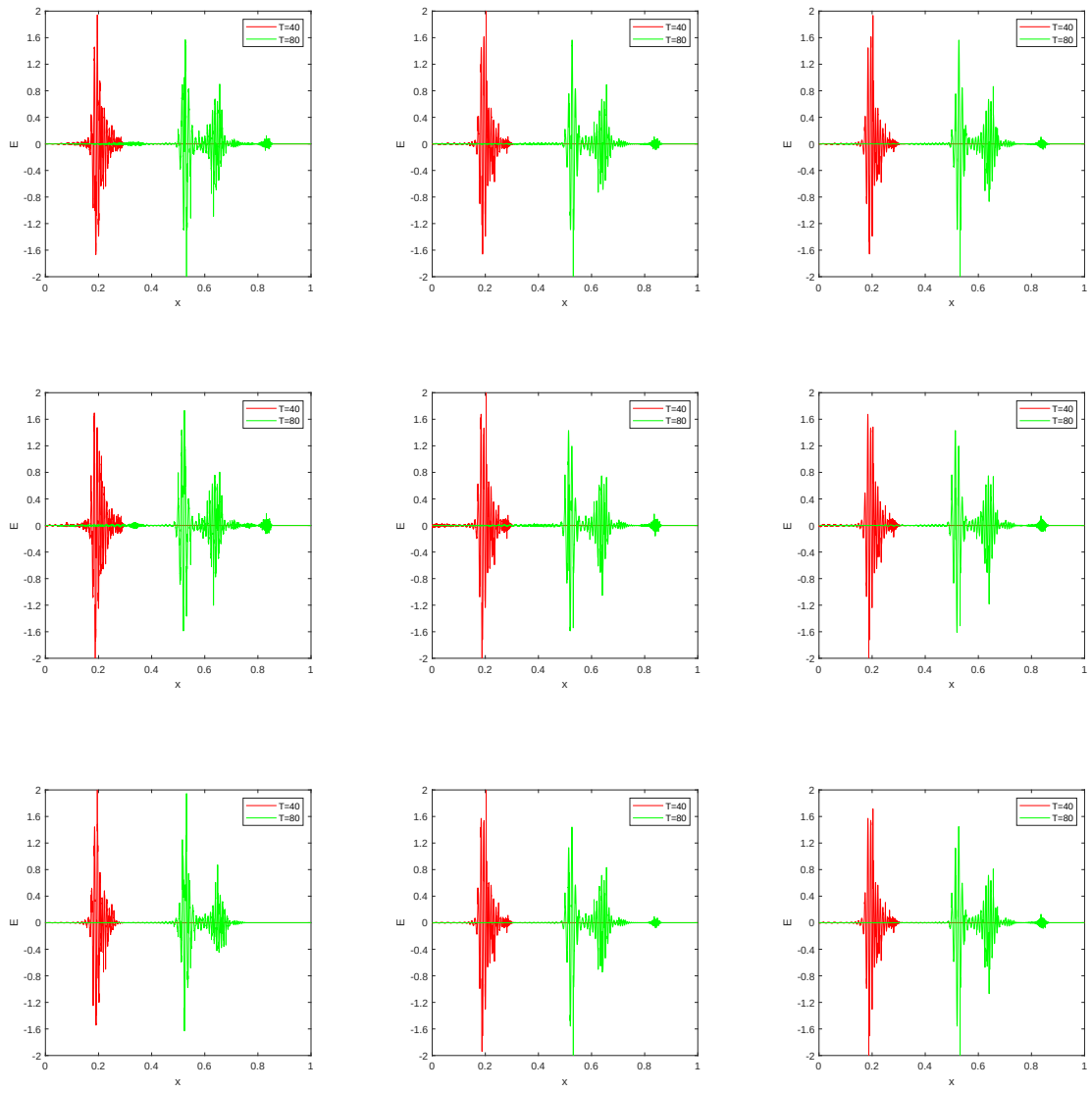
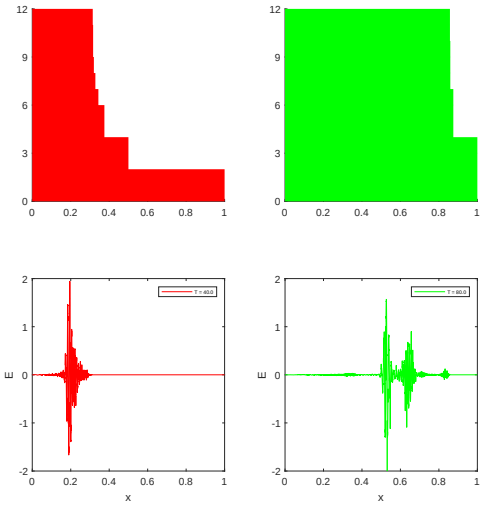
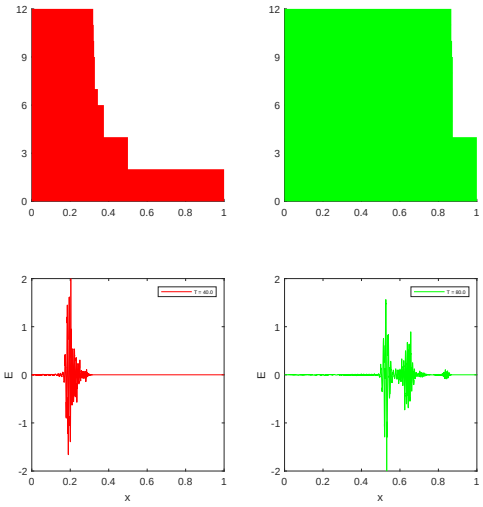


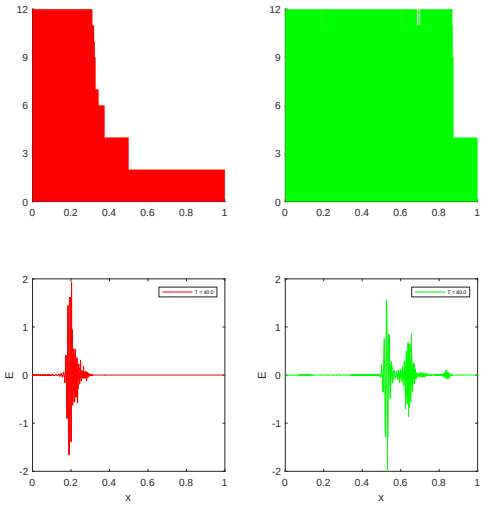
Figure A.9: Transient second order ($M = 2$) temporal soliton propagation with the adaptive Sparse Grid DG fully implicit scheme, in Subsection 2.5.2. Maximum mesh level $N = 11$. Refinement threshold $\varepsilon = 10^{-5}$, and coarsen threshold is $\eta = 10^{-6}$. First column: piecewise polynomial degree $k = 1$; second column: $k = 2$; third column: $k = 3$. First row: alternating flux I; second row: alternating flux II; third row: upwind flux. Solutions at time $T = 40$ and $T = 80$ are plotted on each case.



(a) $k = 1$, Alternating flux I

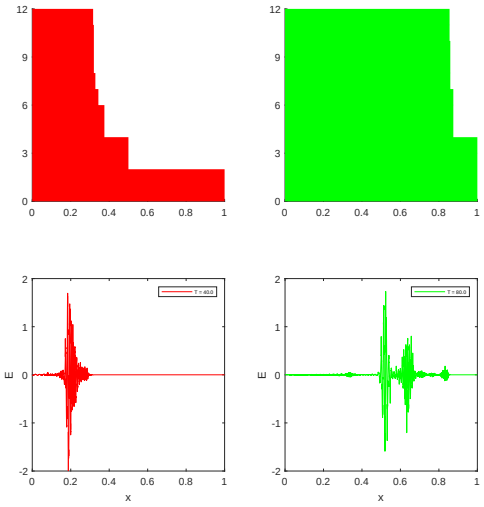


(b) $k = 2$, Alternating flux I

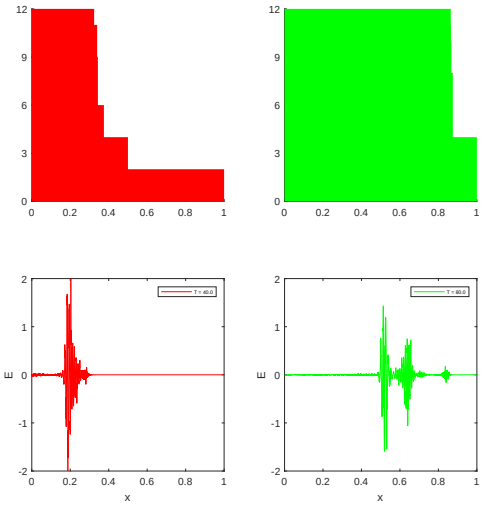


(c) $k = 3$, Alternating flux I

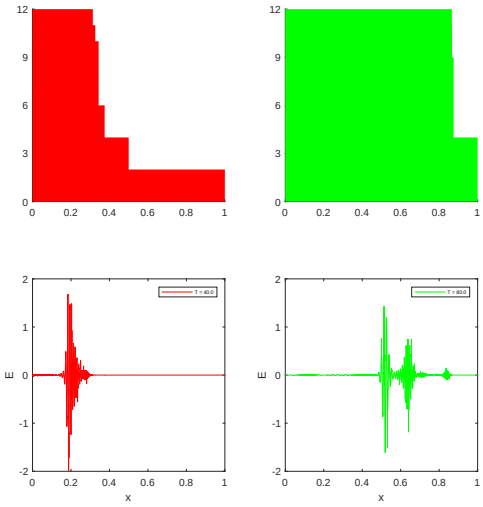
Figure A.10: Comparison of active elements and electric field solutions, at $T = 40$ and $T = 80$, for $k = 1, 2, 3$, of alternating flux I case, as in Figure A.9.



(a) $k = 1$, Alternating flux II

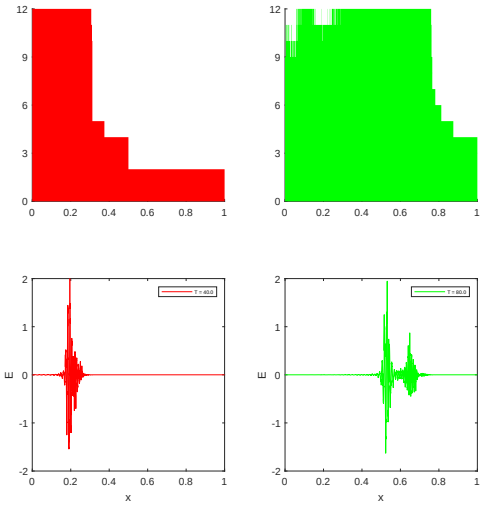


(b) $k = 2$, Alternating flux II

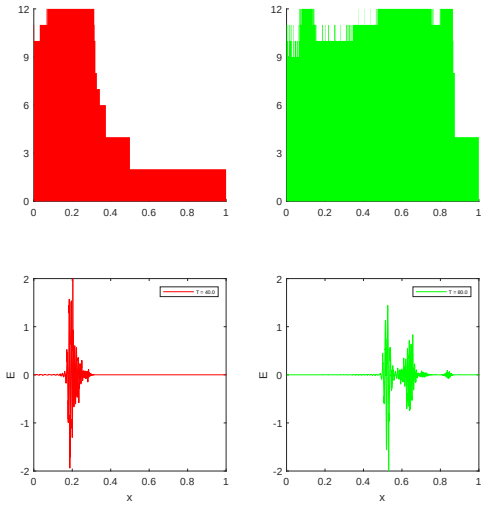


(c) $k = 3$, Alternating flux II

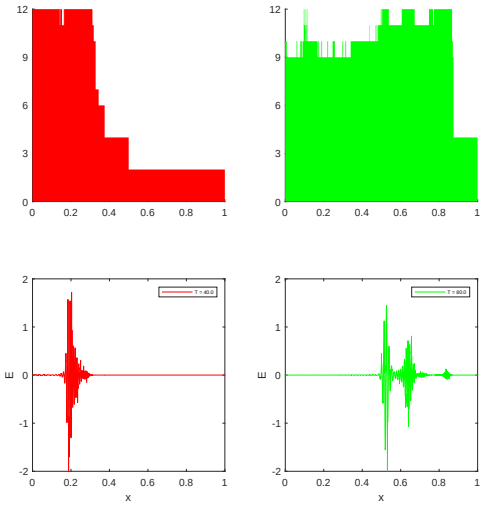
Figure A.11: Comparison of active elements and electric field solutions, at $T = 40$ and $T = 80$, for $k = 1, 2, 3$, of alternating flux II case, as in Figure A.9.



(a) $k = 1$, Upwind flux



(b) $k = 2$, Upwind flux

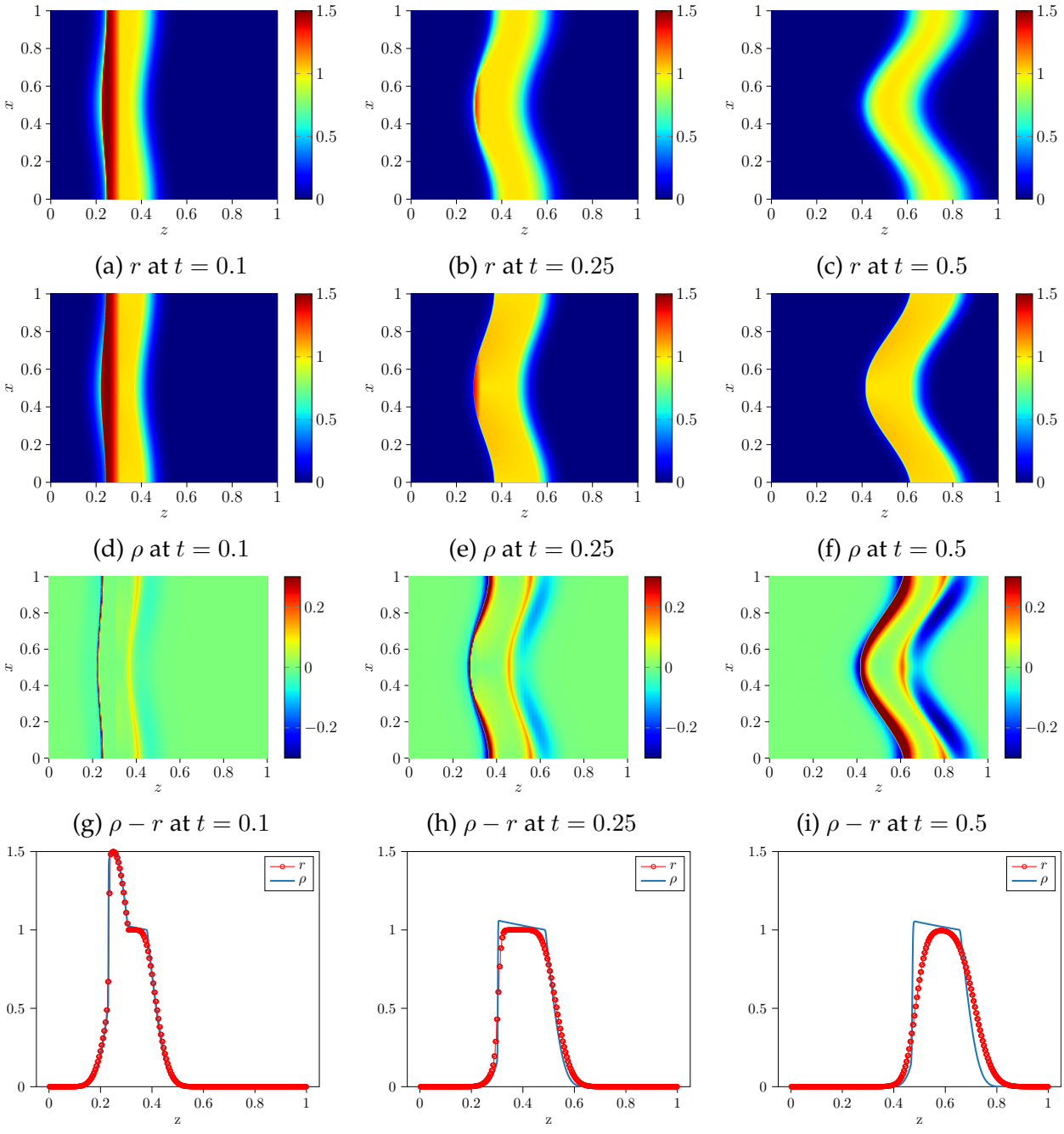


(c) $k = 3$, Upwind flux

Figure A.12: Comparison of active elements and electric field solutions, at $T = 40$ and $T = 80$, for $k = 1, 2, 3$, of upwind flux case, as in Figure A.9.

APPENDIX B

NUMERICAL EXPERIMENTS OF ASYNCHRONOUS DATA FLOW MODELS



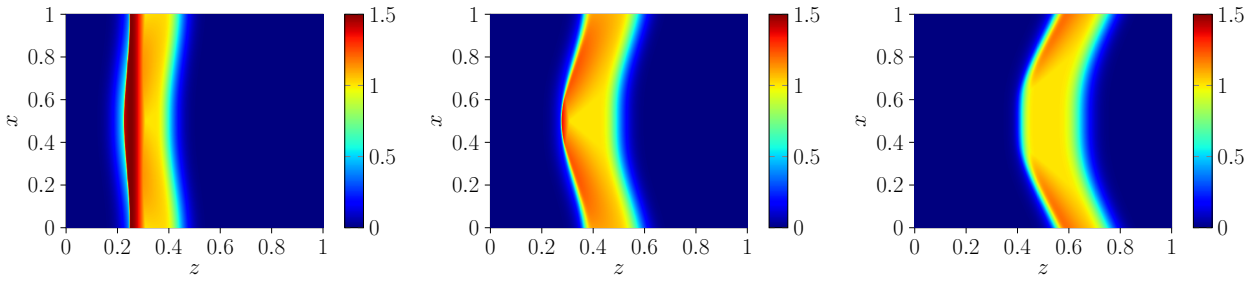
(a) r at $t = 0.1$ (b) r at $t = 0.25$ (c) r at $t = 0.5$

(d) ρ at $t = 0.1$ (e) ρ at $t = 0.25$ (f) ρ at $t = 0.5$

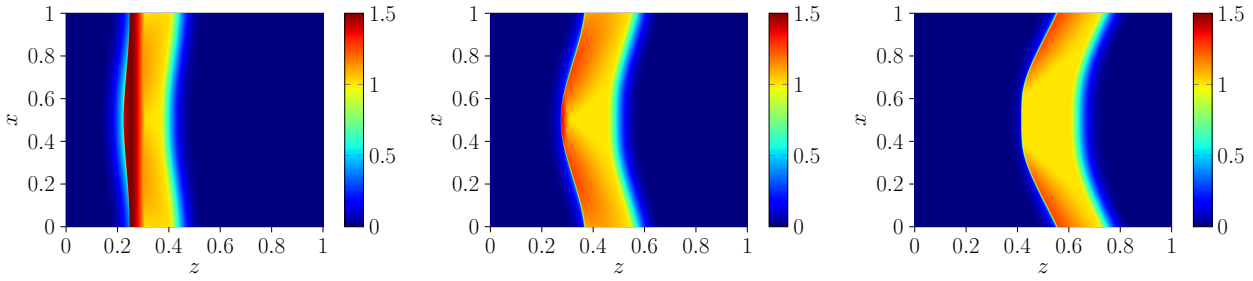
(g) $\rho - r$ at $t = 0.1$ (h) $\rho - r$ at $t = 0.25$ (i) $\rho - r$ at $t = 0.5$

(j) ρ,r at $(x, t) = (0.3, 0.1)$ (k) ρ,r at $(x, t) = (0.3, 0.25)$ (l) ρ,r at $(x, t) = (0.3, 0.5)$

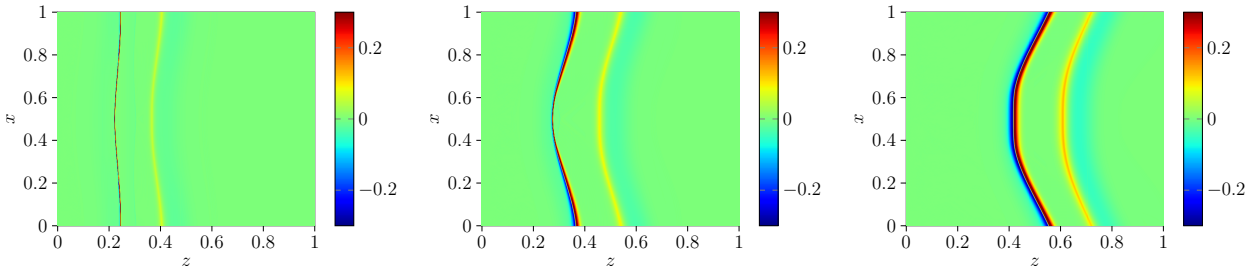
Figure B.1: Discrete solution r and continuum solution ρ of $\eta = 0.2$, in Example 3.5.1. From left to right, columns correspond to solutions at $t = 0.1$, $t = 0.25$, and $t = 0.5$. Discrete solution is computed with $(i^{\max}, k^{\max}) = (1000, 200)$. Continuum solution is computed on a $10^3 \times 10^3$ mesh.



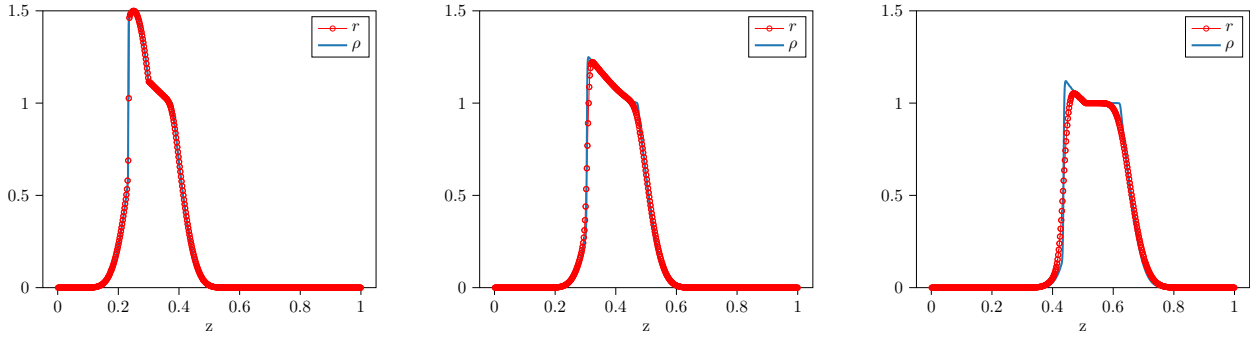
(a) r at $t = 0.1$ (b) r at $t = 0.25$ (c) r at $t = 0.5$



(d) ρ at $t = 0.1$ (e) ρ at $t = 0.25$ (f) ρ at $t = 0.5$

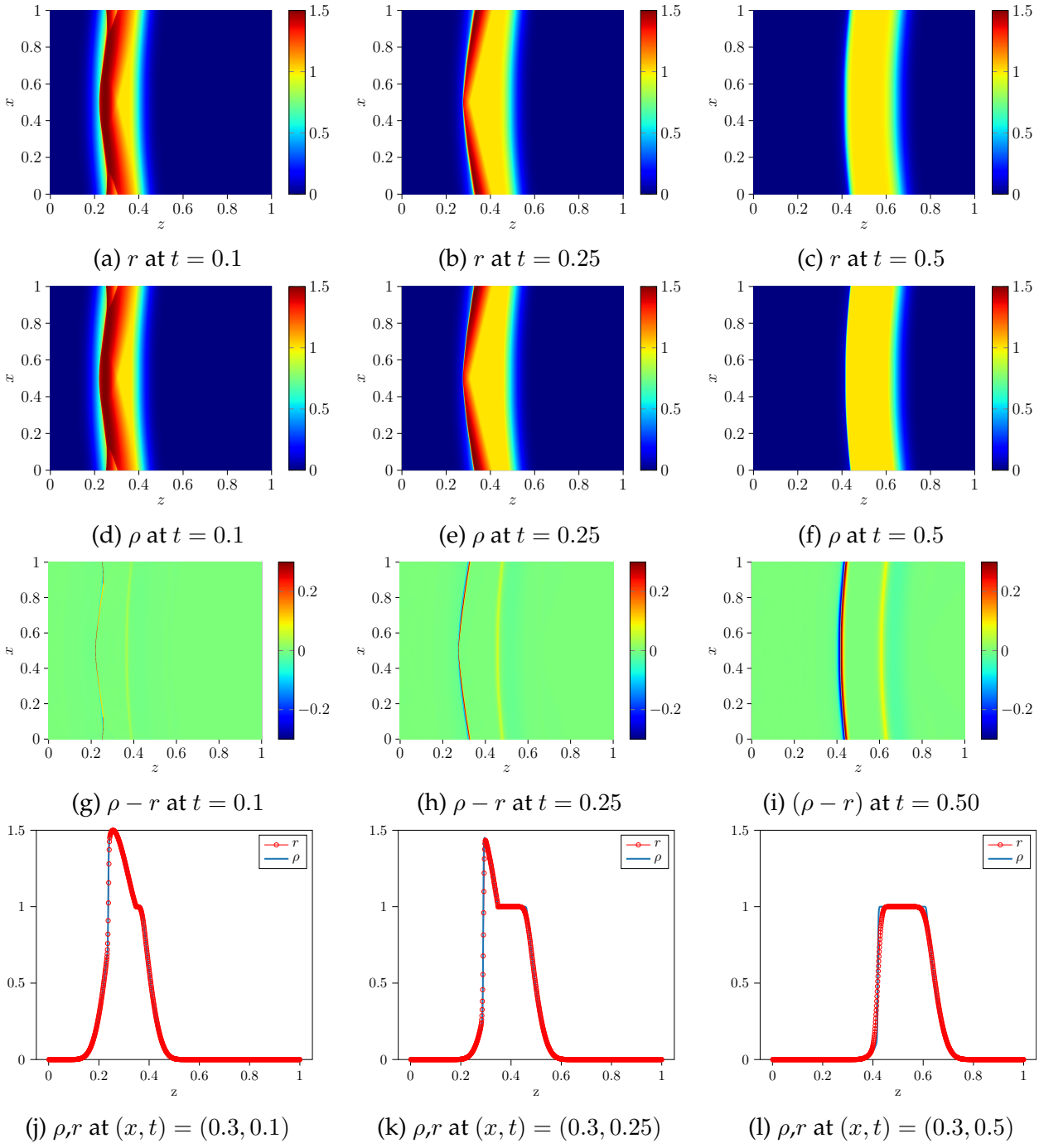


(g) $\rho - r$ at $t = 0.1$ (h) $\rho - r$ at $t = 0.25$ (i) $\rho - r$ at $t = 0.5$



(j) ρ,r at $(x, t) = (0.3, 0.1)$ (k) ρ,r at $(x, t) = (0.3, 0.25)$ (l) ρ,r at $(x, t) = (0.3, 0.5)$

Figure B.2: Discrete solution r and continuum solution ρ of $\eta = 1$ case, in Example 3.5.1. From left to right, columns correspond to solutions at $t = 0.1$, $t = 0.25$, and $t = 0.5$. Discrete solution is computed with $(i^{\max}, k^{\max}) = (500, 500)$. Continuum solution is computed on a $10^3 \times 10^3$ mesh.



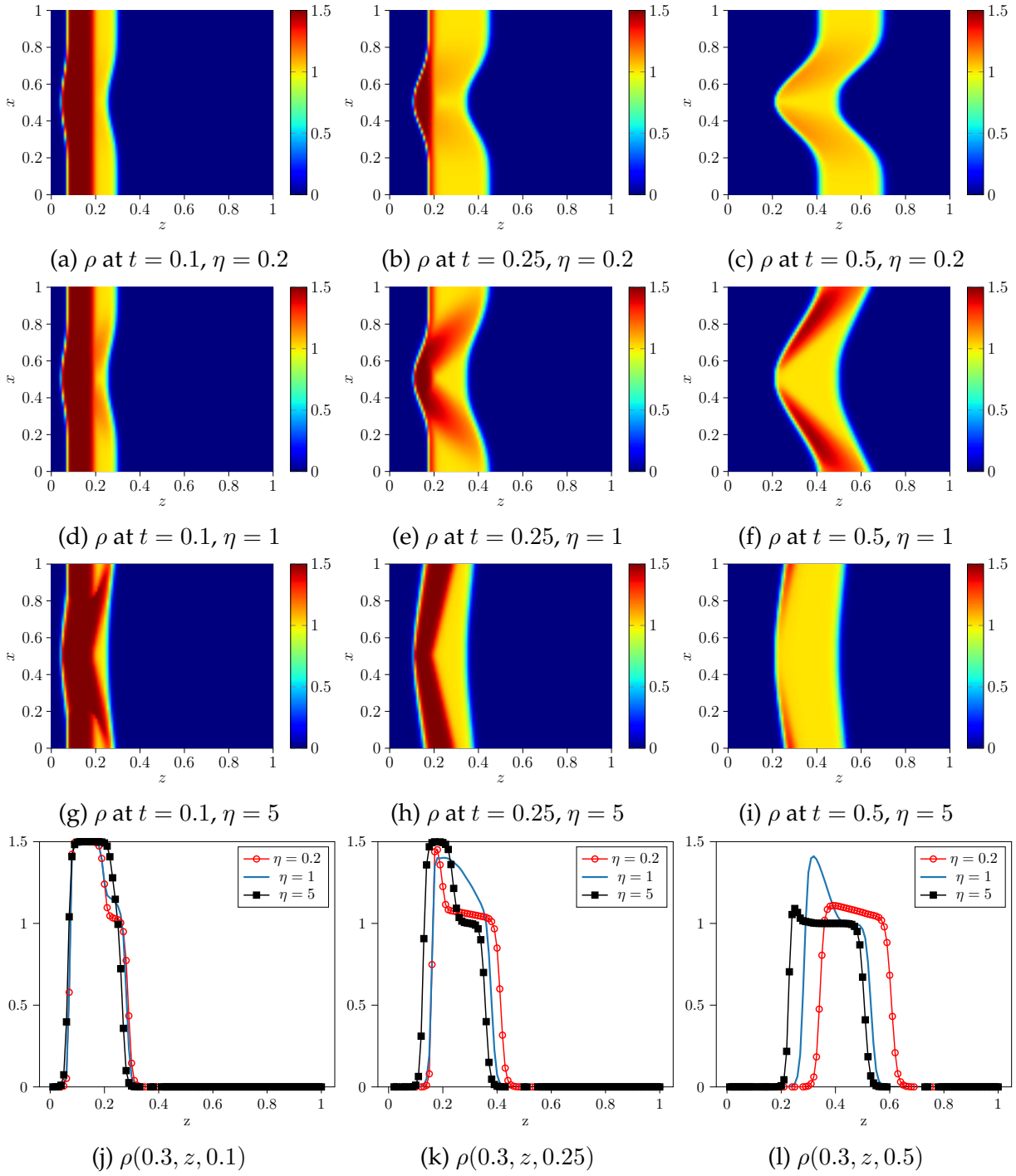
(a) r at $t = 0.1$ (b) r at $t = 0.25$ (c) r at $t = 0.5$

(d) ρ at $t = 0.1$ (e) ρ at $t = 0.25$ (f) ρ at $t = 0.5$

(g) $\rho - r$ at $t = 0.1$ (h) $\rho - r$ at $t = 0.25$ (i) $(\rho - r)$ at $t = 0.50$

(j) ρ,r at $(x, t) = (0.3, 0.1)$ (k) ρ,r at $(x, t) = (0.3, 0.25)$ (l) ρ,r at $(x, t) = (0.3, 0.5)$

Figure B.3: Discrete solution r and continuum solution ρ of $\eta = 5$ case, in Example 3.5.1. From left to right, columns correspond to solutions at $t = 0.1$, $t = 0.25$, and $t = 0.5$. Discrete solution is computed with $(i^{\max}, k^{\max}) = (200, 1000)$. Continuum solution is computed on a $10^3 \times 10^3$ mesh.



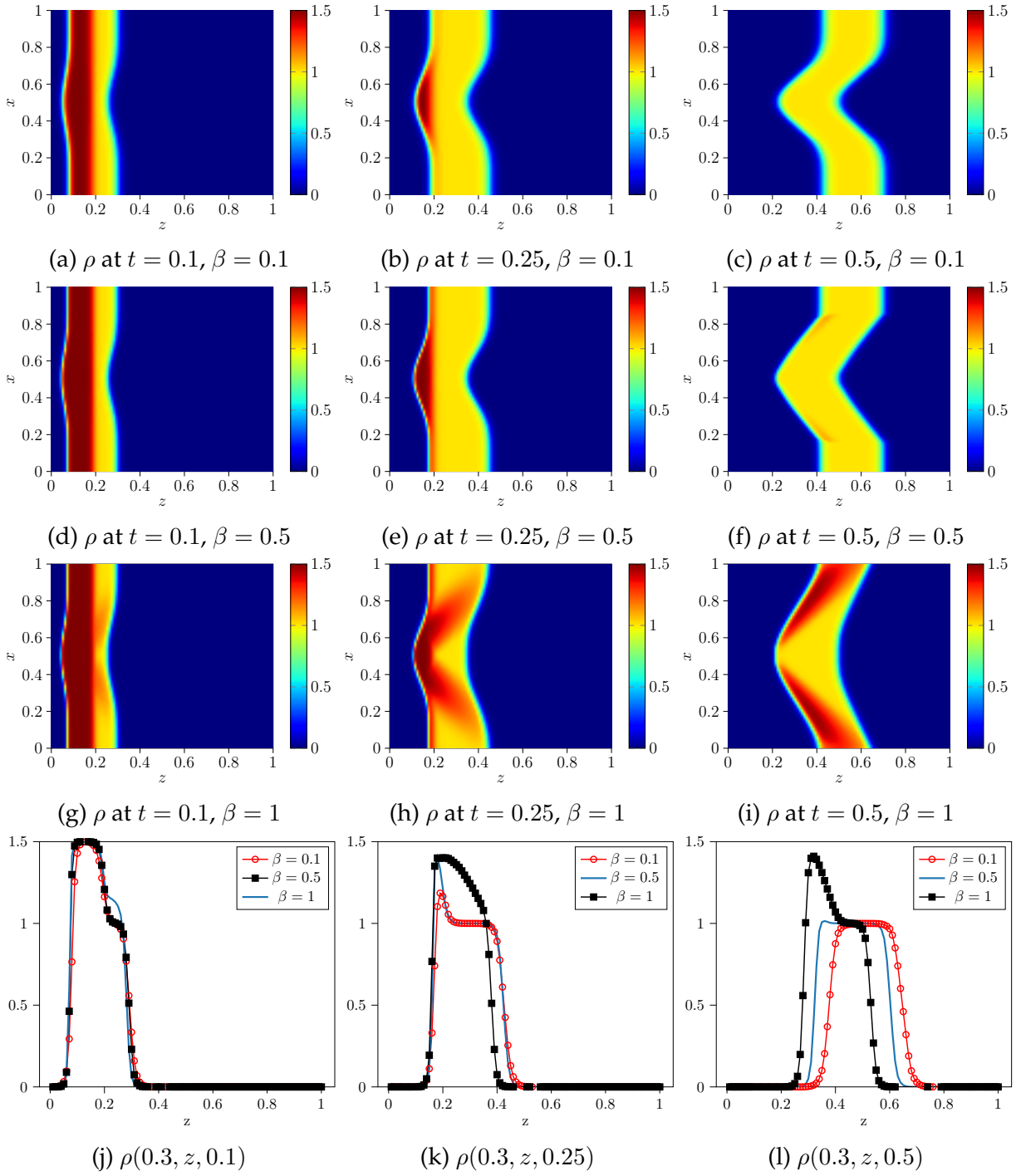
(a) ρ at $t = 0.1$, $\eta = 0.2$ (b) ρ at $t = 0.25$, $\eta = 0.2$ (c) ρ at $t = 0.5$, $\eta = 0.2$

(d) ρ at $t = 0.1$, $\eta = 1$ (e) ρ at $t = 0.25$, $\eta = 1$ (f) ρ at $t = 0.5$, $\eta = 1$

(g) ρ at $t = 0.1$, $\eta = 5$ (h) ρ at $t = 0.25$, $\eta = 5$ (i) ρ at $t = 0.5$, $\eta = 5$

(j) $\rho(0.3, z, 0.1)$ (k) $\rho(0.3, z, 0.25)$ (l) $\rho(0.3, z, 0.5)$

Figure B.4: The effects on ρ due to variations in η, in Example 3.5.2. As η increases, the throttling effect of local slowdown spreads more quickly, and data is not processed as quickly.



(a) ρ at $t = 0.1$, $\beta = 0.1$ (b) ρ at $t = 0.25$, $\beta = 0.1$ (c) ρ at $t = 0.5$, $\beta = 0.1$

(d) ρ at $t = 0.1$, $\beta = 0.5$ (e) ρ at $t = 0.25$, $\beta = 0.5$ (f) ρ at $t = 0.5$, $\beta = 0.5$

(g) ρ at $t = 0.1$, $\beta = 1$ (h) ρ at $t = 0.25$, $\beta = 1$ (i) ρ at $t = 0.5$, $\beta = 1$

(j) $\rho(0.3, z, 0.1)$ (k) $\rho(0.3, z, 0.25)$ (l) $\rho(0.3, z, 0.5)$

Figure B.5: The effects on ρ due to variations in β, in Example 3.5.3. Larger values of β lead to more throttling.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Vrushali A Bokil, Yingda Cheng, Yan Jiang, and Fengyan Li. Energy stable discontinuous Galerkin methods for Maxwell's equations in nonlinear optical media. *Journal of Computational Physics*, 350:420–452, 2017.

[2] Maohui Lyu, Vrushali A Bokil, Yingda Cheng, and Fengyan Li. Energy stable nodal discontinuous Galerkin methods for nonlinear Maxwell's equations in multidimensions. *Journal of Scientific Computing*, 89(2):1–42, 2021.

[3] Zixuan Wang, Qi Tang, Wei Guo, and Yingda Cheng. Sparse grid discontinuous Galerkin methods for high-dimensional elliptic equations. *Journal of Computational Physics*, 314:244–263, 2016.

[4] Wei Guo and Yingda Cheng. A sparse grid discontinuous Galerkin method for high-dimensional transport equations and its application to kinetic simulations. *SIAM Journal on Scientific Computing*, 38(6):A3381–A3409, 2016.

[5] Wei Guo and Yingda Cheng. An adaptive multiresolution discontinuous Galerkin method for time-dependent transport equations in multidimensions. *SIAM Journal on Scientific Computing*, 39(6):A2962–A2992, 2017.

[6] Juntao Huang and Yingda Cheng. An adaptive multiresolution discontinuous Galerkin method with artificial viscosity for scalar hyperbolic conservation laws in multidimensions. *SIAM Journal on Scientific Computing*, 42(5):A2943–A2973, 2020.

[7] Zhanjing Tao, Yan Jiang, and Yingda Cheng. An adaptive high-order piecewise polynomial based sparse grid collocation method with applications. *Journal of Computational Physics*, 433:109770, 2021.

[8] Yingda Cheng, Wei Guo, Juntao Huang, Kai Huang, Yuan Liu, and Zhanjing Tao. Adaptive multiresolution discontinuous Galerkin (DG) C++ package for solving partial differential equations in high dimensions. https://github.com/JuntaoHuang/adaptive-multiresolution-DG, 2019-2022.

[9] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc Web page. https://petsc.org/, 2022.

[10] Richard C Barnard, Kai Huang, and Cory Hauck. A mathematical model of asynchronous data flow in parallel computers. *IMA Journal of Applied Mathematics*, 85(6):865–891, 2020.

[11] Chi-Wang Shu. High order numerical methods for time dependent Hamilton-Jacobi equations. In *Mathematics and computation in imaging science and information processing*, pages 47–91. World Scientific, 2007.

[12] Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. *Advanced numerical approximation of nonlinear hyperbolic equations*, pages 325–432, 1998.

[13] William H Reed and Thomas R Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[14] Bernardo Cockburn and Chi-Wang Shu. The Runge-Kutta local projection-discontinuous-Galerkin finite element method for scalar conservation laws. *ESAIM: Mathematical Modelling and Numerical Analysis*, 25(3):337–361, 1991.

[15] Bernardo Cockburn and Chi-Wang Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. general framework. *Mathematics of computation*, 52(186):411–435, 1989.

[16] Bernardo Cockburn, San-Yih Lin, and Chi-Wang Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems. *Journal of computational Physics*, 84(1):90–113, 1989.

[17] Bernardo Cockburn, Suchung Hou, and Chi-Wang Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV. the multidimensional case. *Mathematics of Computation*, 54(190):545–581, 1990.

[18] Bernardo Cockburn and Chi-Wang Shu. The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems. *Journal of Computational Physics*, 141(2):199–224, 1998.

[19] Bernardo Cockburn and Chi-Wang Shu. *The P^1-RKDG method for two-dimensional Euler equations of gas dynamics*, volume 91-9. NASA Langley Research Center. Institute for Computer Applications in Science and Engineering (ICASE), 1991.

[20] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of computational physics*, 77(2):439–471, 1988.

[21] Sigal Gottlieb, David I Ketcheson, and Chi-Wang Shu. *Strong stability preserving Runge-Kutta and multistep time discretizations*. World Scientific, 2011.

[22] Chi-Wang Shu. TVB uniformly high-order schemes for conservation laws. *Mathematics of Computation*, 49(179):105–121, 1987.

[23] Jianxian Qiu and Qiang Zhang. Stability, error estimate and limiters of discontinuous Galerkin methods. In *Handbook of Numerical Analysis*, volume 17, pages 147–171. Elsevier, 2016.

[24] Francesco Bassi and Stefano Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. *Journal of computational physics*, 131(2):267–279, 1997.

[25] Bernardo Cockburn and Chi-Wang Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM journal on numerical analysis*, 35(6):2440–2463, 1998.

[26] Bernardo Cockburn, George E Karniadakis, and Chi-Wang Shu. The development of discontinuous Galerkin methods. In *Discontinuous Galerkin Methods*, pages 3–50. Springer, 2000.

[27] Chi-Wang Shu. Discontinuous Galerkin methods: general approach and stability. *Numerical solutions of partial differential equations*, 201, 2009.

[28] Chi-Wang Shu. Discontinuous Galerkin method for time-dependent problems: survey and recent developments. *Recent developments in discontinuous Galerkin finite element methods for partial differential equations*, pages 25–62, 2014.

[29] Douglas N Arnold, Franco Brezzi, Bernardo Cockburn, and Donatella Marini. Discontinuous Galerkin methods for elliptic problems. In *Discontinuous Galerkin Methods*, pages 89–101. Springer, 2000.

[30] Jacques Louis Lions. *Problemes aux Limites non Homogenes a Donnees Irregulieres*, pages 283–292. Springer Berlin Heidelberg, Berlin, Heidelberg, 1968.

[31] Joachim Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. In *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*, volume 36, pages 9–15. Springer, 1971.

[32] Ivo Babuška. The finite element method with penalty. *Mathematics of computation*, 27(122):221–228, 1973.

[33] Douglas N. Arnold. *An Interior Penalty Finite Element Method with Discontinuous Elements*. PhD thesis, The University of Chicago, 1979.

[34] Douglas N Arnold. An interior penalty finite element method with discontinuous elements. *SIAM journal on numerical analysis*, 19(4):742–760, 1982.

[35] Douglas N Arnold, Franco Brezzi, Bernardo Cockburn, and L Donatella Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM journal on numerical analysis*, 39(5):1749–1779, 2002.

[36] Béatrice Rivière. *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[37] Chi-Wang Shu. A brief survey on discontinuous Galerkin methods in computational fluid dynamics. *Advances in mechanics*, 43(6):541–553, 2013.

[38] Bernardo Cockburn. Discontinuous Galerkin methods for convection-dominated problems. In *High-order methods for computational physics*, pages 69–224. Springer, 1999.

[39] Bernardo Cockburn and Chi-Wang Shu. Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. *Journal of scientific computing*, 16(3):173–261, 2001.

[40] Douglas N. Arnold etc. Foreword for the special issue on discontinuous Galerkin method. *J. Sci. Comput.*, 22–23(1–3), 2005.

[41] Bernardo Cockburn and Chi-Wang Shu. Foreword. *J. Sci. Comput.*, 40(1–3):1–3, Jul 2009.

[42] Clint Dawson. Foreword. *Computer Methods in Applied Mechanics and Engineering*, 195(25):3183, 2006. Discontinuous Galerkin Methods.

[43] Yanlai Chen, Bo Dong, and Chi-Wang Shu. A foreword to the special issue in honor of Professor Bernardo Cockburn on his 60th birthday: A life time of discontinuous schemings. *J. Sci. Comput.*, 77(3):1303–1309, Dec. 2018.

[44] Jan Hesthaven, Jennifer Ryan, Chi-Wang Shu, Jaap Vegt, Yan Xu, Qiang Zhang, and Zhimin Zhang. Preface to focused issue on discontinuous galerkin methods. *Communications on Applied Mathematics and Computation*, pages 1–2, 10 2021.

[45] Jan S Hesthaven and Tim Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[46] Guido Kanschat. *Discontinuous Galerkin methods for viscous incompressible flow*. Springer, 2008.

[47] B.Q. Li. *Discontinuous Finite Elements in Fluid Dynamics and Heat Transfer*. Computational Fluid and Solid Mechanics. Springer London, 2005.

[48] Daniele Antonio Di Pietro and Alexandre Ern. *Mathematical aspects of discontinuous Galerkin methods*, volume 69. Springer Science & Business Media, 2011.

[49] V. Dolejší and M. Feistauer. *Discontinuous Galerkin Method: Analysis and Applications to Compressible Flow*. Springer Series in Computational Mathematics. Springer International Publishing, 2015.

[50] Eric T Chung, Patrick Ciarlet Jr, and Tang Fei Yu. Convergence and superconvergence of staggered discontinuous Galerkin methods for the three-dimensional Maxwell's equations on Cartesian grids. *Journal of Computational Physics*, 235:14–31, 2013.

[51] Bernardo Cockburn, Fengyan Li, and Chi-Wang Shu. Locally divergence-free discontinuous Galerkin methods for the Maxwell equations. *Journal of Computational Physics*, 194(2):588–610, 2004.

[52] Jan S Hesthaven and Tim Warburton. Nodal high-order methods on unstructured grids: I. time-domain solution of Maxwell's equations. *Journal of Computational Physics*, 181(1):186–221, 2002.

[53] Stephen D Gedney, John C Young, Tyler C Kramer, and J Alan Roden. A discontinuous Galerkin finite element time-domain method modeling of dispersive media. *IEEE transactions on antennas and propagation*, 60(4):1969–1977, 2012.

[54] Yunqing Huang, Jichun Li, and Wei Yang. Interior penalty DG methods for Maxwell's equations in dispersive media. *Journal of Computational Physics*, 230(12):4559–4570, 2011.

[55] Stéphane Lanteri and Claire Scheid. Convergence of a discontinuous Galerkin scheme for the mixed time-domain Maxwell's equations in dispersive media. *IMA Journal of Numerical Analysis*, 33(2):432–459, 2013.

[56] Tiao Lu, Pingwen Zhang, and Wei Cai. Discontinuous Galerkin methods for dispersive and lossy Maxwell's equations and PML boundary conditions. *Journal of Computational Physics*, 200(2):549–580, 2004.

[57] Eric T Chung and Patrick Ciarlet Jr. A staggered discontinuous Galerkin method for wave propagation in media with dielectrics and meta-materials. *Journal of Computational and Applied Mathematics*, 239:189–207, 2013.

[58] Jichun Li. Development of discontinuous Galerkin methods for Maxwell's equations in metamaterials and perfectly matched layers. *Journal of Computational and Applied Mathematics*, 236(5):950–961, 2011.

[59] Jichun Li and Jan S Hesthaven. Analysis and application of the nodal discontinuous Galerkin method for wave propagation in metamaterials. *Journal of Computational Physics*, 258:915–930, 2014.

[60] Jichun Li, Cengke Shi, and Chi-Wang Shu. Optimal non-dissipative discontinuous Galerkin methods for Maxwell's equations in Drude metamaterials. *Computers & Mathematics with Applications*, 73(8):1760–1780, 2017.

[61] Elisabeth Blank. *The Discontinuous Galerkin Method for Maxwell's Equations: Application to Bodies of Revolution and Kerr-Nonlinearities*. PhD thesis, Karlsruhe, Karlsruher Institut für Technologie (KIT), Diss., 2013, 2013.

[62] Loula Fezoui and Stéphane Lanteri. *Discontinuous Galerkin methods for the numerical solution of the nonlinear Maxwell equations in 1d*. PhD thesis, Inria, 2015.

[63] Juntao Huang and Chi-Wang Shu. A second-order asymptotic-preserving and positivity-preserving discontinuous Galerkin scheme for the Kerr–Debye model. *Mathematical Models and Methods in Applied Sciences*, 27(03):549–579, 2017.

[64] Zhichao Peng, Vrushali A Bokil, Yingda Cheng, and Fengyan Li. Asymptotic and positivity preserving methods for Kerr-Debye model with Lorentz dispersion in one dimension. *Journal of Computational Physics*, 402:109101, 2020.

[65] Rand Corporation and Richard Bellman. *Adoptive control processes: A guided tour*. Princeton University Press, 1961.

[66] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

[67] Jochen Garcke and Michael Griebel. *Sparse grids and applications*, volume 88. Springer Science & Business Media, 2012.

[68] C Zenger. Sparse Grids, Parallel Algorithms for Partial Differential Equations: Proceedings of the Sixth GAMM-Seminar, Kiel, January 1990, Notes on Numerical Fluid Mechanics (Vol. 31), W. *Hackbusch, eds., Vieweg, Braunschweig*, 1991.

[69] Sergei Abramovich Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Doklady Akademii Nauk*, volume 148, pages 1042–1045. Russian Academy of Sciences, 1963.

[70] Konstantin Ivanovich Babenko. Approximation of periodic functions of many variables by trigonometric polynomials. In *Doklady Akademii Nauk*, volume 132, pages 247–250. Russian Academy of Sciences, 1960.

[71] Vladimir Nikolaevich Temlyakov. Approximations of functions with bounded mixed derivative. *Trudy Matematicheskogo Instituta imeni VA Steklova*, 178:3–113, 1986.

[72] F-J Delvos. d-variate boolean interpolation. *Journal of Approximation Theory*, 34(2):99–114, 1982.

[73] G Baszenski, F-J Delvos, and S Jester. Blending approximations with sine functions. In *Numerical Methods in Approximation Theory, Vol. 9*, pages 1–19. Springer, 1992.

[74] Chin Bo Liem and Tsimin Shih. *Splitting Extrapolation Method, the: A New Technique In Numerical Solution Of Multidimensional Prob*, volume 7. World Scientific, 1995.

[75] Michael Griebel. *Sparse grids and related approximation schemes for higher dimensional problems*. SFB 611, 2005.

[76] Bradley K Alpert. A class of bases in L^2 for the sparse representation of integral operators. *SIAM journal on Mathematical Analysis*, 24(1):246–262, 1993.

[77] Beylkin Alpert, Gregory Beylkin, David Gines, and Lev Vozovoi. Adaptive solution of partial differential equations in multiwavelet bases. *Journal of Computational Physics*, 182(1):149–190, 2002.

[78] Stephane Mallat. A wavelet tour of signal processing: An approximation tour, 1999.

[79] Yuan Liu, Yingda Cheng, Shanqin Chen, and Yong-Tao Zhang. Krylov implicit integration factor discontinuous Galerkin methods on sparse grids for high dimensional reaction-diffusion equations. *Journal of Computational Physics*, 388:90–102, 2019.

[80] Wei Guo. A sparse grid discontinuous Galerkin method for the high-dimensional Helmholtz equation with variable coefficients. *arXiv preprint arXiv:1905.08917*, 2019.

[81] Zhanjing Tao, Wei Guo, and Yingda Cheng. Sparse grid discontinuous Galerkin methods for the Vlasov-Maxwell system. *Journal of Computational Physics: X*, 3:100022, 2019.

[82] Jianguo Huang and Yue Yu. A sparse grid discrete ordinate discontinuous Galerkin method for the radiative transfer equation. *arXiv preprint arXiv:2101.09070*, 2021.

[83] Juntao Huang, Yuan Liu, Wei Guo, Zhanjing Tao, and Yingda Cheng. An adaptive multiresolution interior penalty discontinuous Galerkin method for wave equations in second order form. *Journal of Scientific Computing*, 85(1):1–31, 2020.

[84] Jie Shen and Haijun Yu. Efficient spectral sparse grid methods and applications to high-dimensional elliptic problems. *SIAM Journal on Scientific Computing*, 32(6):3228–3250, 2010.

[85] Andreas Zeiser. Fast matrix-vector multiplication in the sparse-grid Galerkin method. *Journal of Scientific Computing*, 47(3):328–346, 2011.

[86] Juntao Huang, Yong Liu, Yuan Liu, Zhanjing Tao, and Yingda Cheng. A class of adaptive multiresolution ultra-weak discontinuous Galerkin methods for some nonlinear dispersive wave equations. *SIAM Journal on Scientific Computing*, 44(2):A745–A769, 2022.

[87] Wei Guo, Juntao Huang, Zhanjing Tao, and Yingda Cheng. An adaptive sparse grid local discontinuous Galerkin method for Hamilton-Jacobi equations in high dimensions. *Journal of Computational Physics*, 436:110294, 2021.

[88] Zhanjing Tao, Juntao Huang, Yuan Liu, Wei Guo, and Yingda Cheng. An adaptive multiresolution ultra-weak discontinuous Galerkin method for nonlinear Schrödinger equations. *Communications on Applied Mathematics and Computation*, 4(1):60–83, 2022.

[89] Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes. *Acta Numerica*, 29:701–762, 2020.

[90] Chi-Wang Shu. High order ENO and WENO schemes for computational fluid dynamics. In *High-order methods for computational physics*, pages 439–582. Springer, 1999.

[91] A. Harten, B. Engquist, S. Osher, and S.R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes. III. *Journal of Computational Physics (Print)*, 71(2):231–303, 1987.

[92] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of computational physics*, 77(2):439–471, 1988.

[93] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes, II. *Journal of Computational Physics*, 83(1):32–78, 1989.

[94] Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. *Journal of computational physics*, 115(1):200–212, 1994.

[95] Oliver Friedrich. Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids. *Journal of computational physics*, 144(1):194–212, 1998.

[96] Changqing Hu and Chi-Wang Shu. High order weighted ENO schemes for unstructured meshes - preliminary results. *Computational fluid dynamics'98*, pages 356–362, 1998.

[97] Changqing Hu and Chi-Wang Shu. Weighted essentially non-oscillatory schemes on triangular meshes. *Journal of Computational Physics*, 150(1):97–127, 1999.

[98] Doron Levy, Gabriella Puppo, and Giovanni Russo. Central WENO schemes for hyperbolic systems of conservation laws. *ESAIM: Mathematical Modelling and Numerical Analysis*, 33(3):547–571, 1999.

[99] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted eno schemes. *Journal of computational physics*, 126(1):202–228, 1996.

[100] Dinshaw S Balsara and Chi-Wang Shu. Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy. *Journal of Computational Physics*, 160(2):405–452, 2000.

[101] Chi-Wang Shu. High order weighted essentially nonoscillatory schemes for convection dominated problems. *SIAM review*, 51(1):82–126, 2009.

[102] Michael G Crandall and Pierre-Louis Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American mathematical society*, 277(1):1–42, 1983.

[103] Michael G Crandall, Hitoshi Ishii, and Pierre-Louis Lions. User's guide to viscosity solutions of second order partial differential equations. *Bulletin of the American mathematical society*, 27(1):1–67, 1992.

[104] Guy Barles. An introduction to the theory of viscosity solutions for first-order Hamilton–Jacobi equations and applications. In *Hamilton-Jacobi equations: approximations, numerical analysis and applications*, pages 49–109. Springer, 2013.

[105] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[106] Stanley Osher and Chi-Wang Shu. High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations. *SIAM Journal on numerical analysis*, 28(4):907–922, 1991.

[107] Guang-Shan Jiang and Danping Peng. Weighted ENO schemes for Hamilton–Jacobi equations. *SIAM Journal on Scientific computing*, 21(6):2126–2143, 2000.

[108] Chi-Wang Shu. Total-variation-diminishing time discretizations. *SIAM Journal on Scientific and Statistical Computing*, 9(6):1073–1084, 1988.

[109] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of computation*, 67(221):73–85, 1998.

[110] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM review*, 43(1):89–112, 2001.

[111] G.P. Agrawal. *Nonlinear Fiber Optics*. Electronics & Electrical. Elsevier Science, 2007.

[112] Peter M Goorjian, Allen Taflove, Rose M Joseph, and Susan C Hagness. Computational modeling of femtosecond optical solitons from Maxwell's equations. *IEEE journal of quantum electronics*, 28(10):2416–2422, 1992.

[113] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[114] Shrirang Abhyankar, Jed Brown, Emil M Constantinescu, Debojyoti Ghosh, Barry F Smith, and Hong Zhang. PETSc/TS: A modern scalable ODE/DAE solver library. *arXiv preprint arXiv:1806.01437*, 2018.

[115] Peter R Brune, Matthew G Knepley, Barry F Smith, and Xuemin Tu. Composing scalable nonlinear algebraic solvers. *SIAM Review*, 57(4):535–565, 2015.

[116] Dana A Knoll and David E Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[117] N. Bloembergen. *Nonlinear Optics*. World Scientific, 1996.

[118] R.W. Boyd. *Nonlinear Optics*. Elsevier Science, 2003.

[119] G. New. *Introduction to Nonlinear Optics*. Cambridge University Press, 2011.

[120] M.N.O. Sadiku. *Computational Electromagnetics with MATLAB, Fourth Edition*. CRC Press, 2018.

[121] L Gilles, SC Hagness, and L Vázquez. Comparison between staggered and unstaggered finite-difference time-domain grids for few-cycle temporal optical soliton propagation. *Journal of Computational Physics*, 161(2):379–400, 2000.

[122] Cheryl V Hile and William L Kath. Numerical solutions of Maxwell's equations for nonlinear-optical pulse propagation. *JOSA B*, 13(6):1135–1145, 1996.

[123] Olivier Bokanowski, Jochen Garcke, Michael Griebel, and Irene Klompmaker. An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations. *Journal of Scientific Computing*, 55(3):575–605, 2013.

[124] Michael Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.

[125] W. Hundsdorfer and J.G. Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2013.

[126] Dimitri van Heesch. Doxygen web page. https://www.doxygen.nl/index.html, 1997-2019.

[127] Jethro H Greene and Allen Taflove. General vector auxiliary differential equation finite-difference time-domain method for nonlinear optics. *Optics express*, 14(18):8305–8310, 2006.

[128] Scientific Grand Challenges. Architectures and technology for extreme scale computing. In *US Department of Energy Workshop Report*, 2009.

[129] Jack Dongarra, Jeffrey Hittinger, John Bell, Luis Chacón, Robert Falgout, Michael Heroux, Paul Hovland, Esmond Ng, Clayton Webster, and Stefan Wild. Applied mathematics research for exascale computing. Technical report, U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research Program, 2014.

[130] David E. Culler, Richard M. Karp, David A. Patterson, Abhijit Sahay, Eunice E. Santos, Klaus E. Schauser, Ramesh Subramonian, and Thorsten von Eicken. LogP: a practical model of parallel computation. *Commun. ACM*, 39:78–85, 1996.

[131] A. Pietracaprina and G. Pucci. The complexity of deterministic PRAM simulation on distributed memory machines. *Theory of Computing Systems*, 30(3):231–247, 1997.

[132] Alberto Nunez, Javier Fernandez, Rosa Filguiera, Felix Garcia, and Jesus Carretero. SIMCAN: A flexible, scalable and expandable simulation platform for modelling and simulating distributed architectures and applications. *Simulation Modelling Practice and Theory*, 20(1):12–32, 2012.

[133] Julian M. Kunkel. Simulating parallel programs on application and system level. *Computer Science - Research and Development*, 28(2):167–174, 2013.

[134] S.S. Dosanjh, R.F. Barrett, D.W. Doerfler, S.D. Hammond, K.S. Hemmert, M.A. Heroux, P.T. Lin, K.T. Pedretti, A.F. Rodrigues, T.G. Trucano, and J.P. Luitjens. Exascale design space exploration and co-design. *Future Generation Computer Systems*, 30:46 – 58, 2014.

[135] Kyle L. Spafford and Jeffrey S. Vetter. Aspen: A domain specific language for performance modeling. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 84:1–84:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[136] Jeffrey S. Vetter and Jeremy S. Meredith. Synthetic program analysis with Aspen. In *Proceedings of the 3rd International Conference on Exascale Applications and Software*, Edinburgh, Scotland, UK, 04/2015 2015. University of Edinburgh, University of Edinburgh.

[137] Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J. Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, and Kent Wenger. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17 – 35, 2015.

[138] A. Aw and M. Rascle. Resurrection of "second order" models of traffic flow. *SIAM Journal on Applied Mathematics*, 60(3):916–938, 2000.

[139] D. Armbruster, D. Marthaler, and C. Ringhofer. Kinetic and fluid model hierarchies for supply chains. *Multiscale Modeling & Simulation*, 2(1):43–61, 2003.

[140] Mapundi K. Banda, Michael Herty, and Axel Klar. Gas flow in pipeline networks. *Networks and Heterogeneous Media*, 1(1):41–56, 2006.

[141] Jens Brouwer, Ingenuin Gasser, and Michael Herty. Gas pipeline models revisited: Model hierarchies, nonisothermal models, and simulations of networks. *Multiscale Modeling & Simulation*, 9(2):601–623, 2011.

[142] Axel Klar and Raimund Wegener. A hierarchy of models for multilane vehicular traffic. I. Modeling. *SIAM J. Appl. Math.*, 59(3):983–1001 (electronic), 1999.

[143] Ernst Hairer, Syvert P Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff problems*. Springer, 1987.

[144] Cory D Hauck, Michael Herty, and Giuseppe Visconti. Qualitative properties of mathematical model for data flow. *Networks & Heterogeneous Media*, 16(4), 2021.