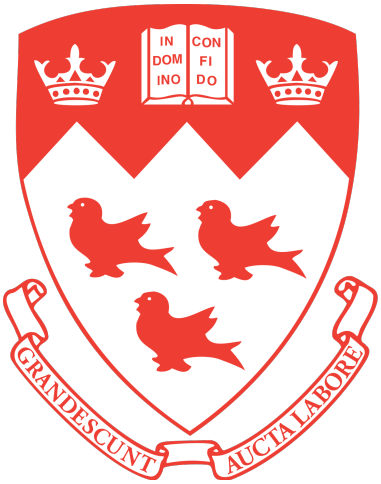
Improving the Numerical Stability of Higher Order Methods with Applications to Fluid Dynamics

Abtin Ameri

Computational Aerodynamics Group, McGill University

Supervisor: Professor Siva Nadarajah



A thesis submitted to McGill University in partial fulfillment of the requirements of the Undergraduate Honours Program.

December 3, 2019

Contents

1 Introduction **6**

1.1 Motivation 6

1.1.1 Shock Waves and Turbulence 7

1.1.2 An Example – Burgers' Equation 9

1.2 Objective 10

1.3 Outline 12

2 The Discontinuous Galerkin Method **13**

2.1 Background 13

2.1.1 Finite Difference (FD) 14

2.1.2 Finite Volume (FV) 14

2.1.3 Finite Element (FE) 15

2.1.4 Discontinuous Galerkin (DG) 16

2.2 Mathematical Formulation of DG in 1D 16

2.2.1 Integration and Interpolation 19

2.2.2 Node Placement 20

3 Numerical Stability **21**

3.1 Nonlinear Problems 21

3.2 Collocation 22

3.3 Flux Splitting and Burgers' Equation 23

3.4 Extension to Euler's Equations . . . 25
3.4.1 DG in Three Dimensions . . . 27
3.5 Two-Point Flux . . . 28
3.6 Implementation . . . 30
3.6.1 `physics` . . . 31
3.6.2 `DG` . . . 31
3.6.3 `numerical_flux` . . . 31
3.6.4 `ode_solver` . . . 32
3.6.5 `parameters` . . . 32

4 Results and Conclusion **33**
4.1 Burgers' Equation . . . 33
4.2 Euler's Equations . . . 36
4.3 Conclusion and Future Work . . . 38

Abstract

The discontinuous Galerkin (DG) method, which has recently gained popularity within the finite element community, is prone to numerical instabilities for high orders unless furnished with stabilizers. The goal of this thesis was to improve the numerical stability of DG schemes without the use of stabilizers. In particular, the nonlinear stability of Burgers' equation was studied, and the findings were extended to Euler's equations of fluid dynamics. It was demonstrated that by splitting the flux term of the governing equations, nonlinear stability can be achieved. In the future, this technique can be extended to the Navier-Stokes equations and the magnetohydrodynamics equations.

Abstrait

La méthode Galerkin discontinue (GD), qui a récemment gagné en popularité au sein de la communauté des éléments finis, est sujette à des instabilités numériques aux ordres élevés, sauf si elle est fournie avec des stabilisants. Le but de cette thèse était d'améliorer la stabilité numérique des schémas GD sans utiliser de stabilisants. En particulier, la stabilité non linéaire de l'équation de Burgers a été étudiée et les résultats ont été étendus aux équations d'Euler de la dynamique des fluides. Il a été démontré qu'en séparant le terme de flux des équations de gouvernance, on pouvait obtenir une stabilité non linéaire. À l'avenir, cette technique pourra être étendue aux équations de Navier-Stokes et de la magnétohydrodynamique.

Acknowledgements

I would like to extend my most sincere gratitude to my supervisor, mentor, and role model, Professor Nadarajah, who guided me throughout this project and pushed me to be a better researcher and person.

Special thanks to Doug Shi-Dong and Aditya Kashi for their extensive help with my project. Without Doug, I would not have been able to make any progress with my code, and without Aditya, I would not have half of the knowledge of C++ I do today. Finally, I would like to thank my friend Zhuo Fan Bao for his support throughout my thesis and my degree.

This thesis is dedicated to my family: my supportive parents, Azam and Hassan, and lovely siblings, Ali and Golnoosh.

Chapter 1

Introduction

1.1 Motivation

Many equations in physics are highly nonlinear partial differential equations (PDEs). A prominent example is the governing equations of fluid dynamics. These equations lack closed-form solutions for many problems of interest. Many techniques exist to find approximate solutions; an example is perturbation theory, which, given a closed form solution, adds small (nonlinear) disturbances to the system and examines the change in behaviour [1]. However, with the current state of computation and rapid advances in high-performance computing, the most powerful method of solving complex, nonlinear equations is through the use of numerical methods.

Numerical methods deal with the discretization of the governing equations of the phenomenon of interest, the development of a numerical algorithm, and the establishment of a numerical code that is to be executed on current high-performance computers. Many problems exist in numerical methods which do not exist in the physical world. The physical world is infinite and smooth, and the world of computers is finite and discrete; the transition from the physical world to the computational world leads to the creation of unwanted effects.

One such effect is numerical instability. In many problems of interest, we are interested

in long-term behaviour of the property of the fluid such as pressure or velocity. Thus, we need to integrate the discretized governing equations in time. It is well known that high-order methods exhibit instabilities during time integration [2, 3]. In fluid mechanics, this is especially the case when simulating shock waves or turbulence.

1.1.1 Shock Waves and Turbulence

Two prominent phenomena in fluid mechanics are shock waves and turbulent flow. Shock waves are present in supersonic flow, and are nature's method of communicating information upstream of the flow. Shock waves cause a drastic change in velocity, pressure, and density over very small distances; thus, they are mathematically modelled as a discontinuity. Oblique shock waves formed around a supersonic fighter jet are shown in fig. 1.1.

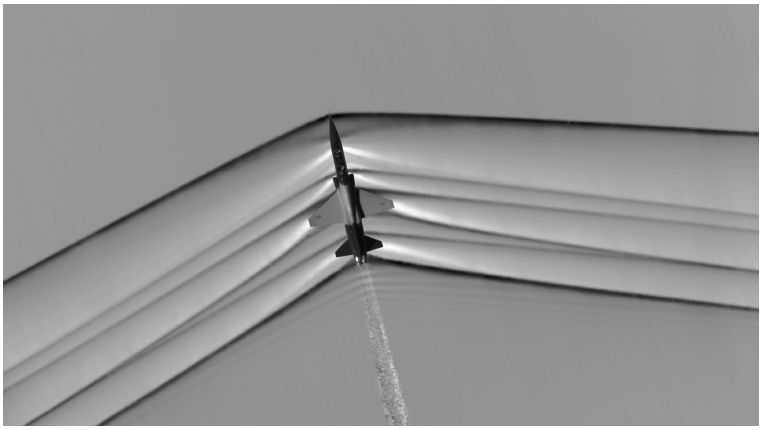


Figure 1.1: Oblique shock waves formed on a supersonic fighter jet. Figure taken from [4].

Turbulent flow presents itself in high Reynolds number regimes [4], with the Reynolds number given by:

$$Re = \frac{\rho u L}{\mu}, \tag{1.1}$$

where ρ is the density, u the velocity, L the characteristic length, and μ the viscosity. Reynolds number is a comparison of the inertial forces to the viscous forces in a fluid. Low Reynolds numbers correspond to laminar flow regimes, where the fluid flows in an orderly,

reversible fashion. As the Reynolds number increases, the flow tends towards the turbulent regime, characterized by a chaotic and unpredictable behaviour of the fluid. This is shown in fig. 1.2. In turbulence, there is a large variation of fluid properties, such as velocity, pressure, and density, over small distances.

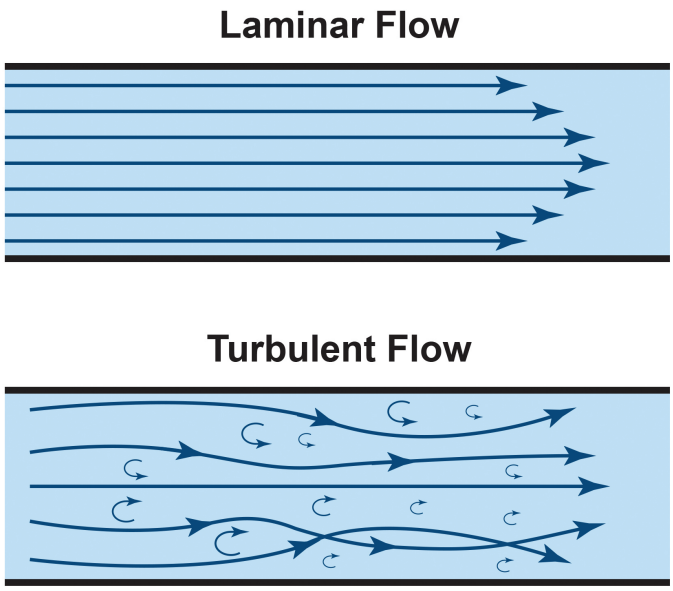


Figure 1.2: Laminar versus turbulent flow. Turbulent flow, which occurs at high Reynolds numbers, is unpredictable and disorderly. Figure taken from [4].

Due to their complex nature, turbulence and shock waves are best studied through numerical methods. However, many challenges arise when simulating these phenomena through simulations. For instance, high-order numerical methods face instabilities when resolving and capturing discontinuities such as shock waves because the solution loses smoothness at the vicinity of the shock and the gradient of the solution diverges. Since computers cannot deal with infinities, the solutions can become numerically unstable unless stabilizers are explicitly added to the scheme.

Shocks are best captured using pure upwinding schemes. Such schemes are known as total variation diminishing (TVD). A key theorem to note is Godunov's theorem [5], which states that schemes that preserve the monotonicity and, hence, the TVD property, are at

most first-order accurate. This indicates that higher-order methods can introduce oscillations near the shock; and, coupled with their inherent low numerical dissipation at high polynomial orders, such oscillations can grow unbounded. Techniques to restore stability and ensure the capture of discontinuities have been an active field of research for the past two decades [6]. For instance, in order to prevent high frequency oscillations in the vicinity of discontinuities from growing, one can use a filter [7]. Additional techniques to capture shocks involve using slope limiters [8], which reduce the scheme to first-order (TVD) at the shock. Furthermore, one can add artificial viscosity to the equations to dissipate the shocks [9]. The dissipation added by artificial viscosity allows the shocks to be captured by high-order schemes. In schemes that involve integration (such as the discontinuous Galerkin method), over-integration of fluxes is a well-known method of reducing aliasing errors [10]. Finally, refining the grid as well as the time steps can improve numerical stability. Absences of any of the mentioned methods results in divergence of the solution, leading to instability for high-order methods. This is best demonstrated through an example.

1.1.2 An Example – Burgers' Equation

Consider Burgers' equation, which is a PDE given as:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0. \tag{1.2}$$

Burgers' equation is of particular interest in the numerical methods community because it is the simplest nonlinear equation that can be studied. The key property of this equation is that, given a smooth initial condition, a shock forms in finite time.

A well-known numerical method that is used to discretize Burger's equation is the discontinuous Galerkin (DG) scheme. DG has recently gained popularity in the finite element community due to its high parallelizability and effectiveness when it comes to solving hyperbolic PDEs [11]. If the PDE is discretized using a standard, high-order DG scheme, the

solution becomes numerically unstable and diverges as soon as a shock forms. Fig. 1.3 verifies this by showing the computed solution of the PDE at various times. It can be seen that after the shock forms, oscillations appear in the vicinity of the shock and grow in time until the solution finally becomes unstable. This is due to the fact that the DG scheme does not satisfy the nonlinear stability requirements, which will be developed in Chapter 3.

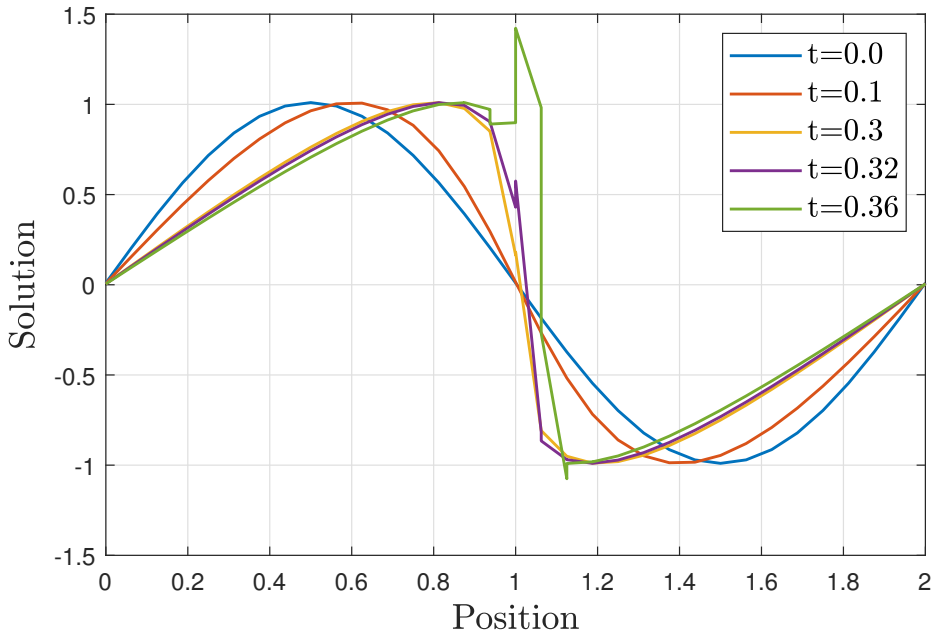


Figure 1.3: Solution of Burgers' equation given the initial condition $u(x,0) = \sin(\pi x) + 0.01$. The method is 8th order accurate in space. After shock formation at $t \approx 0.3s$, the standard DG solution becomes numerically unstable. The simulation cannot be run after $t = 0.36s$

1.2 Objective

There are various issues with current shock capturing methods. Adding artificial viscosity involves adding non-physical terms to the governing equations, which means one does not solve the original equations aimed to be solved; furthermore, it increases the numerical dissipation and consequently it reduces the numerical accuracy. Adding filters and slope-limiters is computationally expensive as it adds additional steps to calculating the solution. Similarly, over-integration can be very expensive, especially when it comes to higher-dimensional

problems and fine grids. Ideally, one should find a way of re-writing the governing equations without changing the actual underlying physics while achieving stability. The objective of this thesis is to develop a scheme that would improve the numerical stability of high-order methods without the use of stabilizers, allowing for longer simulation run times and for the study of long-term behaviour of fluids. Specifically, Burgers' equation and Euler's equations of fluid mechanics are investigated.

The techniques developed in this thesis, although initially focused on shock capture, can be applied to turbulent flow as well. This is because turbulence and shock waves are similar, as both involve large variations in fluid properties over small distances. Turbulent flow is generally studied through large eddy simulations (LES), which solves the filtered Navier-Stokes equations. An example of LES used to study turbulent combustion flames is shown in fig. 1.4. High-order LES methods, similar to high order shock capture methods, are prone to instabilities [12]. Thus, methods that improve the stability of high-order shock capture schemes can be applied to high-order LES schemes, facilitating the study of long-term turbulent flow behaviour.

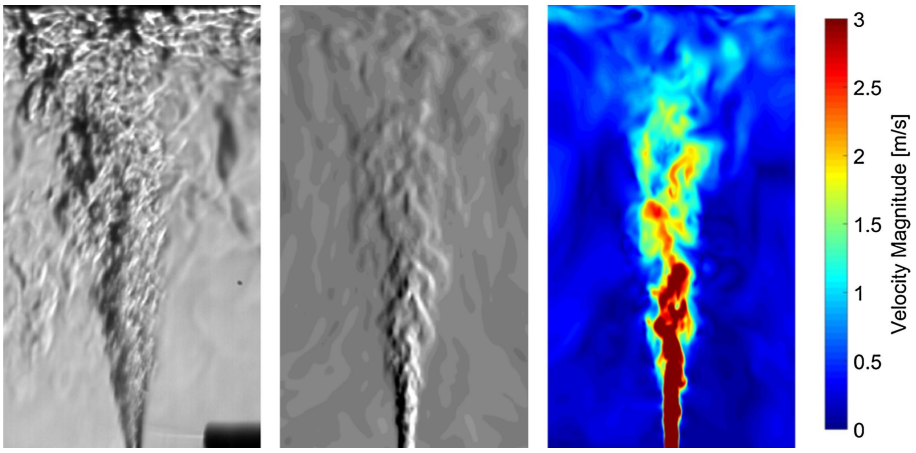


Figure 1.4: An example of LES applicable to study of turbulence in combustion flames. Figure taken from [13].

1.3 Outline

This thesis is structured as follows: Chapter 2 describes the DG method, which is the numerical scheme used to discretize Burgers' equation and Euler's equations of fluid dynamics. Chapter 3 delves into the theory of numerical stability and introduces a technique, known as flux splitting, which can help improve the stability of high-order methods. Burgers' equation will initially be considered before analyzing the Euler's equations due to its simplicity. Finally, Chapter 4 will discuss the results obtained through numerical simulations. Conclusion and future work end the chapter.

Chapter 2

The Discontinuous Galerkin Method

2.1 Background

There are various methods of discretizing PDEs. The three primary approaches are finite difference (FD), finite volume (FV), and finite element (FE). The scheme considered in this thesis is discontinuous Galerkin (DG), which inherits its properties from both FE and FV. Each method will be explained briefly below, in addition to its advantages and disadvantages being assessed, before delving deeper into the theory of DG.

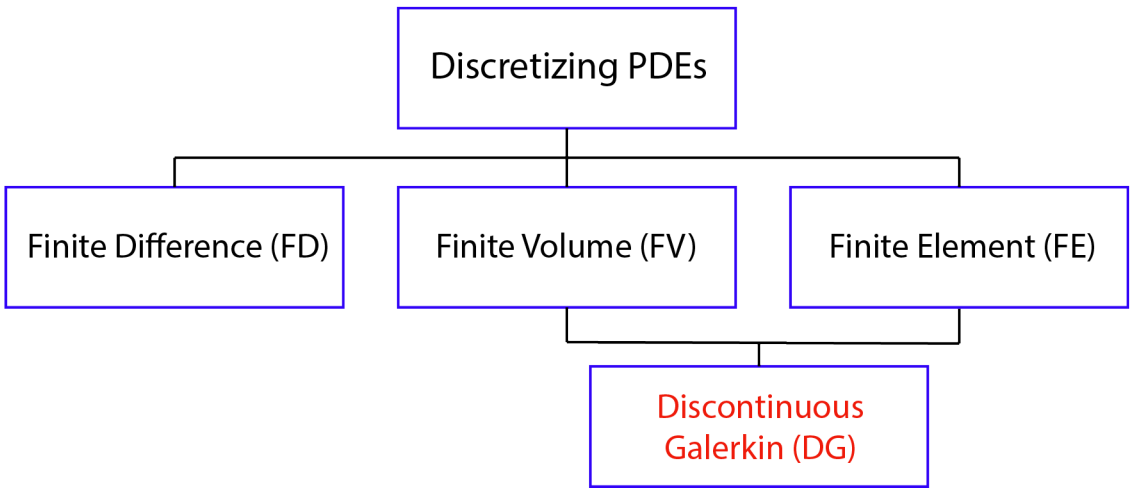


Figure 2.1: Classification of various approaches used to discretize and solve PDEs. The DG method is a hybridization of FV and FE.

2.1.1 Finite Difference (FD)

FD is a very common discretization which has a much gentler learning curve compared to the rest of the methods. In FD, one solves for the solution at individual points on the grid. Operations such as differentiation are done by using the neighbouring points. For instance, given two points (x_i, y_i) and (x_{i+1}, y_{i+1}), the first-order first derivative with respect to x is given by:

$$\left.\frac{dy}{dx}\right|_{x=x_i} \approx \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{2.1}$$

The advantage of FD is that it is relatively easy to implement, and is usually the first choice for quickly solving simple problems. Moreover, extensive research has been done in the field and most of the theory is well-understood [14]. However, a major drawback of FD is that it becomes increasingly difficult to implement for complex geometries. Furthermore, it requires solving a global system, which can be computationally expensive for fine grids. Finally, high-order methods require a large stencil, which increases the communication cost between processors when the computation domain is parallelized across multiple computing cores. The increased communication cost degrades the parallel scalability of high-order finite difference approaches.

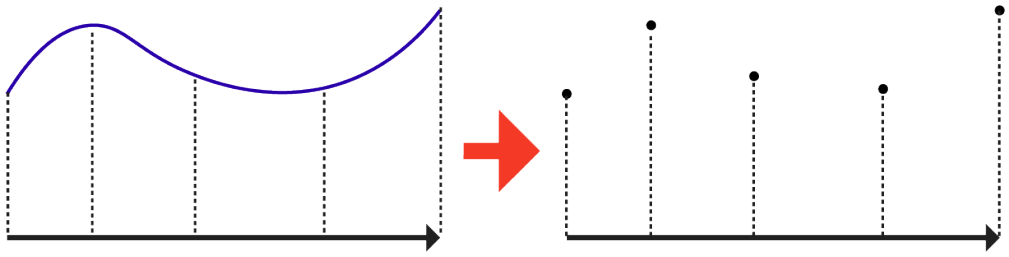


Figure 2.2: FD reduces the solution to points in the domain. The derivatives are calculated with respect to the points.

2.1.2 Finite Volume (FV)

FV is different from FD in the sense that it divides the domain into cells (or elements) as opposed to reducing the solution to individual points. FV takes the average of the solution

over each cell and, thus, assumes the solution to be constant over each element. This means that the solution has to be discontinuous across boundaries [15].

FV is excellent at dealing with complex geometries as it allows us to divide the grid into triangular or quadrilateral elements for two-dimensional problems, and tetrahedra or hexahedra for three-dimensional problems. In addition, because of the boundary discontinuities, the scheme is highly parallelizable. This allows the computational grid to be divided into sub-grids, which are then fed into different processors of a supercomputer, allowing for shorter run times. However, higher-order FV methods suffer the same fate where the need for large stencils leads to a reduction of the parallel efficiency.

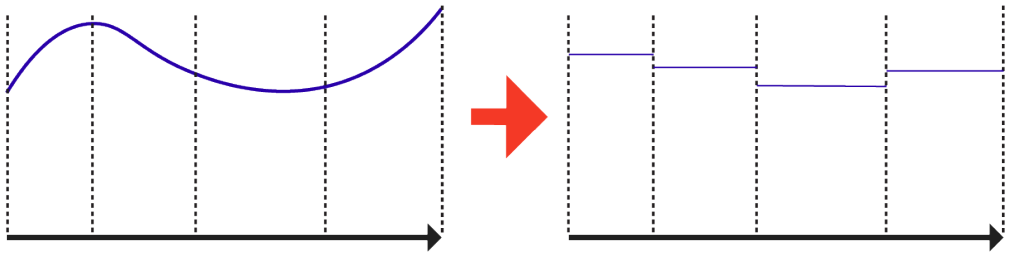


Figure 2.3: FV divides the domain into cells and averages the solution over each element, creating discontinuities across cell boundaries.

2.1.3 Finite Element (FE)

Similar to FV, FE divides the domain into cells, which means it is great at dealing with complex geometries. However, instead of taking the average of the solution over each cell, FE approximates the solution as a linear combination of a finite number of basis functions, and the solution is required to be continuous across the domain. FE has a great advantage over FD and FV because high-order schemes don't require a large stencil; we can simply increase the number of basis functions in the cells without increasing the dependency of the cells on each other. The disadvantage, however, is that one would be required to solve a global system since the solution is continuous across cell boundaries. Thus, this method is computationally expensive for fine grids or high orders.

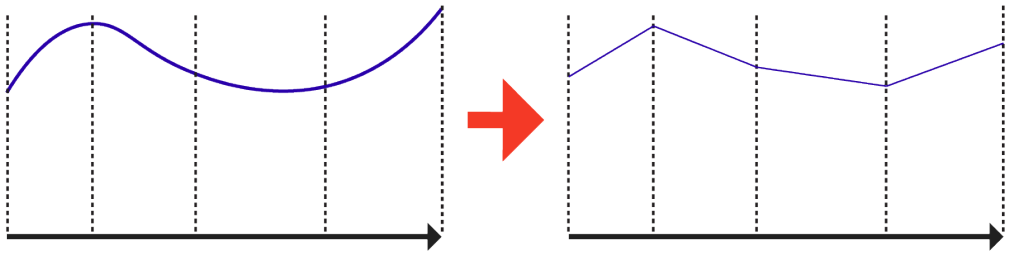


Figure 2.4: FE divides the domain into cells and approximates the solution as a linear combination of basis functions. This method constrains the solution to be continuous.

2.1.4 Discontinuous Galerkin (DG)

Lastly, DG brings together attractive features of FE and FV: the solution is a linear combination of basis functions, but it is allowed to be discontinuous at the cell boundaries. Thus, DG is highly parallelizable and higher-order schemes don't require a larger stencil. The disadvantage of DG is that the discontinuity leads to more degrees of freedom compared to FE, which can be computationally expensive for fine grids. However, this is usually masked by the high parallelizability of the scheme. For these reasons, DG has gained popularity in recent years within the finite element community.

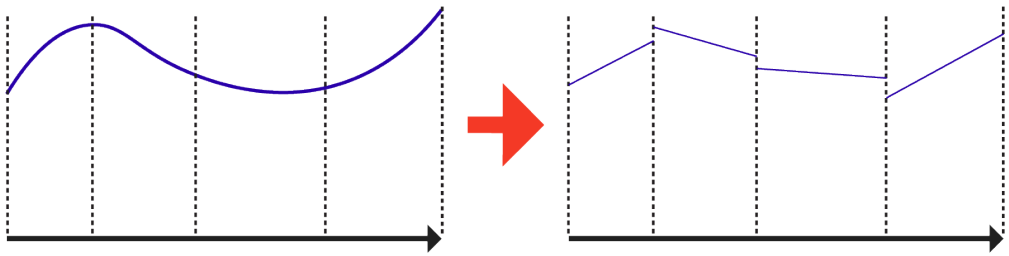


Figure 2.5: DG is similar to FE, but allows for discontinuities across cell boundaries.

2.2 Mathematical Formulation of DG in 1D

Let us assume a nonlinear PDE of the form:

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \tag{2.2}$$

where $u(x,t)$ is the solution and $f(u(x,t))$ is the flux [11]. The PDE is solved on the domain Ω. The first step in the DG approach is to approximate the domain Ω with a computational domain Ω_h, with K non-overlapping elements defined as:

$$\Omega \approx \Omega_h = \bigcup_{k=1}^{K} D_k, \text{ where } D_k = [x^k, x^{k+1}]. \tag{2.3}$$

One can then approximate the solution and flux in element D_k as a linear combination of p basis functions $\{\psi_n\}$:

$$u(x,t) \approx u_h^k = \sum_{n=1}^{p} u_h^n \psi_n(x), \tag{2.4}$$

$$f(u(x,t)) \approx f_h^k = \sum_{n=1}^{p} f_h^n \psi_n(x), \tag{2.5}$$

where p is the polynomial approximation order, and the coefficients u_h^n, f_h^n are unknowns. Note, however, that f_h^n depends on u_h^n, so essentially u_h^n are the only unknowns. Next, we multiply (2.2) by a test function ϕ_m, and integrate over D_k:

$$\int_{D_k} \left(\frac{\partial u_h^k}{\partial t} \phi_m + \frac{\partial f_h^k}{\partial x} \phi_m \right) dx = 0. \tag{2.6}$$

This yields a single equation for p unknown coefficients u_h^n. Hence, we require p test functions to obtain a solvable system of equations.

The problem with (2.6) is that there is no interaction between cell D_k and its neighbouring cells D_{k-1} and D_{k+1}. In order to couple the flux between the cells and enforce conservation, we integrate by parts and replace the flux at the boundaries by a numerical flux f^*, which takes in values of the solution at both sides of the boundary:

$$\int_{D_k} \left(\frac{\partial u_h^k}{\partial t} \phi_m - f_h^k \frac{\partial \phi_m}{\partial x} \right) dx = -[f^* \phi_m]_{x^k}^{x^{k+1}}, \tag{2.7}$$

giving the weak formulation of DG. A second integration by parts generates the strong

formulation:

$$\int_{D_k} \left(\frac{\partial u_h^k}{\partial t}\phi_m + \frac{\partial f_h^k}{\partial x}\phi_m \right) dx = -[(f_h^k - f^*)\phi_m]_{x^k}^{x^{k+1}}. \tag{2.8}$$

The strong form will be the main consideration of this thesis. The choice of numerical flux is dependent on the physics of the problem as long as it maintains the consistency criterion:

$$f^*(u,u) = f(u). \tag{2.9}$$

We can set the test and basis functions to be the same. A natural choice is Lagrange polynomials:

$$\phi_j(x) = \psi_j(x) = \ell_j^k(x) = \prod_{i=0, i\neq j}^{p} \frac{x - x_i}{x_j - x_i}. \tag{2.10}$$

The Lagrange polynomials interpolate the solution points within the cells. The polynomials satisfy the following property:

$$\ell_i(x_j) = \delta_{ij}. \tag{2.11}$$

By collocating both the solution nodes and flux nodes, where the flux is evaluated, a nodal DG scheme is constructed, where the solution is approximated as:

$$u_h = \sum_{n=1}^{p} u_h^n(x_n, t)\ell_n^k(x). \tag{2.12}$$

Using (2.10), (2.8) can be re-written in operator form:

$$\underline{\underline{M}}^k \frac{d\underline{u}_h^k}{dt} + \underline{\underline{S}}^k \underline{f}_h^k = (f_h^k - f^*)\underline{\ell}^k \Big|_{x^k}^{x^{k+1}}, \tag{2.13}$$

where

$$\underline{u}_h^k = \begin{bmatrix} u_h^1 \\ u_h^2 \\ \vdots \\ u_h^p \end{bmatrix}, \ \underline{f}_h^k = \begin{bmatrix} f_h^1 \\ f_h^2 \\ \vdots \\ f_h^p \end{bmatrix}, \ \underline{\ell}^k(x) = \begin{bmatrix} \ell_1^k(x) \\ \ell_2^k(x) \\ \vdots \\ \ell_p^k(x) \end{bmatrix}. \tag{2.14}$$

$\underline{\underline{S}}^k$ and $\underline{\underline{M}}^k$ are the element's stiffness and mass matrices, respectively:

$$\underline{\underline{M}}^k_{ij} = \int_k \ell^k_i(x)\ell^k_j(x)dx,\ \underline{\underline{S}}^k_{ij} = \int_k \ell^k_i(x)\frac{d\ell^k_j(x)}{dx}dx. \tag{2.15}$$

The two matrices are related by the differentiation matrix $\underline{\underline{D}}^k$:

$$\underline{\underline{S}}^k = \underline{\underline{M}}^k\ \underline{\underline{D}}^k. \tag{2.16}$$

The integrals are calculated using quadrature. We can solve for $\frac{d\underline{u}^k_h}{dt}$ and integrate in time using a classic explicit ODE solver, such as explicit Euler or Runge-Kutta. For explicit schemes, the time step needs to satisfy a Courant-Friedrichs-Lewy (CFL) condition to maintain stability [11]. The global solution is the direct sum of the local solutions:

$$u_h(x,t) = \bigoplus_{k=1}^{K} u^k_h(x,t). \tag{2.17}$$

The optimal order of convergence of DG schemes is $O(\Delta x^{p+1})$, which can be obtained through a grid convergence study.

2.2.1 Integration and Interpolation

In order to carry out the integrals in DG, one can use quadrature. This would require that the integral would be approximated using quadrature weights and nodes. For instance, the mass matrix would be evaluated as follows:

$$\underline{\underline{M}}^k_{ij} = \int_k \ell^k_i(x)\ell^k_j(x)dx \approx \sum_{m=1}^{N} w_m\ell^k_i(x_m)\ell^k_j(x_m), \tag{2.18}$$

where w_m are the quadrature weights and x_m are the quadrature points. These values depend on the type of quadrature used.

2.2.2 Node Placement

The placement of the integration and interpolation nodes plays an important role in the development of the scheme. For integration, one can use quadratures such as Gauss-Legendre (better known as Gauss) or Gauss-Lobatto [16]. Gauss-Legendre quadrature can exactly integrate up to polynomials of order $2n-1$ using n quadrature points. The placement of the Gauss-Legendre nodes corresponds to the roots of the Legendre polynomials [17].

Gauss-Lobatto quadrature can exactly integrate up to polynomials of order $2n-3$ using n quadrature points. The endpoints of the interval of integration are always included as quadrature points. Because the integration power of Gauss-Lobatto is lower than that of Gauss-Legendre, one might assume that Gauss-Legendre is the optimal choice. However, in some cases, choosing Gauss-Lobatto nodes is advantageous [11].

As for interpolation, a natural choice would be to use evenly-spaced nodes throughout the element. However, one can show that the mass matrix obtained from evenly spaced interpolation nodes is ill-conditioned for higher-order methods, which is not optimal since the mass matrix needs to be inverted. Choosing a spacing similar to quadratures optimizes the conditioning of the mass matrix. Gauss-Legendre and Gauss-Lobatto, thus, are very good choices for interpolation [11]. Gauss-Lobatto is especially a good choice because it has nodes at the endpoints. This offers an advantage when dealing with surface integrals, where the solution does not need to be interpolated to the element faces as is required for Gauss-Legendre [18].

Chapter 3

Numerical Stability

3.1 Nonlinear Problems

The DG method faces challenges for nonlinear problems, as was alluded to in Chapter 1. This chapter delves deeper into the underlying theory of nonlinear stability. Gassner [19] and Ranocha [18] reported that the energy of the system:

$$\frac{1}{2}||u_h||^2 = \frac{1}{2}\int u_h^2 dx = \frac{1}{2}\sum_i w_i (u_h^i)^2, \tag{3.1}$$

is not formally bounded. Gassner [19] found the rate of change of kinetic energy of the standard DG scheme to be:

$$\frac{d}{dt}\left(\frac{1}{2}||u_h||^2\right) = -\oint (f^* - \frac{1}{3}f_h)u_h dx + \int f_h \frac{du_h}{dx} dx, \tag{3.2}$$

where the first term is a surface integral (evaluated at the boundaries of the cells), and the second term is a volume integral. For stability, the following condition is required:

$$\frac{d}{dt}\left(\frac{1}{2}||u_h||^2\right) \leq 0. \tag{3.3}$$

The first term in (3.2) can be ensured to be negative by choosing a particular numerical flux, f^*. However, it is not possible to control the second term; as the solution becomes discontinuous, $\frac{du_h}{dx}$ approaches infinity near the discontinuity, which causes the energy of the system to increase, resulting in nonlinear instability.

Gassner [19] demonstrated that through collocation of the interpolation and integration nodes and splitting the flux term, stability can be guaranteed. Although Ranocha [18] extended this to schemes that lack collocation, the schemes used in this thesis are collocated for simplicity.

3.2 Collocation

Typically, the nodes used for integration are at different positions from the nodes used for interpolation. This is especially the case when overintegrating the flux for nonlinear problems. Collocation requires the integration and interpolation nodes to be at the same location, as shown in fig. 2.5. This simplifies the problem, but, more importantly, allows one to draw parallels between DG and FD. There has been extensive research conducted on stability for FD schemes [20], so it would be possible to extend such concepts to DG. One key result of collocation is the diagonalization of the mass matrix:

$$\underline{\underline{M}}^k_{ij} \approx \sum_{m=1}^{N} w_m \ell_i^k(x_m)\ell_j^k(x_m) = \sum_{m=1}^{N} w_m \delta_{im}\delta_{jm} = w_i\delta_{ij}. \tag{3.4}$$

Because the mass matrix now only contains the quadrature weights, it can be used for integration. For instance, one can integrate two functions u and v on a domain as follows:

$$\int uvdx = \sum_{m=1}^{N} w_m u_m v_m = \underline{u}^T \underline{\underline{M}}\ \underline{u} \tag{3.5}$$

An additional relation involving the mass and differentiation matrices is obtained:

$$(\underline{\underline{M}}\ \underline{\underline{D}}) + (\underline{\underline{M}}\ \underline{\underline{D}})^T = \underline{\underline{B}}, \tag{3.6}$$

where $\underline{\underline{B}} = diag([-1\ 0 \dots 0\ 1])$ is the boundary operator. This is essentially the discretized version of integration by parts, known as summation by parts. In order to see this more clearly, we can pre- and post-multiply (3.6) by the vectors $\underline{u}^T$ and $\underline{v}$:

$$\underline{u}^T(\underline{\underline{M}}\ \underline{\underline{D}})\underline{v} + \underline{u}^T(\underline{\underline{M}}\ \underline{\underline{D}})^T\underline{v} = \underline{u}^T\underline{\underline{B}}\underline{v} \tag{3.7}$$

$$\underline{u}^T\underline{\underline{M}}\ (\underline{\underline{D}}\underline{v}) + (\underline{\underline{D}}\ \underline{u})^T(\underline{\underline{M}})\underline{v} = \underline{u}^T\underline{\underline{B}}\underline{v} \tag{3.8}$$

$$\int u\frac{dv}{dx}dx + \int \frac{du}{dx}vdx = uv\Big|_{x_1}^{x_p} \tag{3.9}$$

These properties can be taken advantage of in order to demonstrate nonlinear stability [19,

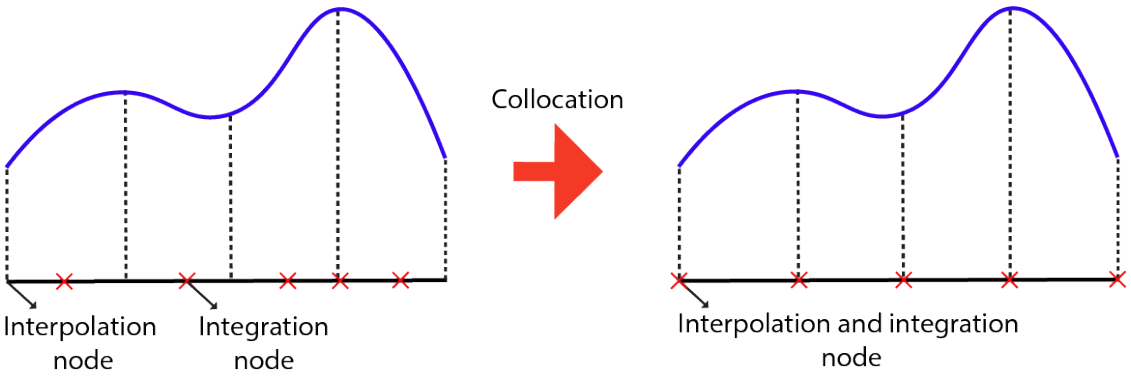


Figure 3.1: The nodes at which the solution is interpolated and integrated can be different. Collocation requires the two nodes to coincide.

21].

3.3 Flux Splitting and Burgers' Equation

In this subsection, we first present the concept of flux splitting and then address its importance to the nonlinear stability of a scheme. For Burgers' equation, flux splitting is done through the product rule:

$$\frac{\partial(\frac{u^2}{2})}{\partial x} = \alpha\frac{\partial(\frac{u^2}{2})}{\partial x} + (1-\alpha)u\frac{\partial u}{\partial x}. \tag{3.10}$$

Through the choice of $\alpha = \frac{2}{3}$, nonlinear stability can be demonstrated [19]. We urge the reader to consult [19] for a detailed proof of nonlinear stability for DG type schemes. The PDE can then be written as:

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0. \tag{3.11}$$

Discretizing (3.11) and writing the terms in operator form gives:

$$\underline{\underline{M}}\ \dot{\underline{u}} + \frac{2}{3}\underline{\underline{M}}\ \underline{\underline{D}}\ \underline{f} + \frac{1}{3}(\underline{\underline{U}}\ \underline{\underline{M}}\ \underline{\underline{D}}\ \underline{u}) = -\underline{\underline{B}}(\underline{f}^* - \underline{f}), \tag{3.12}$$

where $\underline{\underline{U}} = diag(\underline{u})$. We can now solve for $\dot{\underline{u}}$:

$$\dot{\underline{u}} = -\frac{2}{3}\underline{\underline{D}}\ \underline{f} - \frac{1}{3}(\underline{\underline{U}}\ \underline{\underline{D}}\ \underline{u}) - \underline{\underline{M}}^{-1}\ \underline{\underline{B}}(\underline{f}^* - \underline{f}), \tag{3.13}$$

and integrate the obtained system of ODEs in time using an ODE solver.

Calculating the energy of the new system, one arrives at:

$$\frac{d}{dt}\left(\frac{1}{2}||u_h||^2\right) = -\oint(f^* - \frac{1}{3}f_h)u_h dx. \tag{3.14}$$

Comparing this with (3.2), we notice that the second term has been eliminated. This means that one can now choose the numerical flux f^* in such a way that the energy of the system is always bounded. Therefore, nonlinear stability would be achieved. A common numerical flux which satisfies this criterion is the local Lax-Friedrichs flux:

$$f^*_{LLF} = \frac{1}{2}(\frac{u_L^2}{2} + \frac{u_R^2}{2}) - \frac{1}{2}\max(|u_L|, |u_R|)(u_L - u_R), \tag{3.15}$$

where u_L, u_R are the solution values at the left and right side of the boundary, respectively.

3.4 Extension to Euler's Equations

The governing equations of fluid mechanics are the Navier-Stokes equations, which relate various properties of fluids, such as pressure, velocity, viscosity, and energy, to each other. In the inviscid limit, the Navier-Stokes equations reduce to Euler's equations of fluid mechanics:

$$\frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{v} = 0 \tag{3.16}$$

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} + \frac{\nabla p}{\rho} = 0 \tag{3.17}$$

$$\frac{\partial e}{\partial t} + \mathbf{v} \cdot \nabla e + \frac{p}{\rho} \nabla \cdot \mathbf{v} = 0, \tag{3.18}$$

where $\mathbf{v} = (u, v, w)$ is the velocity vector, ρ the density, p the total gas pressure, and $e = \rho\tilde{u} + \frac{1}{2}\mathbf{v}^2$ the energy per unit mass, with $\tilde{u}$ being the internal energy. The equation of state usually used for the gas is $p = (\gamma - 1)\rho\tilde{u}$, where γ is the heat capacity ratio. (3.16) can be rewritten in conservative form:

$$\partial_t \mathbf{U} + \nabla \cdot \mathbf{F} = 0, \tag{3.19}$$

where $\mathbf{U}$ and $\mathbf{F}$ are the state and flux vectors, respectively:

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho\mathbf{v} \\ \rho e \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \rho\mathbf{v} \\ \rho\mathbf{v} \otimes \mathbf{v} + p \\ (\rho e + p)\mathbf{v} \end{pmatrix}. \tag{3.20}$$

$\mathbf{F}$ is essentially a 5×3 matrix. One can expand $\mathbf{F}$ to examine each matrix entry:

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_x & \mathbf{F}_y & \mathbf{F}_z \end{pmatrix} = \begin{pmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p & \rho uv & \rho uw \\ \rho uv & \rho v^2 + p & \rho vw \\ \rho uw + p & \rho vw & \rho w^2 + p \\ (\rho e + p)u & (\rho e + p)v & (\rho e + p)w \end{pmatrix}. \tag{3.21}$$

(3.19) is what will be analyzed throughout this thesis, as it facilitates discretization through DG. (3.19) is a compaction of 5 coupled, nonlinear PDEs governing the behaviour of 5 unknowns. The equations are, thus, complicated to analyze and even more so to discretize and simulate numerically.

Standard DG simulations of Euler's equations can lead to instabilities when there are large variations in the solution (such as turbulence or shocks), which is similar to the behaviour encountered when studying Burgers' equation. Drawing parallels from Burgers' equation, an ansatz would be that the kinetic energy of the system is not formally bounded, which results in nonlinear stability. In the FD community, there is numerous evidence that splitting the flux terms using the chain rule can increase the nonlinear stability of the schemes [22, 23, 24, 25, 26]. For instance, Kennedy and Gruber [26] showed that splitting double products in the flux as:

$$(ab)_x = \frac{1}{2}\left((ab)_x + a_x b + a b_x\right), \tag{3.22}$$

and triple products as:

$$(abc)_x = \frac{1}{4}(abc)_x + \frac{1}{4}(a_x(bc) + b_x(ac) + c_x(ab)) + \frac{1}{4}(a(bc)_x + b(ac)_x + c(ab)_x), \tag{3.23}$$

improves stability. Applying this to $\mathbf{F}_x$ in (3.21) gives:

$$\mathbf{F}_x^{KG} = \begin{pmatrix} \frac{1}{2}((\rho u)_x + \rho(u)_x + u(\rho)_x) \\ \frac{1}{4}((\rho u^2)_x + \rho(u^2)_x + 2u(\rho u)_x + u^2(\rho)_x + 2\rho u(u)_x) + p_x \\ \frac{1}{4}((\rho uv)_x + \rho(uv)_x + u(\rho v)_x + v(\rho u)_x + uv(\rho)_x + \rho v(u)_x + \rho u(v)_x) \\ \frac{1}{4}((\rho uw)_x + \rho(uw)_x + u(\rho w)_x + w(\rho u)_x + uw(\rho)_x + \rho w(u)_x + \rho u(w)_x) \\ \frac{1}{2}((pu)_x + p(u)_x + u(p)_x) + \frac{1}{4}((\rho eu)_x + \rho(eu)_x + e(\rho u)_x + u(\rho e)_x + eu(\rho)_x + \rho u(e)_x + \rho e(u)_x) \end{pmatrix}. \tag{3.24}$$

Similar formulas can be developed for $\mathbf{F}_y$ and $\mathbf{F}_z$. Gassner [27] extended these ideas to DG by taking advantage of collocation. Before delving deeper, however, the formulation of DG in three dimensions needs to be discussed.

3.4.1 DG in Three Dimensions

DG can quite easily be extended to 3 dimensions when dealing with quad and hex elements. This is because higher dimensions will just be tensor products of the one-dimensional formulation of DG. The DG scheme will be developed for Euler's equations, where we consider a collocated scheme in strong form.

Let $l \in [1, 5]$ represent the component of the state vector $\mathbf{U}$. The grid is discretized into K elements, with each element having $(p+1)^3$ nodes for the solution. Let us consider an element $[-1, 1]^3$. We represent each component of the solution vector as a linear combination of Lagrange polynomials in three dimensions. For instance, for the first component, one gets:

$$U^1 = \rho(x, y, z, t) \approx \sum_{i,j,k=0}^{p} \rho_{ijk}(t)\ell_i(x)\ell_j(y)\ell_k(z), \tag{3.25}$$

where $\rho_{ijk}(t)$ is the nodal value of the solution at point (x_i, y_j, z_k). Because of the similarity to the one-dimensional case, (2.8) can be expanded to three dimensions, resulting in the following:

$$\frac{\partial \underline{U}}{\partial t} + \tilde{F}_x + \tilde{F}_y + \tilde{F}_z = 0, \tag{3.26}$$

where

$$(\tilde{F}_x)^l_{ijk} = \Big[(F_x)^{*,l}(1, y, z) - (F_x)^l_{pjk}\Big] - \Big[(F_x)^{*,l}(-1, y, z) - (F_x)^l_{0jk}\Big] + \sum_{m=0}^{p} D_{im}(F_x)^l_{mjk}, \tag{3.27}$$

$$(\tilde{F}_y)^l_{ijk} = \Big[(F_y)^{*,l}(x, 1, z) - (F_y)^l_{ipk}\Big] - \Big[(F_y)^{*,l}(x, -1, z) - (F_y)^l_{i0k}\Big] + \sum_{m=0}^{p} D_{jm}(F_y)^l_{imk}, \tag{3.28}$$

$$(\tilde{F}_z)^l_{ijk} = \Big[(F_z)^{*,l}(x, y, 1) - (F_z)^l_{ijp}\Big] - \Big[(F_z)^{*,l}(x, y, -1) - (F_z)^l_{ij0}\Big] + \sum_{m=0}^{p} D_{km}(F_z)^l_{ijm}. \tag{3.29}$$

The first two terms correspond to the boundaries and involve the numerical fluxes $\underline{F}^{*,l}$. The third term in each equation corresponds to the volume terms. Gassner [27] demonstrated that, using the formulation of (3.26) with a few modifications, one can achieve stability.

3.5 Two-Point Flux

The difficulty with the split formulation is the additional terms that need to be discretized. For instance, the fifth component of $\mathbf{F}_x^{KG}$ has 10 terms, all of which involve derivatives. As equations become more complex, the number of split formulations and the terms in the formulations increases exponentially. This poses a challenge when it comes to implementing split formulation schemes in code. Fortunately, Chan [28] found a way to remedy this issue by introducing a two-point flux. For instance, for Burgers' equation, one can substitute the flux $f = \frac{u^2}{2}$ with:

$$f_s(u_i, u_j) = \frac{1}{6}(u_i^2 + u_i u_j + u_j^2) \tag{3.30}$$

Gassner [27] also used this concept to simplify the implementation of the split formulation of Euler's equations. In particular, he derived the split formulations of single, double, and triple products (as given in (3.22) and (3.23)):

$$2\sum_{m=0}^{p} D_{im}\{\{a\}\}_{im} = (\underline{\underline{D}}\ \underline{a})_i \tag{3.31}$$

$$2\sum_{m=0}^{p} D_{im}\{\{a\}\}_{im}\{\{b\}\}_{im} = \frac{1}{2}(\underline{\underline{D}}\ \underline{a}\ \underline{b} + \underline{a}\ \underline{\underline{D}}\ \underline{b} + \underline{b}\ \underline{\underline{D}}\ \underline{a}) \tag{3.32}$$

$$2\sum_{m=0}^{p} D_{im}\{\{a\}\}_{im}\{\{b\}\}_{im}\{\{c\}\}_{im} = \frac{1}{4}(\underline{\underline{D}}\ \underline{a}\ \underline{b}\ \underline{c} + \underline{a}\ \underline{\underline{D}}\ \underline{b}\ \underline{c} + \underline{b}\ \underline{\underline{D}}\ \underline{a}\ \underline{c} + \underline{c}\ \underline{\underline{D}}\ \underline{a}\ \underline{b} \tag{3.33}$$

$$+ \underline{b}\ \underline{c}\ \underline{\underline{D}}\ \underline{a} + \underline{a}\ \underline{c}\ \underline{\underline{D}}\ \underline{b} + \underline{a}\ \underline{b}\ \underline{\underline{D}}\ \underline{c}), \tag{3.34}$$

where

$$\{\{a\}\}_{im} = \frac{1}{2}(a_i + a_m), \tag{3.35}$$

is the average. Thus, one can use a two-point flux for the three components of $\mathbf{F}$. In this case, the fluxes in (3.27) become:

$$(\tilde{F}_x)^l_{ijk} = \left[(F_x)^{*,l}(1,y,z) - (F_x)^l_{pjk}\right] - \left[(F_x)^{*,l}(-1,y,z) - (F_x)^l_{0jk}\right] + \sum_{m=0}^{p} D_{im}(F_x)^{\#,l}_{mjk}(U_{ijk}, U_{mjk}), \tag{3.36}$$

$$(\tilde{F}_y)^l_{ijk} = \left[(F_y)^{*,l}(x,1,z) - (F_y)^l_{ipk}\right] - \left[(F_y)^{*,l}(x,-1,z) - (F_y)^l_{i0k}\right] + \sum_{m=0}^{p} D_{jm}(F_y)^{\#,l}_{imk}(U_{ijk}, U_{imk}), \tag{3.37}$$

$$(\tilde{F}_z)^l_{ijk} = \left[(F_z)^{*,l}(x,y,1) - (F_z)^l_{ijp}\right] - \left[(F_z)^{*,l}(x,y,-1) - (F_z)^l_{ij0}\right] + \sum_{m=0}^{p} D_{km}(F_z)^{\#,l}_{ijm}(U_{ijk}, U_{ijm}), \tag{3.38}$$

where $\mathbf{F}^{\#}$ is a numerical two-point volume flux:

$$\mathbf{F}^{\#}_x(U_{ijk}, U_{mjk}) = \begin{pmatrix} \{\{\rho\}\}\{\{u\}\} \\ \{\{\rho\}\}\{\{u\}\}^2 + \{\{p\}\} \\ \{\{\rho\}\}\{\{u\}\}\{\{v\}\} \\ \{\{\rho\}\}\{\{u\}\}\{\{w\}\} \\ \{\{\rho\}\}\{\{u\}\}\{\{e\}\} + \{\{p\}\}\{\{u\}\} \end{pmatrix} \tag{3.39}$$

$$\mathbf{F}^{\#}_y(U_{ijk}, U_{imk}) = \begin{pmatrix} \{\{\rho\}\}\{\{v\}\} \\ \{\{\rho\}\}\{\{u\}\}\{\{v\}\} \\ \{\{\rho\}\}\{\{v\}\}^2 + \{\{p\}\} \\ \{\{\rho\}\}\{\{v\}\}\{\{w\}\} \\ \{\{\rho\}\}\{\{v\}\}\{\{e\}\} + \{\{p\}\}\{\{v\}\} \end{pmatrix} \tag{3.40}$$

$$\mathbf{F}^{\#}_z(U_{ijk}, U_{ijm}) = \begin{pmatrix} \{\{\rho\}\}\{\{w\}\} \\ \{\{\rho\}\}\{\{u\}\}\{\{w\}\} \\ \{\{\rho\}\}\{\{v\}\}\{\{w\}\} \\ \{\{\rho\}\}\{\{w\}\}^2 + \{\{p\}\} \\ \{\{\rho\}\}\{\{w\}\}\{\{e\}\} + \{\{p\}\}\{\{w\}\} \end{pmatrix}. \tag{3.41}$$

This is much simpler to implement numerically, as it allows us to avoid having to deal with multiple terms that stem from the product rule. The numerical flux F^* used is:

$$F^*(U_L, U_R) = F^{\#}(U_L, U_R) - \frac{1}{2}\lambda_{max}[U_L - U_R], \tag{3.42}$$

where λ_{max} is an approximation of the maximum of the wave speeds at the interface. (3.42) is similar to the local Lax-Friedrichs flux, except it uses the two-point volume flux as opposed to a simple solution average [27].

3.6 Implementation

The DG method was implemented in C++ using *dealii* [29], a finite element library that provides the triangulation, basis functions, and other tools necessary to set up a general finite element problem. The code is referred to as PHiLiP (Parallel High-Order Library for PDEs) and can be viewed on Github: `github.com/dougshidong/PHiLiP`. C++ was chosen due to its low-level nature, which results in faster computational times compared to high-level languages such as MATLAB and Python. In addition, C++ is object-oriented, which increases the opportunity of writing compact, easy to understand, yet sophisticated code.

PHiLiP contains the physics required to solve the following PDEs: linear advection, diffusion, convection-diffusion, Burgers', and Euler's. The elements used are lines, quadrilaterals, and hexahedra for one, two, and three dimensions, respectively. Furthermore, PHiLiP is equipped with a message passing interface (MPI) to allow for parallelized computation. This is a key feature as it allows one to run expensive simulations on supercomputers by breaking down the problem into smaller bits and feeding them to many processors simultaneously. In addition, the code implements DG in both the weak and strong form, though only the strong form is used in this thesis. Lastly, PHiLiP is equipped with *hp* adaptivity, which allows for the modification of the grid or the individual elements' polynomial approximation order throughout run-time to ensure that the solution is accurately represented over

the entire computational domain.

The object-oriented nature of C++ is fully utilized through the usage of class inheritance and polymorphism. This effective compartmentalization of the code allows for creation of cleaner software and is advantageous when it comes to debugging. The main classes in PHiLiP are `physics`, `DG`, `numerical_flux`, `ode_solver`, `linear_solver`, `parameters`, `postprocessor`, and `testing`. The purpose of some of the classes is explained below.

3.6.1 `physics`

This class contains the physics required to solve the PDEs. There is a base class, which contains the solution and functions such as `convective_flux`, which is fed to `DG`. If new equations need to be added to the code (such as the Navier-Stokes equations), we can simply create a new class that derives from the base `physics` class without directly interacting with the `DG` class. The derived class would then define the functions in the base class (such as `convective_flux`).

3.6.2 `DG`

This class contains the backbone of the DG scheme, and hosts two important functions: `assemble_cell_terms` and `assemble_face_terms`. `assemble_cell_terms` calculates the volume integrals in the cells and adds the contributions from the degrees of freedom in the cells to the right-hand-side of the system. Similarly, `assemble_face_terms` computes the surface integrals. The two functions depend on the formulation of the DG scheme (weak versus strong).

3.6.3 `numerical_flux`

This class only deals with the numerical flux and is responsible for passing information regarding the numerical flux to the `assemble_face_terms` function in `DG`. Similar to `physics`,

there is a base class which contains the minimal information. If we want to add a new numerical flux, we can create a new class which derives from the base class and re-defines the functions.

3.6.4 `ode_solver`

This class works with the output of the `DG` class. `DG` assembles a right-hand-side vector for a system of ODEs. `ode_solver` then integrates the system of ODEs in time. The class has the flexibility of implicit or explicit integration. The former would be implicit Euler, and the latter could be RK4 or explicit Euler. The implicit scheme requires the use of the `linear_solver` because one has to solve a nonlinear system of equations at each time step using Newton's method.

3.6.5 `parameters`

The `parameters` class contains all the input that will be fed to the program. The parameters are provided to the code through a *.prm* file. This file contains information such as the formulation of the scheme (weak or strong), the time integration method (explicit or implicit) and the type of PDE to be solved.

Chapter 4

Results and Conclusion

4.1 Burgers' Equation

To test the numerical stability of the split form, the following initial condition was used:

$$u(x,0) = \sin(\pi x) + 0.01, \tag{4.1}$$

on the grid $[0,2]$ with periodic boundary conditions. The solution is initially smooth, but a shock forms at $t \approx 0.3s$. In Chapter 1, it was seen that the standard DG scheme fails beyond that point. However, the split formulation should maintain stability. The solution at various times is plotted in fig. 4.1 for $p = 7$. Comparing this to fig. 1.3, we can see the improved stability, allowing the simulation to be run for nearly 10 times longer. Although stability is achieved, parts of the solution are meaningless; oscillations form in the vicinity of the shock due to the Gibbs phenomenon, and they are an artifact of fitting a high-order solution through a discontinuity. Furthermore, the energy of the system ($\frac{1}{2}||u_h||^2$) is plotted in fig. 4.2 as a function of time for $p = 3, 5, 7$. It can be seen that the energy is non-increasing at all times, as predicted mathematically. Lastly, the order of convergence of the scheme was tested. This is important because if a scheme does not provide optimal orders, there is likely an issue with the implementation that needs to be addressed. The order

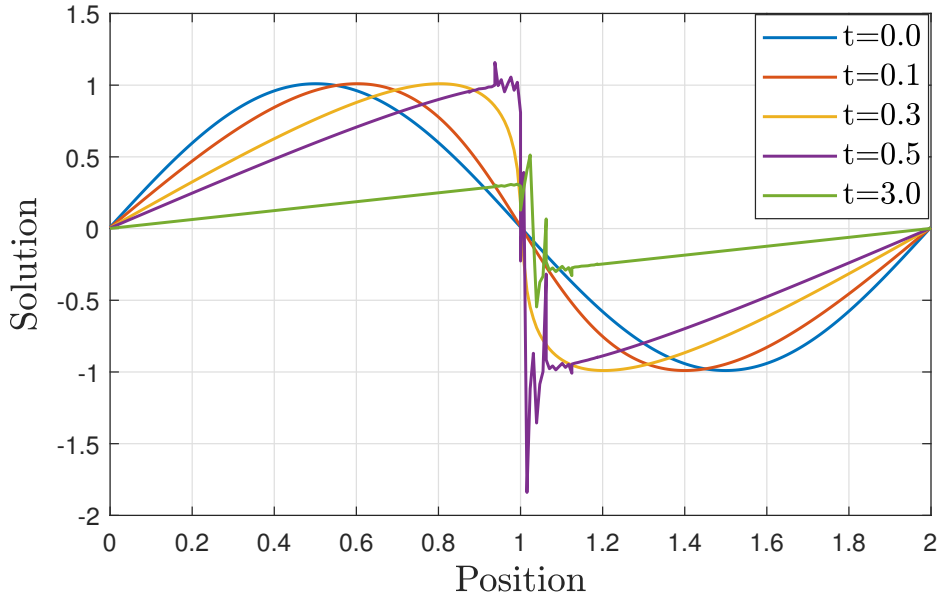


Figure 4.1: Depiction of the solution at various times with $p = 7$ and 32 elements. At around $t = 0.3s$, the shock forms. After that, oscillations form around the shock but the solution maintains stability as the shock propagates forward.

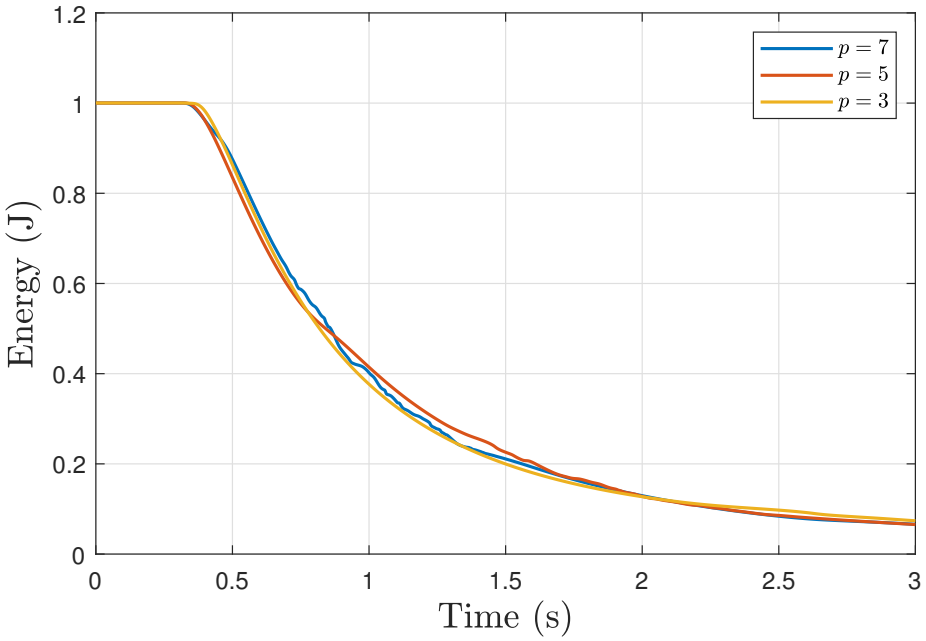


Figure 4.2: Energy $\frac{1}{2}||u_h||^2$ of the system is nonincreasing, as predicted mathematically.

of convergence of DG schemes is $O(\Delta x^{p+1})$, which is demonstrated in fig. 4.3 for various polynomial approximation orders. It is important to note that once the shock forms, the

order of convergence of the scheme decreases, so it is imperative that the convergence study is conducted for a smooth solution. Table 4.1 summarizes the convergence test for the three

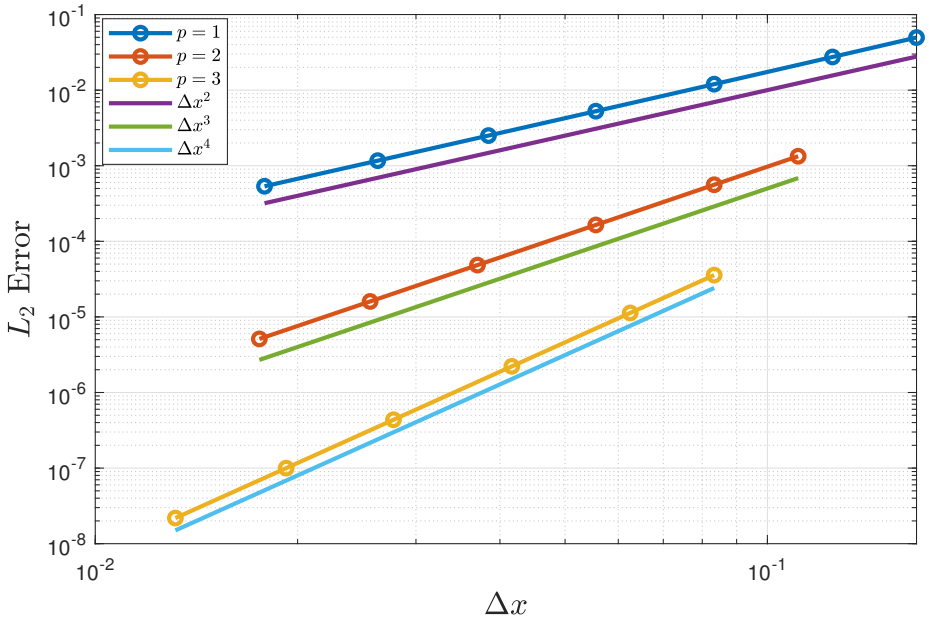


Figure 4.3: Burgers' equation gives the correct order of accuracy of $O(\Delta x^{p+1})$, where p is the polynomial approximation order.

finest grids and confirms the expected $p + 1$ order of accuracy.

p	Cells	Degrees of freedom	Δx	L_2 error	Order of accuracy
1	13	26	3.84×10^{-2}	2.50×10^{-3}	2.02
1	19	38	2.63×10^{-2}	1.17×10^{-3}	2.01
1	28	56	1.78×10^{-2}	5.35×10^{-4}	2.01
2	9	27	3.70×10^{-2}	4.84×10^{-5}	3.02
2	13	39	2.56×10^{-2}	1.60×10^{-5}	3.01
2	19	57	1.75×10^{-2}	5.11×10^{-6}	3.01
3	9	36	2.78×10^{-2}	4.37×10^{-7}	4.01
3	13	52	1.92×10^{-2}	9.98×10^{-8}	4.01
3	19	76	1.31×10^{-2}	2.18×10^{-8}	4.01

Table 4.1: Various parameters in the convergence study. The orders of accuracy match the expected $p + 1$ value.

4.2 Euler's Equations

The test case used was the inviscid Taylor Green Vortex (TGV). The initial condition given was:

$$\rho = 1, \tag{4.2}$$

$$u = \sin(x)\cos(y)\cos(z), \tag{4.3}$$

$$v = -\cos(x)\sin(y)\cos(z), \tag{4.4}$$

$$w = 0, \tag{4.5}$$

$$p = \frac{100}{\gamma} + \frac{1}{16}(\cos(2x)\cos(2z) + 2\cos(2y) + 2\cos(2x) + \cos(2y)\cos(2z)), \tag{4.6}$$

on the domain $[0, 2\pi]^3$ with periodic boundary conditions. This test case is suitable because the flow becomes turbulent within the first few seconds of run-time, as shown in fig. 4.4. This turbulence is artificial since Euler's equations are inviscid. Furthermore, due to the periodicity of the domain, the kinetic energy of the system is conserved. This makes the test case ideal because one can investigate the changes in kinetic energy and attribute the results to various aspects of the discretization. The kinetic energy is calculated by integrating over

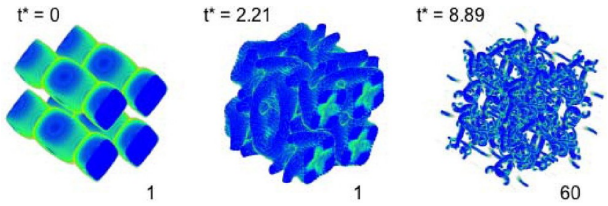


Figure 4.4: Solution of the TGV test case at various times, showing turbulence development. Figure taken from [30].

the domain as follows:

$$K = \int \frac{1}{2}\rho(u^2 + v^2 + w^2)dV = \int \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{2\rho}dV. \tag{4.7}$$

Because the TGV test cases for the Euler's equations were computationally expensive, they were submitted as MPI jobs on Beluga, which is a supercomputer under Calcul Quebec. Table 4.2 summarizes the information regarding some test cases that were run. The degrees of freedom were evenly divided between the processors, and only information about the solution at the surfaces of the elements needed to be exchanged between the cores to calculate the numerical fluxes.

Number of Processors	Memory/Processor	Run Time	Grid	p	Degrees of Freedom
400	1Gb	24 hours	8^3	2	69120
400	1Gb	24 hours	6^3	3	69120

Table 4.2: Various parameters used for the jobs submitted to Beluga. The simulations took 24 hours to run.

Figure 4.5 shows the kinetic energy of the system as a function of time for three different runs. The standard DG scheme, with $p = 2$ and on an 8^3 grid, only runs for $1.75s$, beyond which point the simulation crashes. This is because, as predicted, the kinetic energy of the system is not bounded. However, using the split form on the same grid with the same polynomial order, the simulation maintains stability and runs for $6.75s$. Although the kinetic energy is theoretically conserved, a decrease is observed in the figure. This is because a coarse grid is chosen with a low polynomial order, which means that the numerical dissipation

of the scheme is high. This dissipation can be decreased by increasing the polynomial approximation order. This is verified by the third curve in the figure, which corresponds to $p = 3$ and a 6^3 grid. The total number of degrees of freedom is maintained to be the same, but the dissipation is notably reduced by increasing the polynomial order.

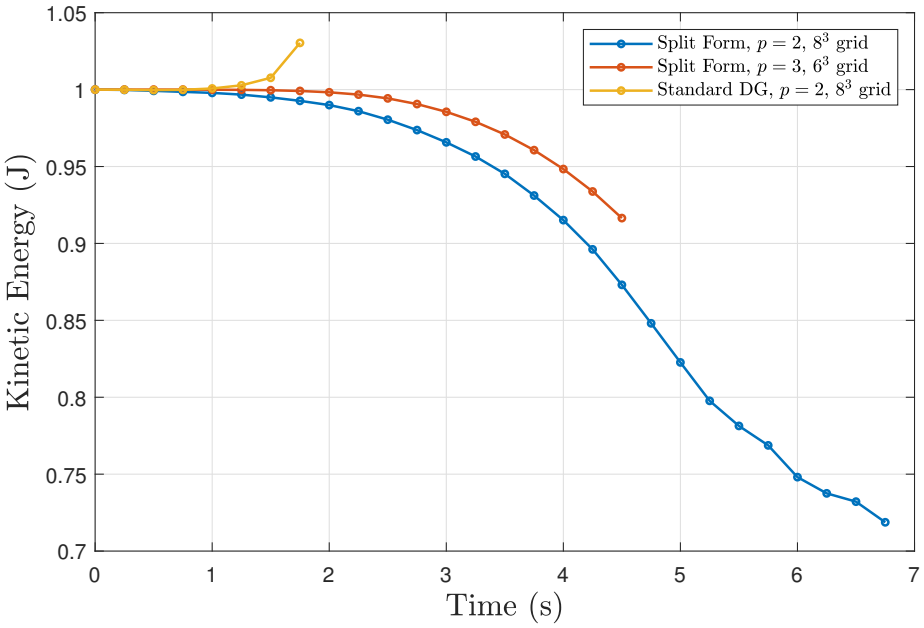


Figure 4.5: Kinetic energy plot of the TGV test case. The standard DG scheme blows up quickly, while the split form maintains stability and bounds the kinetic energy.

4.3 Conclusion and Future Work

This thesis investigated the nonlinear numerical stability of DG schemes applied to Burgers' equation and Euler's equations of fluid dynamics. It is well-known that high-order methods suffer from instabilities, preventing one from running simulations for the desired duration. Extensive research has been done on numerical stability in FD schemes. These concepts were extended to DG through the use of collocation and flux splitting. It was shown mathematically that flux splitting leads to a formulation for the rate of change of energy of Burgers' equations that would satisfy nonlinear stability through a particular choice of numerical flux

f^*. This concept was extended to Euler's equations as well.

To verify the mathematical concepts, the schemes were coded in C++ and were compared against the standard DG scheme. In all cases, the split formulations demonstrated increased robustness and stability compared to the standard schemes. Although analytically the formulations are exact, splitting the flux using the product rule leads to a scheme that is quite different numerically.

Future work would involve extending the concept of flux splitting to the Navier-Stokes equations. The Navier-Stokes equations are used to perform LES and study turbulent flow, and it is crucial to develop a scheme that is numerically stable and that enables running LES for long periods. Furthermore, this can be extended to the magnetohydrodynamics equations of plasma physics, as those equations also suffer from instabilities for high-order methods.

Bibliography

[1] William D McComb. "The physics of fluid turbulence". In: *Chemical physics* (1990).

[2] Juan Manzanero et al. "Insights on Aliasing Driven Instabilities for Advection Equations with Application to Gauss–Lobatto Discontinuous Galerkin Methods". In: *Journal of Scientific Computing* 75.3 (2018), pp. 1262–1281. ISSN: 1573-7691. DOI: 10.1007/s10915-017-0585-6. URL: https://doi.org/10.1007/s10915-017-0585-6.

[3] GA Blaisdell, ET Spyropoulos, and JH Qin. "The effect of the formulation of nonlinear terms on aliasing errors in spectral methods". In: *Applied Numerical Mathematics* 21.3 (1996), pp. 207–219.

[4] Joseph A Schetz and Allen E Fuhs. *Fundamentals of fluid mechanics*. John Wiley & Sons, 1999.

[5] Elewterio F Toro. *Godunov methods: Theory and applications*. Springer Science & Business Media, 2012.

[6] Per-Olof Persson and Jaime Peraire. "Sub-cell shock capturing for discontinuous Galerkin methods". In: *44th AIAA Aerospace Sciences Meeting and Exhibit*. 2006, p. 112.

[7] Paulien van Slingerland, Jennifer K Ryan, and Cornelis Vuik. "Position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering for improving discontinuous Galerkin solutions". In: *SIAM Journal on Scientific Computing* 33.2 (2011), pp. 802–825.

[8] Anne Burbeau, Pierre Sagaut, and Ch-H Bruneau. "A problem-independent limiter for high-order Runge–Kutta discontinuous Galerkin methods". In: *Journal of Computational Physics* 169.1 (2001), pp. 111–150.

[9] Garrett E Barter. *Shock capturing with PDE-based artificial viscosity for an adaptive, higher-order discontinuous Galerkin finite element method*. Tech. rep. MASSACHUSETTS INST OF TECH CAMBRIDGE DEPT OF AERONAUTICS and ASTRONAUTICS, 2008.

[10] Rodrigo Moura, Spencer Sherwin, and Joaquim Peiro. "On DG-based iLES approaches at very high Reynolds numbers". In: *Report, Research Gate* (2015).

[11] J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Texts in Applied Mathematics. Springer New York, 2007. ISBN: 9780387720654. URL: https://books.google.ca/books?id=APQkDOmwyksC.

[12] AG Kravchenko and Parviz Moin. "On the effect of numerical errors in large eddy simulations of turbulent flows". In: *Journal of computational physics* 131.2 (1997), pp. 310–322.

[13] Yiran Chen et al. "Large eddy simulation of impinging flames: Unsteady ignition and flame propagation". In: *Fuel* 255 (2019), p. 115734. ISSN: 0016-2361. DOI: https://doi.org/10.1016/j.fuel.2019.115734.

[14] B. Gustafsson, H.O. Kreiss, and J. Oliger. *Time-Dependent Problems and Difference Methods*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 2013. ISBN: 9781118548523. URL: https://books.google.ca/books?id=LaseAAAAQBAJ.

[15] Randall J LeVeque et al. *Finite volume methods for hyperbolic problems*. Vol. 31. Cambridge university press, 2002.

[16] Steven C Chapra and Raymond P Canale. *Numerical methods for engineers*. Vol. 2. Mcgraw-hill New York, 1998.

[17] Milton Abramowitz. "Handbook of mathematical functions". In: *National Bureau of Standards Applied Mathematics Series* 55 (1972).

[18] Hendrik Ranocha, Philipp Öffner, and Thomas Sonar. "Summation-by-parts operators for correction procedure via reconstruction". In: *Journal of Computational Physics* 311 (2016), 299–328. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2016.02.009. URL: http://dx.doi.org/10.1016/j.jcp.2016.02.009.

[19] Gregor J Gassner. "A skew-symmetric discontinuous Galerkin spectral element discretization and its relation to SBP-SAT finite difference methods". In: *SIAM Journal on Scientific Computing* 35.3 (2013), A1233–A1253.

[20] Cyril W Hirt. "Heuristic stability theory for finite-difference equations". In: *Journal of Computational Physics* 2.4 (1968), pp. 339–355.

[21] Gregor J Gassner. "A kinetic energy preserving nodal discontinuous Galerkin spectral element method". In: *International Journal for Numerical Methods in Fluids* 76.1 (2014), pp. 28–50.

[22] Mark H Carpenter et al. "Entropy Stable Spectral Collocation Schemes for the Navier–Stokes Equations: Discontinuous Interfaces". In: *SIAM Journal on Scientific Computing* 36.5 (2014), B835–B867.

[23] F Ducros et al. "High-order fluxes for conservative skew-symmetric-like schemes in structured meshes: application to compressible flows". In: *Journal of Computational Physics* 161.1 (2000), pp. 114–139.

[24] Antony Jameson. "Formulation of kinetic energy preserving conservative schemes for gas dynamics and direct numerical simulation of one-dimensional viscous compressible flow in a shock tube using entropy and kinetic energy preserving schemes". In: *Journal of Scientific Computing* 34.2 (2008), pp. 188–208.

[25] Sergio Pirozzoli. "Numerical methods for high-speed flows". In: *Annual review of fluid mechanics* 43 (2011), pp. 163–194.

[26] Christopher A. Kennedy and Andrea Gruber. "Reduced aliasing formulations of the convective terms within the Navier–Stokes equations for a compressible fluid". In: *Journal of Computational Physics* 227.3 (2008), pp. 1676 –1700. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2007.09.020. URL: http://www.sciencedirect.com/science/article/pii/S0021999107004251.

[27] Gregor J. Gassner, Andrew R. Winters, and David A. Kopriva. "Split form nodal discontinuous Galerkin schemes with summation-by-parts property for the compressible Euler equations". In: *Journal of Computational Physics* 327 (2016), 39–66. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2016.09.013. URL: http://dx.doi.org/10.1016/j.jcp.2016.09.013.

[28] Jesse Chan. "On discretely entropy conservative and entropy stable discontinuous Galerkin methods". In: *Journal of Computational Physics* 362 (2018), pp. 346–374.

[29] Wolfgang Bangerth, Ralf Hartmann, and Guido Kanschat. "deal. II—a general-purpose object-oriented finite element library". In: *ACM Transactions on Mathematical Software (TOMS)* 33.4 (2007), p. 24.

[30] Dimitris Drikakis et al. "Simulation of transition and turbulence decay in the Taylor–Green vortex". In: *Journal of Turbulence* 8 (2007), N20.