A DG HWENO Scheme for Hyberbolic Equations

by

Matthieu Serrano

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Febuary 2004

Author .....
Department of Aeronautics and Astronautics
January 30, 2004

Certified by .....
Jaime Peraire
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Edward M. Greitzer
H N Slater Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

A DG HWENO Scheme for Hyberbolic Equations

by

Matthieu Serrano

Submitted to the Department of Aeronautics and Astronautics
on January 30, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

In an effort to build a higher order discontinuous Galerkin (DG) finite element solver for the nonlinear Euler equations of gas dynamics, we develop a shock capturing scheme for hyperbolic equations. The Hermite Weighted Essentially Non-Oscillatory (HWENO) methodology introduced by Qiu [10, 14] is used as the starting point for the proposed limiter. We present a general approach for building a limiter for Runge-Kutta time marching schemes which reconstructs the higher order moments of troubled cells using only information of neighboring cells. This technique is used to develop a limiter in 1-D for $\mathbf{P}_2$ to $\mathbf{P}_5$ interpolants on non-uniform grids and in 2-D for $\mathbf{P}_2$ interpolants on triangular unstructured grids. Numerical results for this limiter are presented for Burgers equation.

Thesis Supervisor: Jaime Peraire
Title: Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank the many people which have helped me during my stay at MIT. First and foremost, I would first like to thank my advisor, Jaime Peraire, for his support and for giving me the freedom and the guidance that I needed throughout my research. I am also grateful to David Darmofal and Robert Haimes for their constant involvement in Project X. Also, this thesis would not have been possible without the help of the PX crew (Chris, James, Mike, Paul and Todd).

The ACDL has been a great place to work but also to meet people. The back room is always a nice place to go to harass people when something isn't working or just to take a break. Life in the lab would not have been the same without Garrett, Ryan, and of course, the French speaking crew (David, Hector, Joe and Amandine).

I would also like to express special thanks to those with whom I played squash and hockey. Although this resulted in many bruises and a lot of broken equipement, the change of atmosphere was always welcomed.

Of course, I must acknowledge my family for giving me a little taste of France through various emails and phone calls.

Finally, I would like to thank Greg, Denis, Benjamin, Matthieu and the many other people with whom I have shared memorable moments during my stay in Boston.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computational fluid dynamics (CFD) has now become a standard tool used by aeronautical engineers. While CFD has matured significantly over the past decades, the accuracy of these methods must be enhanced further. Considering the already very large computational costs of aerodynamic simulations, the increase of accuracy should be gained in a more efficient manner than through h-refinement techniques. As Fidkowski *et al.* [1] have presented, higher order methods present significant benefits when accuracy requirements are stringent.

In the context of aerodynamics, creating a higher order solver presents several challenging tasks. These include the creation of higher order grids and visualization techniques. From the solver perspective, the most significant barrier to the development of higher order solvers is obtaining efficient shock capturing capabilities.

For multi-dimensional cases, structured and unstructured grids may be used to discretize the computational domain. In 2D, the natural extension of 1D grids are structured grids which are based on rectangles. This can further be extended to 3D structured grids. Nevertheless for general geometries, unstructured grids are prefered. These unstructured grids are based mainly on triangles and on tetrahedra. Although the mathematical simplicity of structured cells render the computations slightly more efficient, the use of unstructed cells make the griding process much easier and can, in many cases, become automatic. From the user stand point, this means days of gridding instead of weeks or even months for complex 3D structures, while the increase in computational time is only marginal. Considering the evolution of the computational power available, this proves to be a very cost efficient

trade-off.

Whereas different methods exist that can provide higher order of accuracy, finite element discontinuous Galerkin (DG) methods present some advantages over traditional finite difference and finite volume methods. In DG methods, the coupling is achieved only through numerical flux functions evaluated on the edges of the cells making the implementation of higher order schemes much easier. The implementation of the boundary conditions is a good example of the advantage gained through the DG formulation. In finite difference and finite volume schemes, single sided stencils must be taken to compute the derivatives at the boundary. For very high order schemes, these stencils becomes very large and are difficult to implement.

In this context, a higher-order discontinuous Galerkin finite element solver for the Euler equations on unstructured grid was developped. Although efficient limiters exist for $\mathbb{P}_1$ schemes, no limiter has yet set a standard for higher-order schemes. Qiu *et al.* [9] have developped a new class of limiters for Runge-Kutta (RK) DG methods based on the Weighted Essentially Non-Oscillatory (WENO) finite volume techniques. A limiter requiring more compact stencils using higher order moments of neighboring cells based on Hermite WENO schemes [10, 14] was then developped.

In this thesis, the HWENO methodology is generalized to create a higher-order limiter. The framework developed by Qui *et al.* [10, 14] is modified by introducing a new method of computing linear weights. Using this method, we construct a limiter for $\mathbb{P}_2$ to $\mathbb{P}_5$ solutions in the one-dimensional case on non-uniform grids. In addition, we propose a two-dimensional $\mathbb{P}_2$ limiter for triangular-based unstructured grids using only the moments of neighboring cells. Results using the proposed limiter on a scalar two-dimensional Burgers equation are presented.

Chapter 2

Overview of Discontinous Galerkin Methods

2.1 Mathematical formulation of the problem

2.1.1 Model equations

In this thesis, first order hyperbolic equations will be discussed. Our model equation will therefore be:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathcal{F}(\mathbf{u}) = 0 \tag{2.1}$$

In this equation the state $\mathbf{u}$ is a vector, $\mathcal{F}$ is a flux and the physical space in which ∇ operates can be of one or two dimensions. The mathematical problem consists in solving the above equation given a set of boundary conditions. The state $\mathbf{u}$ is defined in a space of infinite dimension and therefore, the above equation can not be solved exactly in general.

Three equations have been used in the context of this thesis: Burgers equation in one and two dimensions and the two-dimensional Euler equations.

Burgers equation

Burgers equation is a scalar shock-generating equation. It has been used as the model equation for the development of the shock limiting procedure.

In the one-dimensional case, we have:

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = 0 \tag{2.2}$$

In 2D, we used the following extension of Burgers equation:

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} + \frac{1}{2}\frac{\partial u^2}{\partial y} = 0 \tag{2.3}$$

Two-dimensional Euler equations

The two-dimensional Euler equations of gas dynamics are given by equation (2.1) where $\mathbf{u}$ is the conservative state vector,

$$\mathbf{u} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix},$$

and $\mathcal{F} = (\mathbf{F}^x, \mathbf{F}^y)$ is the inviscid flux,

$$\mathbf{F}^x = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix}, \quad \mathbf{F}^y = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{pmatrix}.$$

where ρ is the density. u and v are the x and y coordinates of the velocity. The total enthalpy is given by $H = E + p/\rho$, where p and E are the pressure and the total energy. The equation of state for a perfect gas is given by

$$p = (\gamma - 1)\left[\rho E - \frac{1}{2}\rho\left(u^2 + v^2\right)\right].$$

2.1.2 Weak formulation

In a compact domain Ω delimited by $\partial\Omega$, let $\mathcal{V}(\Omega)$ be the Sobolev space of acceptable solutions for the problem. It's definition contains the regularity of the solution (*e.g.* how many times the solution can be differentiated) and the boundary conditions that must be verified. In order to derive the DG formulation, the problem must be reformulated in a weak form.

If $\mathbf{u} \in \mathcal{V}(\Omega)$:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathcal{F}(\mathbf{u}) = 0 \quad &\Rightarrow \quad \forall \mathbf{v} \in \mathcal{V}(\Omega),\ (\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathcal{F}(\mathbf{u})) \cdot \mathbf{v} = 0 \\ &\Rightarrow \quad \forall \mathbf{v} \in \mathcal{V}(\Omega),\ \int_\Omega \left((\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathcal{F}(\mathbf{u})) \cdot \mathbf{v} \right) d\mathbf{x} = 0 \end{aligned}$$

Through integration by parts, we write

$$\begin{aligned} \int_\Omega (\nabla \cdot \mathcal{F}(\mathbf{u})) \cdot \mathbf{v} d\mathbf{x} \quad &= \quad - \int_\Omega (\nabla \mathbf{v}) \cdot \mathcal{F}(\mathbf{u}) d\mathbf{x} \\ &\quad + \int_{\partial\Omega} \mathbf{v} \left(\mathcal{F}(\mathbf{u}) \cdot \hat{\mathbf{n}} \right) ds \end{aligned} \tag{2.4}$$

where $\hat{\mathbf{n}}$ is an outward-pointing normal of the domain and $\mathcal{F}(\mathbf{u}) \cdot \hat{\mathbf{n}}$ is the flux on the boundary $\partial\Omega$.

Finally, we write the weak formulation of the mathematical problem:

$$\forall \mathbf{v} \in \mathcal{V}(\Omega),\ \int_\Omega \left(\frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v} \right) d\mathbf{x} - \int_\Omega (\nabla \mathbf{v}) \cdot \mathcal{F}(\mathbf{u}) d\mathbf{x} + \int_{\partial\Omega} \mathbf{v} \left(\mathcal{F}(\mathbf{u}) \cdot \hat{\mathbf{n}} \right) ds = 0 \tag{2.5}$$

The weak formulation serves as the basis for numerical methods such as finite elements and DG methods in particular. It is well known that when shocks are present, weak solutions are not unique.

The physically relevant solution to the problem will be the weak solution verifying the following entropy inequality:

$$\frac{\partial U(\mathbf{u})}{\partial t} + \frac{\partial F(\mathbf{u})}{\partial x} \leq 0 \tag{2.6}$$

for any convex function $U(\mathbf{u})$ and consistent entropy flux $F(\mathbf{u})$ satisfying $\nabla F = \nabla U \cdot \nabla \mathcal{F}$. For Burgers equation, we use $U = \mathbf{u}^2$. For Euler's equation, U is a function of the physical entropy s.

2.2 Discontinuous Galerkin discretization

Numerical methods approximate the mathematical solution which belongs to a space of infinite dimensions with a solution with a large but finite number of degrees of freedom. A viable numerical method must verify that when the number of degrees of freedom increases, the numerical solution converges towards the mathematical solution.

The finite element methodology is one of such methods. It approximates the Sobolev space $\mathcal{V}(\Omega)$ with the subspace of finite dimensions $\mathcal{V}_h^p(\Omega)$. In this subspace, Ω is discretized into a grid for which h is a norm of the size of the cells. In each of these cells, the elements $\mathbf{v} \in \mathcal{V}_h^p(\Omega)$ are polynomials of degree p.

2.2.1 Grids

In the following discussion, we will use a subdivision T_h of the domain Ω such that $\bar{\Omega} = \bigcup_{\kappa \in T_h} \bar{\kappa}$. The cells κ will be intervals $I_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ in the one dimensional case and triangles in the two dimensional case.

2.2.2 Discontinuous Galerkin methods

We obtain the Galerkin formulation by solving the weak form of the mathematical problem, (2.5),in $\mathcal{V}_h^p(\Omega)$ instead of $\mathcal{V}(\Omega)$. Thus, the Galerkin formulation is: $\mathbf{u}_h \in \mathcal{V}_h^p(\Omega)$ is the solution to the problem if and only if:

$$\begin{aligned} \forall \mathbf{v}_h \in \mathcal{V}_h^p(\Omega),\ 0 &= \int_\Omega \left(\frac{\partial \mathbf{u}_h}{\partial t} \cdot \mathbf{v}_h \right) d\mathbf{x} \\ &- \int_\Omega (\nabla \mathbf{v}_h) \cdot \mathcal{F}(\mathbf{u}_h) d\mathbf{x} \\ &+ \int_{\partial\Omega} \mathbf{v}_h \left(\mathcal{F}(\mathbf{u}_h) \cdot \hat{\mathbf{n}} \right) ds \end{aligned} \tag{2.7}$$

Now let $\phi_j \in \mathbb{P}_p[\mathbf{x}]$ be a basis of $\mathcal{V}_h^p(\Omega)$. Then,

$$\forall \mathbf{v}_h \in \mathcal{V}_h^p(\Omega),\ \mathbf{v}_h = \sum_j \overline{\mathbf{v}}_j \phi_j$$

In particular, $\mathbf{u}_h$ may be written as:

$$\mathbf{u}_h(x,t) = \sum_j \overline{\mathbf{u}}_j(t) \phi_j(x)$$

The degrees of freedom of our problem will therefore be the vector $\mathbf{u}_h$ of entries $\overline{\mathbf{u}}_j$. We can also define the mass matrix $\mathcal{M}$ and the residual vector $\mathbf{R}$ such that:

$$\mathcal{M}_{i,j} = \int_\Omega \phi_i \cdot \phi_j d\mathbf{x} \tag{2.8}$$

$$\begin{aligned}\mathbf{R}_i(\mathbf{u}_h) &= -\int_\Omega (\nabla\phi_i)\cdot\mathcal{F}(\mathbf{u}_h)d\mathbf{x} \\ &\quad + \int_{\partial\Omega}\phi_i\left(\mathcal{F}(\mathbf{u}_h)\cdot\hat{\mathbf{n}}\right)ds \end{aligned} \tag{2.9}$$

Nothing has been said on the choice of the ϕ_j except that they form a basis. The basis will naturally be chosen such that the support of the various ϕ_j be as small as possible in terms of number of cells of the grid T_h. In Discontinuous Galerkin methods, ϕ_j are defined over each element and are discontinuous across elements. Thus, they are chosen to have a support of only one cell. In this case, the mass matrix can be written as a block diagonal matrix where each block corresponds to the mass matrix of a given cell. In (2.8) the integration over Ω is equivalent to integrating over the support of $\phi_i\cdot\phi_j$ which is either null or consists of a given cell κ. This characeristic does not depend on the order of the polynomials used as interpolants and is what makes the DG formulation well adapted for a higher order solver.

Let us now consider an element of our basis ϕ_i whose support is a given element κ. Then, the residual may be written as:

$$\begin{aligned}\mathbf{R}_i(\mathbf{u}_h) &= -\int_\kappa (\nabla\phi_i)\cdot\mathcal{F}(\mathbf{u}_h)d\mathbf{x} \\ &\quad + \int_{\partial\kappa\setminus\partial\Omega}\phi_i\mathcal{H}(\mathbf{u}_h^+,\mathbf{u}_h^-,\hat{\mathbf{n}})ds \\ &\quad + \int_{\partial\kappa\cap\partial\Omega}\phi_i\mathcal{H}_{\mathrm{bc}}(\mathbf{u}_h^+,\mathbf{u}_h^-,\hat{\mathbf{n}})ds \end{aligned} \tag{2.10}$$

where $\mathcal{H}(\mathbf{u}_h^+,\mathbf{u}_h^-,\hat{\mathbf{n}})$ and $\mathcal{H}_{\mathrm{bc}}(\mathbf{u}_h^+,\mathbf{u}_h^-,\hat{\mathbf{n}})$ are numerical flux functions related to $\mathcal{F}$ for interior and boundary edges, respectively. Also, the $()^+$ and $()^-$ notation indicates the trace value taken from the interior and exterior of the element. The numerical flux function used to evaluate the boundary flux on $\partial\Omega$ need not coincide with that used for the interior edges.

In DG methods, the coupling is achieved through the numerical flux function $\mathcal{H}(\mathbf{u}_h^+,\mathbf{u}_h^-,\hat{\mathbf{n}})$ on the edges of the cells. One important feature of these numerical fluxes is that in the case when $\mathbf{u}_h^+ = \mathbf{u}_h^-$ as would be the case when the solution is continuous, then:

$$\mathcal{H}(\mathbf{u}_h,\mathbf{u}_h,\hat{\mathbf{n}}) = \mathcal{F}(\mathbf{u}_h)\cdot\hat{\mathbf{n}} \tag{2.11}$$

In this respect, the numerical flux $\mathcal{H}$ can be seen as the flux $\mathcal{F}$ evaluated for an appro-

priate value between the trace values on both sides of the edge. This appropriate value may be in some cases the interior state $\mathbf{u}_h^+$, or the exterior state $\mathbf{u}_h^-$ or any value between these two, *e.g.* a value which is a local minimum or maximum of the flux $\mathcal{F}$.

The Discontinuous Galerkin formulation will be equivalent to: $\mathbf{u}_h \in \mathcal{V}_h^p(\Omega)$ is the solution to the problem if and only if:

$$\forall i, \quad \int_\Omega \left(\sum_j \frac{\partial \overline{\mathbf{u}}_j}{\partial t} \phi_j \right) \cdot \phi_i d\mathbf{x} + \mathbf{R}_i(\mathbf{u}_h) = 0 \tag{2.12}$$

This final formulation (2.12) is the Galerkin formulation of the problem and will naturately be rewritten in the matrix form:

$$\mathcal{M} \frac{\partial \overline{\mathbf{u}}_h}{\partial t} + \mathbf{R}(\mathbf{u}_h) = 0 \tag{2.13}$$

2.2.3 Entropy inequality

It can be shown that if the method presented above with a flux satisfying (2.11) converges, it will converge to a weak solution of the PDE. In order to garantee the convergence to the physically relevant solution, it is useful to be able to prove a cell entropy inequality.

Jiang *et al.* [3] have proven the cell entropy inequality for a class of high-order DG finite element methods. This proof shows a great advantage of the DG method over finite difference and finite volume schemes since it ensures that if the algorithm converges then it will converge to the entropy satisfying solution. The proof of the cell entropy inequality will be given in the semi-discrete form in one space dimension for the square entropy. In this case, we can rewrite (2.7) as, $\forall \mathbf{v} \in \mathcal{V}_h^p(\Omega)$:

$$\int_{I_j} \frac{\partial \mathbf{u}}{\partial t}(x,t)\mathbf{v}(x)dx + \mathcal{H}_{j+\frac{1}{2}}(t)\mathbf{v}(x_{j+\frac{1}{2}}^-) - \mathcal{H}_{j-\frac{1}{2}}(t)\mathbf{v}(x_{j-\frac{1}{2}}^+) - \int_{I_j} \mathcal{F}(\mathbf{u}(x,t))\frac{\partial \mathbf{v}}{\partial x}(x)dx = 0$$

where, $\mathcal{H}_{j+\frac{1}{2}}(t) = \mathcal{H}(\mathbf{u}(x_{j+\frac{1}{2}}^-, t), \mathbf{u}(x_{j-\frac{1}{2}}^+, t))$ is a Lipschitz continuous monotone flux, or more generally an E-flux as defined by Osher[4]: for all $\mathbf{u}$ between $\mathbf{u}^+$ and $\mathbf{u}^-$:

$$(\mathcal{H}(\mathbf{u}^-, \mathbf{u}^+) - \mathcal{F}(\mathbf{u}))(\mathbf{u}^+ - \mathbf{u}^-) \leq 0 \tag{2.14}$$

Then taking $\mathbf{v}(x) = \mathbf{u}(x,t)$ and defining:

$$\begin{aligned} g(\mathbf{u}) &= \int^{\mathbf{u}} \mathcal{F}(v)dv \\ \hat{F}_{j+\frac{1}{2}}(t) &= \mathcal{H}_{j+\frac{1}{2}}(t)\mathbf{u}(x^-_{j+\frac{1}{2}}) - g(\mathbf{u}(x^-_{j+\frac{1}{2}},t)) \end{aligned}$$

we have:

$$\int_{I_j} \frac{1}{2}\frac{\partial \mathbf{u}^2}{\partial t}dx + \hat{F}_{j+\frac{1}{2}}(t) - \hat{F}_{j-\frac{1}{2}}(t) + A_j(t) = 0$$

where:

$$A_j(t) = -\mathcal{H}_{j-\frac{1}{2}}(t)(\mathbf{u}(x^+_{j-\frac{1}{2}},t) - \mathbf{u}(x^-_{j+\frac{1}{2}},t)) + g(\mathbf{u}(x^+_{j-\frac{1}{2}},t)) - g(\mathbf{u}(x^-_{j+\frac{1}{2}},t))$$

Using the mean value theorem, there exists ξ between $\mathbf{u}(x^+_{j-\frac{1}{2}},t)$ and $\mathbf{u}(x^-_{j+\frac{1}{2}},t)$ such that:

$$A_j(t) = -\left(\mathcal{H}_{j-\frac{1}{2}}(t) - \mathcal{F}(\xi)\right)(\mathbf{u}(x^+_{j-\frac{1}{2}},t) - \mathbf{u}(x^-_{j+\frac{1}{2}},t)) \geq 0$$

The above inequality flows from the properties of the E-flux (2.14). Finally noticing that $\hat{F}_{j+\frac{1}{2}}$ is consistent with the entropy flux for the square entropy:

$$F(u) = \int^{\mathbf{u}} \mathcal{F}(v)vdv = \mathcal{F}(\mathbf{u})\mathbf{u} - \int^{\mathbf{u}} \mathcal{F}(v)dv = \mathcal{F}(\mathbf{u})\mathbf{u} - g(\mathbf{u}),$$

we have now proven the cell entropy inequality:

$$\int_{I_j} \frac{1}{2}\frac{\partial \mathbf{u}^2}{\partial t}dx + \hat{F}_{j+\frac{1}{2}}(t) - \hat{F}_{j-\frac{1}{2}}(t) \leq 0$$

In order for this property to hold, an E-flux must be chosen. For Burgers equation in 1D and 2D, we use a Godunov flux which satisfies (2.14). The 2D Euler flow solver uses the Roe-averaged flux function[5] with entropy fix. Although this flux is not an E-flux, for subsonic cases as is the case in this thesis, this is not a problem. Additional flux functions are described, for instance, in the review article by Cockburn and Shu [2].

2.2.4 Higher-order interpolants and geometry representation

In order to achieve the high level of accuracy that was our goal, high order interpolants are used. The basis chosen is then implemented on a reference element. Unstructured

grids are used in 2D in which case the reference element is a triangle. A mapping from the actual elements of the grid to the reference element must then be computed. In order to gain the full advantage of the higher order interpolants, the geometry of the domain must also be discretized using higher order elements. These elements have the same topological features as the reference elements but the edges in 2D are no longer defined by a first order polynomial but by a polynomial of higher order fitted to the geometry. A higher-order 2D grid generator was created.

Interpolants

Two bases were implemented: Lagrange interpolants using regularly spaced nodes and a hierarchical basis. When using Lagrange interpolants, the degrees of freedom are the value of the solution at the given Lagrange points. In this respect, Lagrange interpolants are well adapated for initialization of the flow but also for post-processing techniques. Nevertheless, for very high order, the position of the Lagrange points will have major consequences on the conditionning of the various matrices, in which case, equidistant Lagrange points are far from optimal [7]. Nevertheless, considering the orders implemented, the ease of implementation of the equidistant Lagrange basis was chosen. In 1D and 2D, orders $\mathbb{P}_0$ to $\mathbb{P}_5$ were implemented.

Motivated by the use of p-multigrid algorithms, a hierarchical basis was implemented making the restriction and prolongation operators much simpler. The basis chosen was the hierarchic basis proposed by Solin *et al.* [8] for which the edge functions coincide exactly with the Lobatto shape functions.

Higher-order geometry representation

Higher-order cells are created using the framework of equidistant Lagrange basis. A higher order element is parametrized by the coordinates of the various Lagrange points in the physical space. This is equivalent to describing a non-linear mapping from the curved triangle to the reference element.

2.2.5 Numerical integration

Numerical integrations are done through the use of Gauss quadrature rules. In a one-dimensional case, let x_g be the quadrature points in a reference element $[-1, 1]$ and w_g

their associated weights. Then, on the reference element, the integration of a function f is approximated by:

$$\int_{-1}^{1} f(x)dx \approx \sum_{g} w_g f(x_g)$$

Many quadrature rules exist for different orders of accuracy. Solin *et al.* [8] present the known and predicted minimum number of points for a given order of accuracy as well as the position and weights of the various Gauss points.

When integrating a function on a curved element, a non-linear mapping from the physical space to the reference space must be computed. This involves a Jacobian which, unlike the case of linear mappings, is not constant. Thus, when the function that is integrated is a non-polynomial function and/or the element edges are curved, integration is no longer exact and must be of sufficiently high order to preserve the order of accuracy of the scheme.

2.3 Time discretization

In Equation (2.13), a time derivative must be computed. If one is interested in steady state, this derivation may be ignored and set to zero. For unsteady computations, this is no longer the case and a discretization in time must be accomplished in order to compute this derivative numerically. Two classes of methods exist to perform time discretization. An explicit method is used for time accurate solutions and an implicit method is used for steady state solutions.

2.3.1 Explicit methods: RKDG methods

Explicit methods are cheaper than implicit methods when considering the computational time of a single iteration and can be used for time accurate solutions. Moreover, they use much less memory than implicit methods.

The simplest example of an explicit method is the forward Euler time marching scheme. Equation (2.13) is then modified to obtain:

$$\mathcal{M}\frac{1}{\Delta t}(\overline{\mathbf{u}}_h(t+\Delta t) - \overline{\mathbf{u}}_h(t)) + \mathbf{R}(\mathbf{u}_h(t)) = 0$$

For the sake of clarity, a new notation will now be introduced using n as the superscript

corresponding to the time level. Then, we can write:

$$\mathcal{M}\frac{1}{\Delta t}(\mathbf{u}^{n+1} - \mathbf{u}^n) + \mathbf{R}(\mathbf{u}^n) = 0$$
$$\mathbf{u}^{n+1} = \mathbf{u}^n - \Delta t \mathcal{M}^{-1} \cdot \mathbf{R}(\mathbf{u}^n)$$

Introducing $\mathbf{L}(\mathbf{u}^n) = -\mathcal{M}^{-1} \cdot \mathbf{R}(\mathbf{u}^n)$, we finally have:

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{L}(\mathbf{u}^n) \tag{2.15}$$

In this simple method, the inverse of the mass matrix is computed once and for all and at each iteration, the residual is computed and multiplied by the inverse of the matrix. Unfortunately, this method is only first order accurate in time. A class of higher order non-linearly stable explicit methods was devised in [6]. These Runge-Kutta (RK) methods offer stability and high order time accuracy. As an example, the third order version of a Runge-Kutta method is given by:

$$\begin{aligned}
\mathbf{u}^{(1)} &= \mathbf{u}^n + \Delta t \mathbf{L}(\mathbf{u}^n) \\
\mathbf{u}^{(2)} &= \frac{3}{4}\mathbf{u}^n + \frac{1}{4}\mathbf{u}^{(1)} + \frac{1}{4}\Delta t \mathbf{L}(\mathbf{u}^{(1)}) \\
\mathbf{u}^{n+1} &= \frac{1}{3}\mathbf{u}^n + \frac{2}{3}\mathbf{u}^{(2)} + \frac{2}{3}\Delta t \mathbf{L}(\mathbf{u}^{(2)})
\end{aligned} \tag{2.16}$$

These methods offer the advantage of being easy to code and cheap per iteration. Moreover, these time discretization can be applied on a wide variety of spatial discretization since they inherit the stability of the first order time stepping [6]. Nevertheless, these methods require small time steps for stability. Thus, if a steady-state solution is needed, this method may not be well suited.

In order to preserve the total variation properties of the scheme, a non-linear shock limiter $\Lambda\Pi$ may need to be applied to these Runge-Kutta schemes[2]. In such a case, the above scheme is modified as:

$$\begin{aligned}
\mathbf{u}^{(1)} &= \Lambda\Pi\left(\mathbf{u}^n + \Delta t \mathbf{L}(\mathbf{u}^n)\right) \\
\mathbf{u}^{(2)} &= \Lambda\Pi\left(\frac{3}{4}\mathbf{u}^n + \frac{1}{4}\mathbf{u}^{(1)} + \frac{1}{4}\Delta t \mathbf{L}(\mathbf{u}^{(1)})\right)
\end{aligned} \tag{2.17}$$

$$\mathbf{u}^{n+1} = \Lambda\Pi\left(\frac{1}{3}\mathbf{u}^n + \frac{2}{3}\mathbf{u}^{(2)} + \frac{2}{3}\Delta t\mathbf{L}(\mathbf{u}^{(2)})\right)$$

When applying this RK time marching scheme to a DG problem, it should be noticed that computing the residual in a given cell requires the degrees of freedom of this cell and all its neighbours and eventually some boundary conditions if the cell is on a boundary. In order to compute the next state, the inverse of the mass matrix is needed. As was previously noted, the mass matrix for a DG scheme is block diagonal with each block corresponding to a cell. Thus, this inversion can be done at the element level.

2.3.2 Implicit methods: steady state solvers

Implicit methods are designed to allow for larger timesteps and in particular are well suited to find the steady state solution to our problem (2.13). In this case, the system that we are trying to solve is $\mathbf{R}(\mathbf{u}) = 0$. If we assume that $\mathbf{u}^n$ is close to the solution, doing a Taylor expansion, we can write:

$$\mathbf{R}(\mathbf{u}) = 0 = \mathbf{R}(\mathbf{u}^n) + \frac{\partial \mathbf{R}}{\partial \mathbf{u}} \cdot (\mathbf{u} - \mathbf{u}^n) + \mathcal{O}(||\mathbf{u} - \mathbf{u}^n||^2)$$

This formulation is the starting point for implicit iterative solvers where a generic iteration can be written as,

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \mathbf{P}^{-1} \cdot \mathbf{R}(\mathbf{u}^n) \tag{2.18}$$

where the preconditioner, $\mathbf{P}$, is an appoximation to $\frac{\partial \mathbf{R}}{\partial \mathbf{u}}$. As was previously stated the residual in a given cell depends on the degrees of freedom of this cell and of its neighbors. Thus, the jacobian $\frac{\partial \mathbf{R}}{\partial \mathbf{u}}$ is a sparse matrix that can be split into blocks, each block corresponding to the derivatives of the residual in a cell i with respect to the degrees of freedom of cell j. Thus, each block on the diagonal is non zero. A few other blocks will be non zero. Due to the number of degrees of freedom in typical problems, the inversion of the jacobian becomes very expensive. Thus, although possible for small problems, the inversion of the full jacobian is impractical. Another option, is to keep only the block diagonals and invert only the local jacobians. It is also possible to associate elements together and to invert the jacobian among these groups of elements. This is equivalent to reorganizing the blocks of

the full jacobian and inverting larger diagonal blocks then previously. For strongly convective systems, one possible reorganization is creating lines of elements in the direction of the convection. The approximated jacobian is denoted by $\mathbf{M}$.

Finally, the unsteady term may be added to the approximate solution in order to obtain the final preconditionner:

$$\mathbf{P} = \mathbf{M} + \frac{1}{\Delta t}\mathcal{M}$$

Mathematically, this addition makes the system more diagonally dominated and hence better conditionned for the iterative method. Physically, the time term is used to alleviate transients during the solution process. As the solution begins to converge, Δt can be increased towards infinity.

Implicit methods can converge much faster than explicit methods to a steady solution. Nevertheless, they require much more memory usage and compute time per iteration since the jacobian must be stored and large matrices must be inverted.

The solvers implemented in the 2D fluid solver include various implicit solvers ranging from a block point implicit solver to p-multigrid schemes using a line solver. These solvers are presented in more detail by Fidkowski *et al.* [1].

Chapter 3

Limiting

Although the higher order polynomials used in DG methods can give accurate representations of the flow in smooth regions, they are not well suited to represent shocks. As a result of this, overshoots and oscillations are created numerically and then propagate to the rest of the domain degrading the overall accuracy of the solution. This motivates the use of shock limiters.

The class of limiters we are considering are computationally expensive and it is therefore desirable to apply them only when necessary. For this purpose, a method should be used to detect the potentially troubled cells where the solution may require limiting. Since this approach will unavoidably flag cells where the solution is smooth, the limiter ought to be such that when applied to smooth flow regions, it does not degrade the accuracy of the underlying scheme.

Existing limiting techniques in the DG context as well as in finite volume will be presented. Mature finite volume techniques have served as guideline for the development of DG limiters. In particular, the WENO and HWENO limiters[9, 10] will serve as the starting point for the proposed limiter.

3.1 Existing limiting techniques

3.1.1 RKDG limiters

A number of limiting techniques have been developped for RKDG methods. For $\mathbb{P}_1$ solutions in 1D and 2D, the standard limiting technique is the minmod limiter described

by Cockburn and Shu [2]. We present this technique in the 1D case. Let $\mathbf{u}_{0,j}$ and $\mathbf{u}_{1,j}$ be the average value and the first moment of $\mathbf{u}$ in a given cell $I_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$. For a given $x \in I_j$, we have:

$$\mathbf{u}(x) = \mathbf{u}_{0,j} + \frac{2}{\Delta x}(x - x_j)\mathbf{u}_{1,j}$$

The limiting process consists in reducing the gradient of $\mathbf{u}$ in troubled cells while maintaining conservation of mass. Thus, $\mathbf{u}_{1,j}$ is replaced by the limited value $\tilde{\mathbf{u}}_{1,j}$:

$$\tilde{\mathbf{u}}_{1,j} = \text{minmod}(\mathbf{u}_{1,j}, \mathbf{u}_{0,j+1} - \mathbf{u}_{0,j}, \mathbf{u}_{0,j} - \mathbf{u}_{0,j-1}) \tag{3.1}$$

where,

$$\text{minmod}(a_1, \ldots, a_n) = \begin{cases} s \ \min_i |a_i|, & \text{if } s = sign(a_1) = \ldots = sign(a_m) \\ 0, & \text{otherwise.} \end{cases}$$

The resulting RKDG scheme with the above limiter can be proven to be total variation diminishing (TVD). It can be rendered total variation bounded (TVB) by modifying the minmod function so as not to degrade accuracy at local extrema [2]. A problem dependant constant M is introduced and the TVB corrected minmod function $\tilde{m}$ is defined as:

$$\tilde{m}(a_1, \ldots, a_n) = \begin{cases} a_1, & \text{if } |a_1| \leq M(\Delta x)^2 \\ \text{minmod}(a_1, \ldots, a_n), & \text{otherwise.} \end{cases}$$

Burbeau *et al.* [11] developed a series of RKDG limiters improving on this TVB method by removing the need for any problem dependant constant.

3.1.2 Applications of WENO methodology to RKDG: WENO and HWENO limiters

Qiu *et al.* [9] have proposed a RKDG limiter inspired by the WENO methodology. This limiter is based on creating various reconstruction polynomials and weighing them in order to reconstruct certain moments that were flagged as requiring limiting. Linear weights must first be computed and then scaled using smoothness indicators to obtain the non-linear weights which will be used for the reconstruction of the moments. This reconstruction which is the core of the WENO schemes will be discussed in more detail in the following section.

This method is capable of achieving higher order limiting which produces non oscillatory solutions whitout degrading the accuracy in the smooth regions. While the results for this limiter are promising, it presents a major drawback inherent to standard finite volume methods: the stencils on which the reconstruction polynomials are defined are very large and some of the advantages of the local discontinous galerkin formulation are therefore lost.

In order to solve this, Qiu *et al.* [10] developped the Hermite WENO methodology. Instead of generating reconstruction polynomials using only cell averages over a wide stencil, this method uses a much more compact stencil by using the high order moments of the neighboring cells. This will serve as the basis for the proposed limiter.

3.1.3 ENO and WENO methods

The Essentially Non-Oscillatory (ENO) [12] method was developped in the context of finite volume and finite difference methods. Using the moments available in neighboring cells a number of reconstruction polynomials P_i are defined. The ENO method chooses the least oscillatory of these reconstruction polynomials and evluates the solution at the Gauss points. In such a case, the order of accuracy of the scheme is determined by the order of the reconstruction polynomials.

The WENO methodology[12] uses reconstruction polynomials but uses weighted sums of all the reconstruction polynomials instead of chosing the least oscillatory one. This method proves to be more robust then the original ENO formulation. Linear weights are chosen such that they match the moments of a reconstruction polynomial of higher order Q. For each P_i, a smoothness indicator is computed. The linear weights are then modified based on the value of the smoothness indicators.

Reconstruction polynomials

We define a stencil as a set of moments which are are associated to a polynomial and a cell. In the case of Hermite schemes where higher-order moments are available in each cell, two different stencils may contain the same cells but utilize different moments. Each reconstruction polynomial P_i is defined over such a stencil S_i. The higher order polynomial Q is then defined over $S = \bigcup_i S_i$. Over each stencil S_i, the moments of P_i will be the same as the moments of Q. In standard WENO methods, Q is chosen to match exactly the moments of the solution. This is possible if the cardinal of the stencil S matches the

number of degrees of freedom of Q. In 1D, this will be the case for the proposed limiter. On the other hand, one can also impose an order of Q such that the number of degrees of freedom is smaller than the cardinal of S as will be the case for the proposed limiter in 2D.

Linear weights

A weighted sum of the reconstruction polynomials is computed. The weights are chosen in such a way as to gain accuracy in the cell of interest. In the case of the WENO and HWENO reconstructions proposed by Qui *et al.* [9, 10], this means that the moment which is being reconstructed given by the weighted sum of the P_i must be equal to that of Q. Thus, the weighted sum of lower order polynomials yields the same moment as a single polynomial of much higher order.

These weights are called linear weights. They depend on the geometry but not on the solution. When the solution is smooth, the use of these linear weights is perfectly equivalent to the use of Q which represents the best approximation for the solution. Nevertheless, when the solution is oscillatory or when a shock is present, Q will not be a good approximation to the solution. In this case, the WENO and HWENO methods modify the weights by using smoothness indicators.

The existence of linear weights is a problem since they are defined through a system of linear equations involving more equations than unknowns. In this case, there could be a unique solution, an infinite amount of solutions or no solution at all. While previous work shows that these linear weights exist and are unique under certain conditions, a method should be developped in order to robustely obtain linear weights in the general context.

3.1.4 Smoothness indicators

The smoothness indicator developed by Jiang *et al.* [13] is used in order to give a measure of the variation of the reconstruction polynomials. These indicators will then be used to scale the linear weights.

In 1D, the smooothness indicators are defined as:

$$\beta_i = \sum_{l=j}^{n+j-1} \int_I \Delta x^{2l-1} \left(\frac{\partial^l}{\partial x} P_i(x) \right)^2 dx$$

Where j is the moment that is being reconstructed, n is the order of the polynomial

which is being limited, I is the troubled cell, and Δx its length.

The generalization to multidimension of the above expression was done by Hu *et al.* [15]:

$$\beta_i = \sum_{j \leq |\alpha| \leq n+j-1} \int_I |\Delta|^{2|\alpha|-1} \left(D^\alpha P_i(\underline{x})\right)^2 d\Omega$$

Here j is the moment that is being reconstructed, n is the order of the polynomial we are limiting, I is the troubled cell, and $|\Delta|$ its volume. The derivative operator is denoted by D and α is a multi-index parameter, *e.g.* in 2D, when $\alpha = (1,2)$, then $|\alpha| = 3$ and $D^\alpha p(x,y) = \partial p^3(x,y)/\partial x \partial y^2$.

3.1.5 Moment reconstruction

Using these smoothness indicators, the linear weights are modified to generate the non-linear weights. This scaling of the linear weights ought to satisfy several constraints. In the case of a cell with a smooth solution, the smoothness indicators of all the reconstruction polynomials will be of the same order of magnitude. In this case, we would like to use the linear weights which are known to provide a higher degree of accuracy. But, when a shock or oscillations are present, the reconstruction polynomials reconstructed from data at either side of the shock will have a large smoothness indicator. In this case, we would like to take predominently the least oscillatory polynomials.

A scaling method which verifies these constraints is:

$$\begin{aligned} \omega_i &= \frac{\overline{\omega_i}}{\sum_k \overline{\omega_k}} \\ \overline{\omega_i} &= \frac{w_i}{(\epsilon + \beta_i)^2} \end{aligned}$$

The reconstructed moment of the solution is then the sum of the moments of the reconstruction polynomials weighted using the non-linear weights ω_i.

3.1.6 Detecting troubled cells

In the WENO and HWENO based limiter [9, 10], the TVB minmod limiter described for the one dimensional case in section 3.1.1 is used to detect troubled cells. A cell is determined to be troubled if one of the minmod functions is activate, *i.e.* does not return the first argument.

In 1D for $\mathbb{P}_1$ solutions, we note that (3.1) can be rewritten as:

$$\tilde{\mathbf{u}}^-_{j+\frac{1}{2}} = \mathbf{u}_{0,j} + \text{minmod}(\mathbf{u}^-_{j+\frac{1}{2}} - \mathbf{u}_{0,j}, \mathbf{u}_{0,j+1} - \mathbf{u}_{0,j}, \mathbf{u}_{0,j} - \mathbf{u}_{0,j-1})$$
$$\tilde{\mathbf{u}}^+_{j-\frac{1}{2}} = \mathbf{u}_{0,j} - \text{minmod}(\mathbf{u}_{0,j} - \mathbf{u}^+_{j-\frac{1}{2}}, \mathbf{u}_{0,j+1} - \mathbf{u}_{0,j}, \mathbf{u}_{0,j} - \mathbf{u}_{0,j-1})$$

The above formulation no longer depends on the order of the interpolants. Thus, for all orders, we compute the two minmod functions and flag a cell as troubled if one of these minmods is activated.

In our implementation, we use the above scheme to detect troubled cells where the minmod function is replaced by the TVB corrected minmod function to reduce the number of wrongfully flagged cells.

For the 2D implementation, the minmod limiter[2, 11, 13, 14] is used to detect troubled cells. In this case, the cell is flagged as being troubled if one of three minmod functions, one on each face of the triangle, is activated.

3.2 Proposed limiter

The WENO and especially the HWENO methods offer some very appealing features for the development of a limiter for higher order DG methods. A limiter for DG methods capable of limiting higher order solutions for multi-dimensional problems will be developped from the HWENO methodology developped by Qiu *et al.* [10]. Qiu proposed a limiter for $\mathbb{P}_2$ RKDG on one-dimensional uniform meshes. Qiu *et al.* [14] then continued the development of this method creating a 2D $\mathbb{P}_2$ limiter for uniform structured meshes.

Based on the methods introduced by Qiu *et al.* , an RKDG limiter for 1D non-uniform meshes and 2D unstructured meshes was developed. This method was designed in order to be generalisable to higher dimensions and to any order. Moreover, the method was to be created using only the moments available in the immediate neighboring cells.

3.2.1 An LP-based method to compute linear weights

As was previously noted, the linear weights are defined through a linear problem which in general has more equations than unknowns but which is typically underconstrained. In order to solve this problem, a linear programming (LP) based method to compute the linear

weights has been developped.

We wish to reconstruct a certain moment M_0. The nm available moments for this reconstruction are:

$$\underline{M} = \begin{bmatrix} M_1 \\ \vdots \\ M_{nm} \end{bmatrix}$$

We express the np reconstruction polynomials P_i, $i \in [0, np-1]$ and Q in a hierarchical basis ψ_j. Let k and nq be the number of degrees of freedom of P_i and Q. The order of Q is higher than the order of the P_i, thus: $k < nq$. Moreover, we have $nq \leq nm$.

$$\begin{aligned} \forall i \in [0, np-1], \qquad P_i(\underline{x}) &= \sum_{j=0}^{k-1} \alpha_i^j \ \psi_j(\underline{x}) \\ Q(\underline{x}) &= \sum_{j=0}^{nq-1} \alpha^j \ \psi_j(\underline{x}) \end{aligned}$$

For each polynomial, we define:

$$\forall i \in [0, np-1], \qquad \underline{\alpha_i} = \begin{bmatrix} \alpha_i^0 \\ \vdots \\ \alpha_i^{k-1} \end{bmatrix} \text{ and } \underline{\alpha} = \begin{bmatrix} \alpha^0 \\ \vdots \\ \alpha^{nq-1} \end{bmatrix}$$

The moments of the polynomials are linear functions of the coefficients $\underline{\alpha_i}$ and $\underline{\alpha}$. Then, we define the matrices $\underline{\underline{C}}$ and $\underline{\underline{E}}$ such that:

$$\begin{aligned} \underline{\underline{C}} \cdot \underline{\alpha} &= \begin{bmatrix} M_1^Q \\ \vdots \\ M_{nm}^Q \end{bmatrix} \qquad (3.2) \\ \forall i \in [0, np-1], \qquad \underline{\underline{E}} \cdot \underline{\alpha_i} &= \begin{bmatrix} M_1^i \\ \vdots \\ M_{nm}^i \end{bmatrix} \end{aligned}$$

It should be noted that these matrices depend only on the geometry and on the polynomial basis used to define the moments. Moreover, since $k < nq$, $\underline{\underline{E}}$ is a subset of $\underline{\underline{C}}$. If we define $\underline{\underline{I'}} = \underline{\underline{I_{nq,k}}}$, the matrix of size $nq \times k$ with ones on the diagonal and zeros everywhere

else, we have the relationship:

$$\underline{\underline{E}} = \underline{\underline{C}} \cdot \underline{\underline{I'}}$$

Let S_i be the stencil over which the reconstruction polynomial P_i is defined. For each i, we define the matrix $\underline{\underline{T_i}}$ such that $\underline{\underline{T_i}} \cdot \underline{M}$ is a vector of size k containing the moments which define P_i, *i.e.* the moments which define the stencil S_i. We now impose that these k moments match the moments of the higher-order polynomial Q. Thus, we impose that:

$$\forall i \in [0, np-1], \qquad \underline{\underline{T_i}} \cdot \underline{\underline{E}} \cdot \underline{\alpha_i} = \underline{\underline{T_i}} \cdot \underline{\underline{C}} \cdot \underline{\alpha} \tag{3.3}$$

The matrix $\underline{\underline{T_i}} \cdot \underline{\underline{E}}$ is always invertible since the moments are linearly independant. If we define the matrix $\underline{\underline{D_i}}$,

$$\forall i \in [0, np-1], \qquad \underline{\underline{D_i}} = (\underline{\underline{T_i}} \cdot \underline{\underline{E}})^{-1} \cdot \underline{\underline{T_i}}$$

then equation (3.3) may be rewritten as:

$$\forall i \in [0, np-1], \qquad \underline{\alpha_i} = \underline{\underline{D_i}} \cdot \underline{\underline{C}} \cdot \underline{\alpha}$$

We wish to find linear weights such that the moment M_0 that we are reconstructing from the weighted sum of P_i is the same as the moment M_0^Q of Q. Thus, the problem for the linear weights,

$$\forall \underline{M}, \qquad \sum_{i=0}^{np-1} w_i M_0^i = M_0^Q \tag{3.4}$$

where w_i are the linear weights which we are trying to determine. We define $\underline{w}$ the vector containing these linear weights:

$$\underline{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_{np-1} \end{bmatrix}$$

The moment M_0^Q is a linear function of the coefficients $\underline{\alpha}$ of Q. Moreover, the P_i are expressed in the basis formed by the k first elements of the basis of Q. Thus, there exists a vector $\underline{m_0}$ such that:

$$\underline{m_0} \cdot \underline{\alpha} = M_0^Q \tag{3.5}$$

$$\forall i \in [0, np-1], \qquad \underline{m_0} \cdot \underline{\underline{I'}} \cdot \underline{\alpha_i} \;=\; M_0^i$$

Let $\underline{e_i}$ be the unit vector in the i-th direction, *i.e.* the vector whose coordinates are all zero except the i-th coordinate which is equal to one. We define $\underline{\underline{A}}$ such that:

$$\underline{\underline{A}} \;=\; \sum_{i=0}^{np-1} \underline{e_i} \otimes \left(\underline{m_0} \cdot \underline{\underline{I'}} \cdot \underline{\underline{D_i}} \cdot \underline{\underline{C}} \right)$$

We then have,

$$\underline{w} \cdot \underline{\underline{A}} \cdot \underline{\alpha} \;=\; \sum_{i=0}^{np-1} w_i M_0^i$$

Therefore, the problem (3.4) can be written in the following form:

$$\forall \underline{\alpha}, \qquad \underline{w} \cdot \underline{\underline{A}} \cdot \underline{\alpha} \;=\; \underline{m_0} \cdot \underline{\alpha} \tag{3.6}$$

or equivalently:

$$\underline{w} \cdot \underline{\underline{A}} \;=\; \underline{m_0} \tag{3.7}$$

We further wish to ensure that the linear weights are positive. Thus, we formulate the following LP problem:

$$\text{minimize } f \text{ where } \begin{cases} f = max_i(w_i) \\ w_i \geq 0 \qquad i = 0..np-1 \\ \underline{w} \cdot \underline{\underline{A}} = \underline{m_0} \end{cases} \tag{3.8}$$

General remarks on this method

Considering the formulation (3.6), one should notice that if Q is of the same order as the reconstruction polynomials (*i.e.* for all $j \geq k, \alpha^j = 0$), then for all i, we have $P_i = Q$. Thus, the first k equations imposed in (3.6) are equivalent to $\sum_i w_i = 1$.

Moreover, it should be noted that the feasibility of the problem (3.8) is linked only to the choice of the stencils defining the reconstruction polynomials and to the order of Q. In the following sections, we present in one and two dimensions these choices which make the

problem feasible.

Finally, it should be noted that the linear weights are determined seperately for each moment. It would be interesting to investigate whether a set of reconstruction polynomials could be used for the reconstruction of multiple moments. Although the number of polynomials might increase, overall, the number of smoothness indicators which will need to be computed at each iterations may be reduced.

3.2.2 Generating the limiter

Computing the linear weights

Since the linear weights only depend on the geometry, they are computed once and stored. In order to apply the method described above, we need to specify:

1. The order of the solution we are considering, the moment we are reconstructing and the stencils which are available to reconstruct the different P_i. This determines:
   (a) The nm available moments: $\underline{M}$
   (b) The order k of the polynomials P_i
2. The order of Q such that $nq \leq nm$
3. The moments that we use to construct the various P_i: this defines the matrices $\underline{\underline{T_i}}$.
4. The geometry and the basis that we use to construct the moments. This determines $\underline{\underline{C}}$, $\underline{\underline{E}}$ and $\underline{m_0}$

Obtaining a limiter

The method described above gives a new method for computing the linear weights which is the vital component required in order to generalize Shu's HWENO limiter. Once the linear weights are computed, we still need to compute at every iteration:

1. The smoothness indicators β_i for all the P_i
2. The non linear weights based on the smoothness indicators
3. The moment we are reconstructing for all the P_i
4. Finally, the reconstructed moment

3.3 One-dimensional implementation

A one dimensional limiter for scalar solutions of order $\mathbb{P}_2$ to $\mathbb{P}_5$ using the previously described method was implemented. In this section, we review the various steps that are required in order to build the limiter. In order to detect troubled cells, the modified minmod detector described in section 3.1.6 is used. When reconstructing for a $\mathbb{P}_n$ solution in a troubled cell, the n highest moments will need to be reconstructed retaining only the zero-th moment in order to keep a conservative scheme. This will be done in n steps. For each step, the steps described in section 3.2.2 will be carried out.

3.3.1 Discretization

For the representation of the reconstruction polynomials, we use a hierarchical basis ψ_k. These ψ_k will be evaluated over the whole stencil S. Thus, in order keep the matrices required to compute the limiter well conditioned, we pick a basis of orthogonal polynomials, namely the Legendre polynomials, over the whole stencil, *i.e.* over three cells: the one where the reconstruction is done and its two neighbours, the left and right cells. In addition, each cell can be mapped to the reference element $[-1, 1]$. We define the mapping functions over these three cells – f, f_l and f_r – such that for each cell, $f(x)$, $f_l(x)$ and $f_r(x)$ are the coordinates in the space where the Legendre polynomials are defined. Thus, for a given $x \in [-1, 1]$ in the reference space associated with the different cells, the value of the reconstruction polynomials can be computed with:

$$\psi_k(f(x)) \qquad \psi_k(f_l(x)) \qquad \psi_k(f_r(x))$$

If λ_l and λ_r denote the length of the left and right cells divided by the length of the cell that is being reconstructed, the mapping functions are given by:

$$\begin{aligned} f(x) &= \frac{x}{3.5} \\ f_l(x) &= \frac{x\lambda_l - (1+\lambda_l)}{3.5} \\ f_r(x) &= \frac{x\lambda_r + (1+\lambda_r)}{3.5} \end{aligned}$$

The factor of 3.5 was chosen in order to approximately spread the Legendre polynomials

from the typical $[-1,1]$ segment over the three cells in the context of a non-uniform mesh.

For implementation purposes, the Legendre polynomials and all its derivatives which need to be computed for the smoothness indicators are calculated using two recursive properties of the Legendre polynomials:

$$\begin{aligned}(i+1)\psi_{i+1}(x) &= (2i+1)x\psi_i(x) - i\psi_{i-1}(x)\\ (1-x^2)\psi_i'(x) &= -ix\psi_i(x) + i\psi_{i-1}(x)\end{aligned}$$

Discretization of the solution

Let ϕ_k be the polynomial basis used to discretize the solution. Let M_k be the moment associated with ϕ_k of a function $u(x)$. If ϕ_k is an orthogonal basis defined on the reference element $[-1,1]$, then the moments are related to the degrees of freedom by the relationship:

$$\begin{aligned}M_k &= \frac{\int_{-1}^{1} u(x)\phi_k(x)dx}{\int_{-1}^{1}\phi_k(x)\phi_k(x)dx}\\ u(x) &= \sum_k M_k\phi_k(x)\end{aligned}$$

In our implementation, the following orthonormalized Legendre basis was chosen:

$$\phi_0 = \sqrt{\frac{1}{2}} \qquad \phi_1 = \sqrt{\frac{3}{2}}x \qquad \phi_2 = \sqrt{\frac{45}{8}}\left(x^2 - \frac{1}{3}\right) \quad \dots$$

Whereas one could chose to prolongate the ϕ_k over the whole stencil to represent the reconstruction polynomials, the conditionning of the matrices computed in the limiter becomes poor, motivating the use of the two bases ϕ_k and ψ_k.

3.3.2 Computing the linear weights

The LP method descrined in section 3.2.1 was applied to the 1D problem. In this case, following the method described by Qiu [10], we need to define distinct sets of reconstruction polynomials for the reconstruction of the different moments in order to obtains a feasible problem. Thus, when reconstructing the j-th moment ($j \in [1,n]$) of our $\mathbb{P}_n$ solution, the order of the polynomials P_i is $n+j-1$, *i.e.* $k = n+j$.

Each of the np polynomial P_i must match the j first moments (the zero-th to the $(j-1)$-th moment) of the cell of interest. The stencil S_i must also contain n other moments taken

from the neighboring elements under the constraints that if a certain moment must be matched, then all the lower order moments must also be matched.

This final constraint gives the number of available moments. In 1D, this means that the n first moments (the zero-th to the $(n-1)$-th moment) of the left and right neighbors need to be considered. Thus, the number of moments is $nm = 2n + j - 1$. In this case, we can pick Q optimally so that $nq = nm$.

The moments that are matched by each P_i can therefore be written explicitly. Each P_i must match the i first moments in the right cell, the $n - i$ first moments in the left cell and the j first moments in the cell that is being reconstructed. Therefore, there are $n + 1$ different P_i for all j (*i.e.* $np = n + 1$). Moreover, from j to $j + 1$, we only add one constraint to each of the P_i and to Q: the constraint to match the j-th moment in the cell that is being reconstructed.

From (3.2) and (3.5), it should be noted that computing the rows of $\underline{\underline{C}}$ or the vector $\underline{m_0}$ is equivalent to computing a vector $\underline{m}$ such that $\underline{m} \cdot \underline{\alpha}$ is a moment. Therefore, given the choice of basis previously described, computing $\underline{\underline{C}}$ and $\underline{m_0}$ is equivalent to computing for all j and k and for $g(x) = f(x)$, $f_l(x)$ or $f_r(x)$:

$$\frac{\int_{-1}^{1} \psi_j(g(x))\phi_k(x)dx}{\int_{-1}^{1} \phi_k(x)\phi_k(x)dx}$$

This analysis shows that the various matrices $\underline{\underline{T_i}}$ and $\underline{\underline{C}}$ must be computed only for the final step and the matrices at each step are a subset of these matrices. The method for obtaining $\underline{\underline{T_i}}$ is straightforward from the description of the moments that are matched by P_i.

Finally, it should be noticed that the choice of stencil previously described for the 1D case makes the problem (3.7) have a unique solution. Moreover, even on non-uniform meshes, the linear weights are always positive. Therefore, the problem (3.8) has only one feasible solution which is the solution of (3.7).

3.3.3 Computing the smoothness indicators

The smooothness indicators are defined as [13]:

$$\begin{aligned}\beta_i &= \sum_{l=j}^{n+j-1} \int_I \Delta x^{2l-1} \left(\frac{\partial^l}{\partial x} P_i(x)\right)^2 dx \\ &= \sum_{l=j}^{n+j-1} 2^{2l--1} \int_{-1}^{1} \left(\sum_{m=0}^{n+j-1} \left(\alpha_i^m \frac{\partial^l \psi_m}{\partial x^l}(f(x))\right)\right)^2 dx \qquad i = 0 \ldots np-1\end{aligned}$$

where j is the moment that is being reconstructed, n is the order of the polynomials we are limiting, I is the troubled cell, and Δx its length.

It is clear that the indicators β_i can be expressed as quadratic forms of $\underline{\alpha_i}$. We define the symetric matrix $\underline{\underline{Q_j}}$ such that:

$$\beta_i = \underline{\alpha_i} \cdot \underline{\underline{Q_j}} \cdot \underline{\alpha_i}$$

Computing the off diagonal terms in terms of diagonal terms of these matrices can be done using the normal methods for quadratic forms. Using matrix notation, we have:

$$2\underline{\underline{Q_j}}(i_1, i_2) = \underline{\underline{Q_j}}(i_1 + i_2, i_1 + i_2) - \underline{\underline{Q_j}}(i_1, i_1) - \underline{\underline{Q_j}}(i_2, i_2)$$

Using the above method, we obtain the final result:

$$\begin{aligned}\underline{\underline{Q_j}}(i_1, i_2) &= 0 \qquad \text{if } min(i_1, i_2) < j \\ \underline{\underline{Q_j}}(i_1, i_2) &= \sum_{l=j}^{min(i_1,i_2)} 2^{2l-1} \int_{-1}^{1} \frac{\partial^l \psi_{i_1}}{\partial x^l}(f(x)) \frac{\partial^l \psi_{i_2}}{\partial x^l}(f(x))\, dx \qquad \text{if } (i_1, i_2) \geq j\end{aligned}$$

The matrices $\underline{\underline{Q_j}}$ are computed using the previously stated recursive relation verified by the Legendre polynomials and using an adequate quadrature rule. It should be noted that the values of $\underline{\alpha_i}$ are never computed when applying the limiter. Therefore, the matrices $\underline{Q_j}$ are not the matrices that are stored. Instead, for each element, we will store the matrices $\underline{\underline{Q_j^i}}$ which depend on the geometry only.

$$\underline{\underline{Q_j^i}} = \underline{\underline{D_i}}^T \cdot \underline{\underline{Q_j}} \cdot \underline{\underline{D_i}}$$

The smoothness indicators will then be computed for the reconstruction of the j-th moment as:

$$\beta_i = \underline{M} \cdot \underline{\underline{Q_j^i}} \cdot \underline{M}$$

3.3.4 Summary of the 1D case

All the components of the 1D limiter have been presented. As was previously noted, the major obstacle for the generalization of the method proposed by Qiu *et al.* [10] is obtaining linear weights. In the 1D case, positive linear weights have always been obtained for all orders and on non-uniform meshes. In fact, the problem was setup such that the linear weights were defined uniquely. This is highly linked with the fact that for all orders, the optimal Q, *i.e.* with $nq = nm$, could be chosen.

The linear weights for $\mathbb{P}_2$ to $\mathbb{P}_5$ on uniform meshes are given in Appendix B.

3.4 Two-dimensional implementation

A two dimensional limiter for scalar $\mathbb{P}_2$ solutions on unstructured grids was developed. The two dimensional detector described in section 3.1.6 is used to detect troubled cells. A set of 18 stencils and associated reconstruction polynomials was required in order to make the problem feasible and is presented in appendix A. The polynomials were used in order to reconstruct all the moments of the troubled cells except the 0-th moment which is preserved for conservation.

3.4.1 Discretization

The reference element is defined as the triangle T, such that $(x, y) \in T$ is given by $0 \leq x \leq 1 - y$ and $0 \leq y \leq 1 - x$. Similarly to the 1D case, the reconstruction polynomials will be defined on the union of the stencils S. As the limiter was developed in order to use only the three immediate neighbors, the reconstruction polynomials will be defined over an area which covers approximately the upper right half of the square $[-1, 1]^2$ as can be seen in Figure 3-1. Thus, similarly to the one-dimensional case, another basis is used for the representation of the reconstruction polynomials to assure that the matrices that will be computed are well conditioned. Let Ψ_k be this basis which is defined using the

one-dimensional Legendre polynomials ψ_k:

$$\begin{aligned}
\Psi_0(x,y) &= \psi_0(x)\psi_0(y) \\
\Psi_1(x,y) &= \psi_1(x)\psi_0(y) \\
\Psi_2(x,y) &= \psi_0(x)\psi_1(y) \\
\Psi_3(x,y) &= \psi_2(x)\psi_0(y) \\
\Psi_4(x,y) &= \psi_1(x)\psi_1(y) \\
\Psi_5(x,y) &= \psi_0(x)\psi_2(y)
\end{aligned}$$

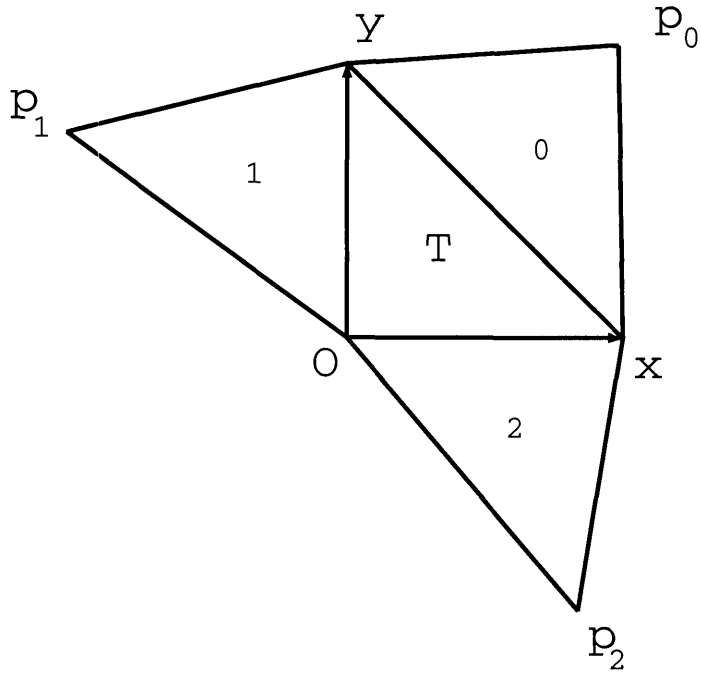

Figure 3-1: Reference Element T with it's three neighbors

As in the 1D case, for each of the four cells in the geometry represented in the above figure, a mapping function must be defined. In order to define this mapping, we denote by p_0, p_1 and p_2 the coordinates in the reference space of the vertices of all three neighbors which do not belong to the reference element T, as is shown in figure 3-1. In the case of a

uniform mesh, these vertices have the following coordinates:

$$p_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad p_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad p_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Using these coordinates, we now define the mapping functions f, f_0, f_1, f_2:

$$\begin{aligned} f(x,y) &= \begin{bmatrix} x \\ y \end{bmatrix} \\ f_0(x,y) &= \begin{bmatrix} x + p_{0,x}\ y \\ 1 - x + (p_{0,y} - 1)y \end{bmatrix} \\ f_1(x,y) &= \begin{bmatrix} p_{1,x}\ y \\ x + p_{1,y}\ y \end{bmatrix} \\ f_2(x,y) &= \begin{bmatrix} 1 - x + (p_{2,x} - 1)y \\ p_{2,y}\ y \end{bmatrix} \end{aligned}$$

The solution is represented using an orthonormal hierarchical basis ϕ_k. This basis is obtained by orthonormalization of the canonical basis using the scalar product defined over a reference element T:

$$(u, v) = \int_T u(x)v(x)d\Omega$$

We number linearly the elements of our basis. Thus, $\phi_0, \phi_1, \phi_2, \phi_3, \phi_4, \phi_5$ are the orthonormalized versions of $1, x, y, x^2, xy, y^2$.

3.4.2 Obtaining the linear weights

The LP method described in section 3.2.1 was used in order to obtain linear weights. All the moments of the neighboring cells were used, therefore $nm = 19$. Whereas in the 1D case, the polynomial Q was optimal in the sense that the $nq = nm$, a smaller order of Q was used in the two dimensional problem. For the $\mathbb{P}_2$ limiter, Q was chosen to be a $\mathbb{P}_3$ thus $nq = 10$. In this case, the weighted sum of P_i will match the moments of all polynomials of order up to 3.

Unfortunately, if the positivity constraint in (3.8) is imposed too strictly, the problem may become infeasible due to numerical round off in the computation of $\underline{\underline{A}}$. Thus, for all

the moments that need to be reconstructed, the problem (3.8) is modified in the following manner:

$$\text{minimize } f \text{ where } \begin{cases} f = max_i(w_i) + p\sum_i \epsilon_i \\ w_i \geq 0,\ \epsilon_i \geq 0 \qquad i = 0..np-1 \\ (\underline{w} - \underline{\epsilon}) \cdot \underline{\underline{A}} = \underline{m_0} \end{cases} \tag{3.9}$$

where p is a penalty factor which was set to 10^3 in the implementation. In this case, the true solution to the original problem (3.7) will be $\underline{w} - \underline{\epsilon}$. Nevertheless, the optimization procedure will as much as possible set $\underline{\epsilon}$ to zero. If this is not possible, the negative value $w_i - \epsilon_i$ is evaluated and set to zero if above a given threshold of 10^{-3}.

This modification was to make the procedure to obtain linear weights more robust to numerical errors. The threshold of 10^{-3} was chosen empirically but is justified by the fact that the non-zero weights are of the order of 10^{-1} making the induced error negligible. During the tests of this method, a non-zero value of epsilon occured once or twice per grid and were always below the threshold.

Chapter 4

Results

In this section, we will present the results obtained with the methods previously described. The DG solvers have first been applied to smooth solutions where optimal order of convergence was obtained. The effects of the proposed limiter on both smooth solutions and solutions with shocks are then discussed.

4.1 Smooth solution

The use of higher order polynomials is most effective for the representation of smooth flows. An accuracy study is done for the various solvers implemented proving that the achieved order of convergence for a $\mathbb{P}_p$ solution is the optimal $p+1$, *i.e.* the norm of the error is $||e|| = \mathcal{O}(h^{p+1})$, where h is the measure of the size of the cells. Moreover, a timing study for the 2D Euler solver shows the advantage of the higher order implementations.

4.1.1 2D Euler

The accuracy of the 2D Euler solver for smooth problems has been extensively studied by Fidkowski *et al.* [1]. The study of the Ringleb flow will be presented to prove the order of convergence of the implemented higher order solver. The results presented are steady state solutions solved using a p-multigrid solver[1].

The Ringleb flow is an exact solution of the Euler equations obtained using the hodograph method[1]. The streamlines and iso-Mach lines for a typical Ringleb solution domain are shown in Figure 4-1.

Taking advantage of the fact that the exact solution is known, we take as our domain

the circle shown inside the regular Ringleb domain. An accuracy study of the L_2 norm was performed using a set of three nested grids (88, 352, and 1408 elements) for orders of interpolation ranging from $\mathbb{P}_0$ to $\mathbb{P}_3$.

Figure 4-2 shows the solution accuracy versus grid size and order. Optimal accuracy convergence of $p+1$ is attained. Figure 4-2 also shows the error plotted versus CPU time to solution. The flow is initialized using the exact solution and the solver converges the residual to zero. A solution was taken to be converged when the error norm came within 1 percent of its final value, determined by converging the solution to machine zero residual beforehand. The advantage of high order interpolation is clear: a $p = 3$ solution on the coarsest grid yields the same accuracy as a $p = 2$ solution on a grid 16 times the size in a time of 17 seconds as compared to 352 seconds on an Intel Pentium 4 2.53 GHz system with 512 MB RAM.

4.1.2 1D Burgers

For the development of the proposed shock limiter, a higher-order DG solver for Burgers equation was developped. In 1D, we solve the equation (2.2) over a periodic domain $[0, 2]$ with the intial condition $u(x, 0) = 0.5 + sin(\pi x)$. Time integration was accomplished using the three stage TVD Runge-Kutta time marching scheme (2.16). At $t = 0.5/\pi$, the flow is still smooth. An accuracy study on uniform grids was performed to prove the order of convergence of the solver without the limiter.

	$\mathbb{P}_2$		$\mathbb{P}_3$		$\mathbb{P}_4$		$\mathbb{P}_5$	
Number of cells	L_1 error	order	L_1 error	order	L_1 error	order	L_1 error	order
5	$2.016e^{-2}$		$6.416e^{-3}$		$8.279e^{-4}$		$7.479e^{-4}$	
10	$3.442e^{-3}$	2.55	$2.75e^{-4}$	4.54	$9.342e^{-5}$	3.15	$1.715e^{-5}$	5.45
20	$4.081e^{-4}$	3.07	$3.113e^{-5}$	3.14	$2.923e^{-6}$	4.99	$5.095e^{-7}$	5.07
40	$5.217e^{-5}$	2.97	$1.802e^{-6}$	4.11	$1.123e^{-7}$	4.70	$4.010e^{-8}$	3.67

Table 4.1: Accuracy study for Burgers equation in 1D

In Table 4.1, we can see that optimal order of convergence is obtained since for $\mathbb{P}_p$ solutions, the solution converges at the order $p+1$. Nevertheless, we can see that for high order, the error bottoms out reaching machine zero, thus limiting the order of convergence.

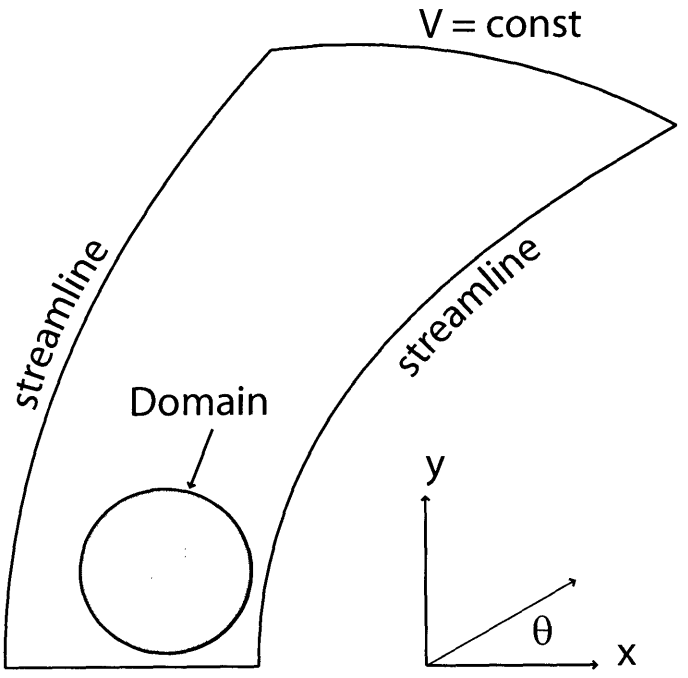

Figure 4-1: Ringleb Flow

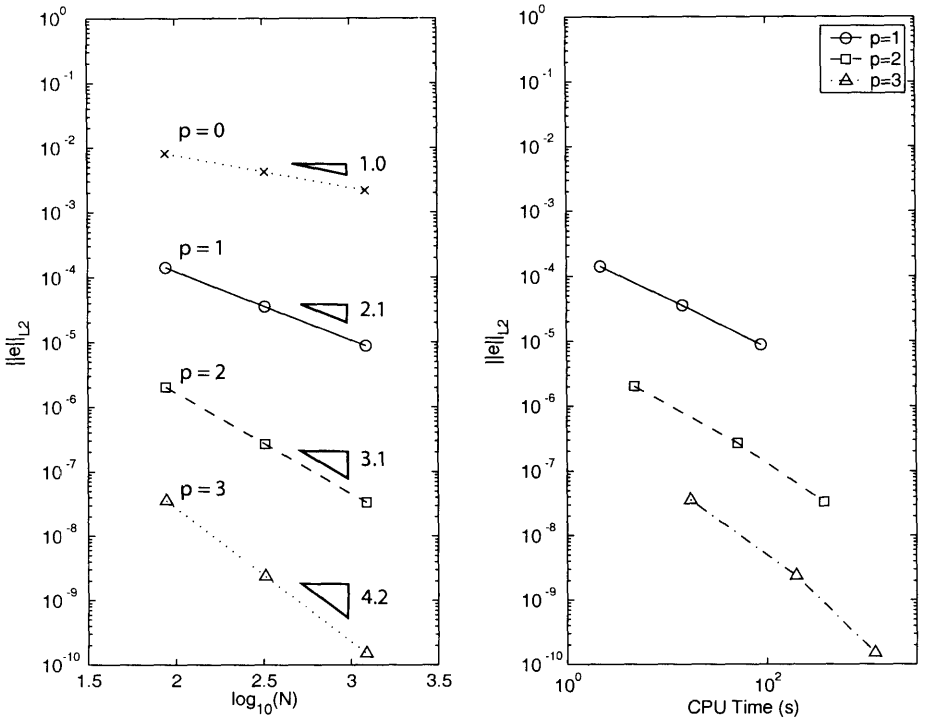

Figure 4-2: Ringleb Flow: accuracy vs. CPU time

4.1.3 2D Burgers

The 2D limiter was developped using an unstructured DG solver for Burgers equation. Using the same framework as the 2D fluid solver, an unsteady solver for the 2D Burgers equation (2.3) using the three stage Runge-Kutta time marching scheme (2.16) was build. The domain considered is the square $[0,4]^2$ with periodic boundary conditions in all directions. The flow is initialized as:

$$u(x,y,0) = 0.5 + sin\left(\frac{\pi}{2}(x+y)\right)$$

An accuracy study for $\mathbb{P}_2$ solutions was made at $t = 0.5/\pi$, when the solution is still smooth, on three nested grids of 176, 704 and 2816 elements. The results presented in table 4.2 prove that third order optimal convergence is achieved.

	$\mathbb{P}_2$	
Grid level	L_1 error	order
h	$7.801e^{-2}$	
h/2	$1.016e^{-2}$	2.94
h/4	$1.283e^{-3}$	2.98

Table 4.2: Accuracy study for Burgers equation in 2D

4.2 Limited Solutions

The proposed limiter is now applied to the solver for Burgers equation. The Runge-Kutta method is then modified in the manner presented in (2.17). We first show that for smooth solutions, the high order of accuracy is maintained and then present the effectiveness of the limiter on flows with shocks.

4.2.1 Limited smooth solutions

One Dimensional Burgers Equation

We now compare the limited solution with the non-limited solution at $t = 0.5/\pi$ using the same flow as in section 4.1.2. For compactness, we only present the results of the comparison for $\mathbb{P}_2$ and $\mathbb{P}_5$ solutions.

For all orders tested, *i.e.* $\mathbb{P}_2$ to $\mathbb{P}_5$, the limiter induced error on the coarse grid levels but this error reduces significantly as the grid becomes finer. This phenomenon is very significant in the $\mathbb{P}_5$ as can be seen in Table 4.4 with orders of convergence up to 8.4. It should be noted that this order does not suggest that the algorithm is super convergent but rather that the error induced by the limiter on coarser grids decreases as the grid becomes finer. Moreover, once the grid has become fine enough that the error of the limited solution is comparable to the non-limited one, the order of convergence becomes $p+1$ for $\mathbb{P}_p$ solutions as can be seen in the $\mathbb{P}_2$ case in Table 4.3.

	$\mathbb{P}_2$		$\mathbb{P}_2$ with limiter	
Number of cells	L_1 error	order	L_1 error	order
5	$2.016e^{-2}$		$1.751e^{-1}$	
10	$3.442e^{-3}$	2.55	$1.397e^{-2}$	3.65
20	$4.081e^{-4}$	3.07	$7.011e^{-4}$	4.32
40	$5.217e^{-5}$	2.97	$6.951e^{-5}$	3.33
80	$6.652e^{-6}$	2.97	$9.069e^{-6}$	2.97

Table 4.3: Impact of the limiter on smooth flows for Burgers equation in 1D with $\mathbb{P}_2$ interpolants

	$\mathbb{P}_5$		$\mathbb{P}_5$ with limiter	
Number of cells	L_1 error	order	L_1 error	order
5	$7.479e^{-4}$		$2.072e^{-2}$	
10	$1.715e^{-5}$	5.45	$1.713e^{-3}$	3.60
20	$5.095e^{-7}$	5.07	$1.483e^{-5}$	6.85
40	$4.010e^{-8}$	3.67	$4.359e^{-8}$	8.41

Table 4.4: Impact of the limiter on smooth flows for Burgers equation in 1D with $\mathbb{P}_5$ interpolants

Two dimensional Burgers equation

We now compare the limited solution of the 2D Burgers with the non-limited solution at $t = 0.5/\pi$. Similarly to the one dimensional case, we can see that the limiter induces a significant amount of error on coarse grids but that this induced error reduces as the grid gets finer.

	$\mathbb{P}_2$		$\mathbb{P}_2$ with limiter	
Number of cells	L_1 error	order	L_1 error	order
Grid level	L_1 error	order	L_1 error	order
h	$7.801e^{-2}$		$1.726e^{-1}$	
h/2	$1.016e^{-2}$	2.94	$4.588e^{-2}$	1.91
h/4	$1.283e^{-3}$	2.98	$3.918e^{-3}$	3.54

Table 4.5: Impact of the limiter on smooth flows for Burgers equation in 2D with $\mathbb{P}_2$ interpolants

4.2.2 Shock Solutions

One-dimensional Burgers equation

We now solve the same Burgers equation with the same initial condition as previously except we now plot the results at $t = 1.5/\pi$ when a shock has already appeared in the solution. In Figure 4-3, we plot the solution without and with the limiter for a $\mathbb{P}_5$ solution on a non-uniform grid containing 20 cells. The grid is obtained by randomly perturbing the nodes of a uniform mesh by $0.2\Delta x$. The overshoots and oscillations present in the non-limited solution are not present in the limited solution. Moreover, the shock is well captured by the limiter.

When considering lower order interpolants such as $\mathbb{P}_2$ as in Figure 4-4, it should be noticed, that the shock is also well captured and in fact, the shock is much better captured in this finer grid then with a coarser grid with higher order interpolants. Nevertheless, the limiter maintains the order of accuracy in the smooth regions and therefore in the smooth region the solution on the coarser grid using $\mathbb{P}_5$ interpolants will be more accurate.

Two dimensional Burgers equation

We solve the two dimensional problem with the same initial condition at $t = 1.5/\pi$ when a shock has already appeared. We present the results on a grid with 2816 elements represented in figure 4-5. Figure 4-6 represents the contour plots of the solution and figure 4-7 presents the cut of the solution at $x = y$.

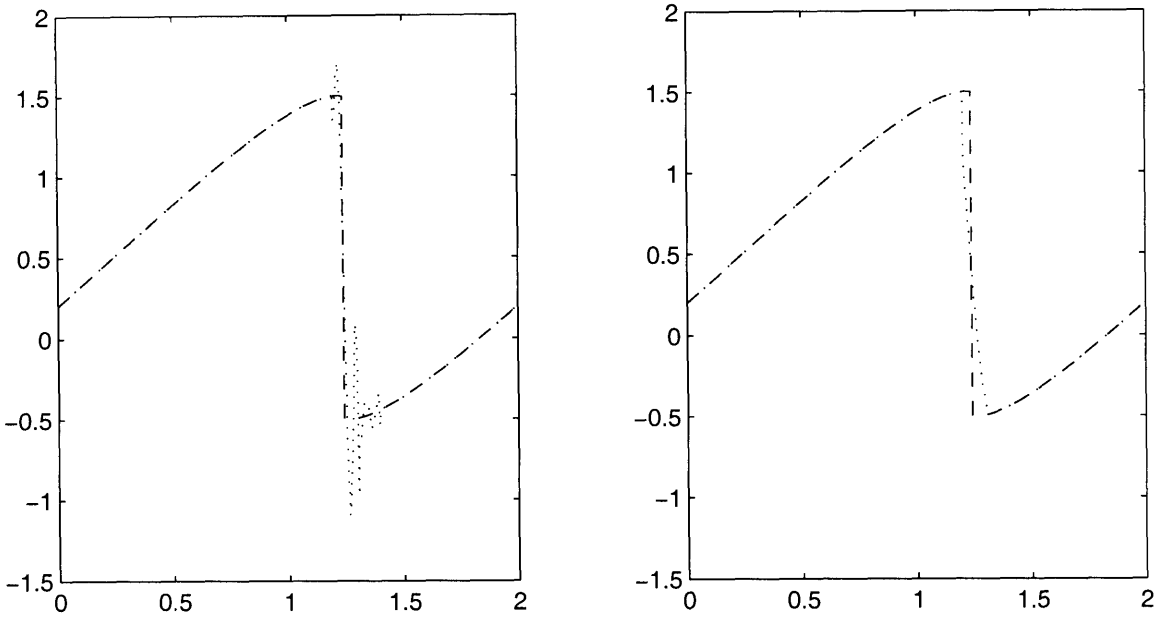

Figure 4-3: Comparison of $\mathbb{P}_5$ solution without and with the limiter on a 20 cell non-uniform grid. Dotted line: numerical solution; dashed line: exact solution.

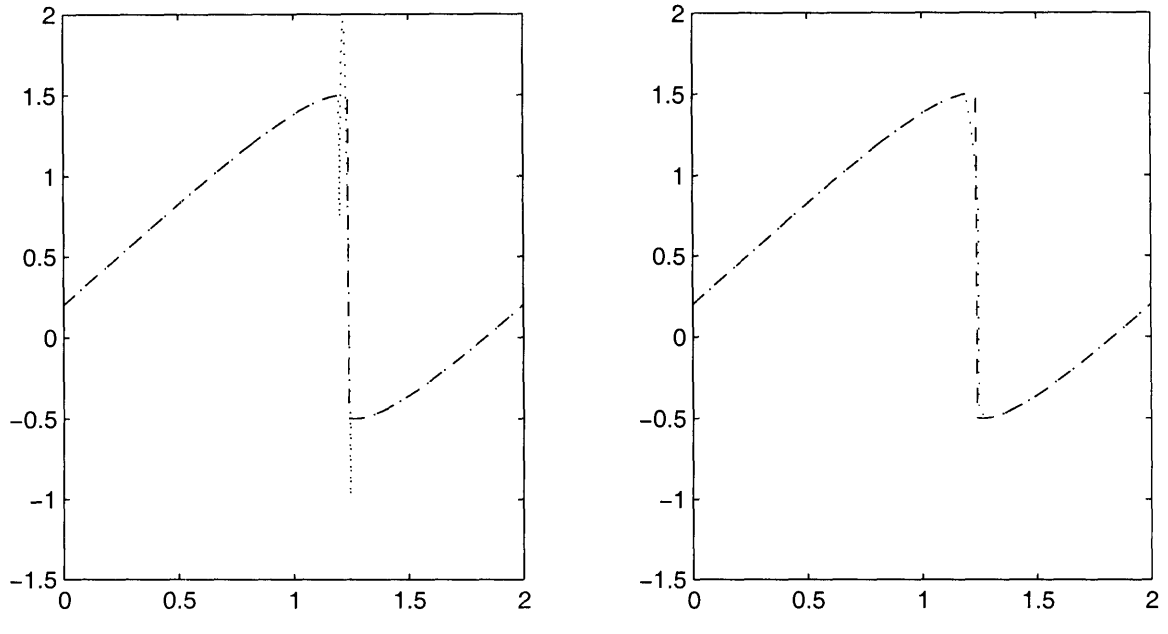

Figure 4-4: Comparison of $\mathbb{P}_2$ solution without and with the limiter on a 40 cell non-uniform grid. Dotted line: numerical solution; dashed line: exact solution.

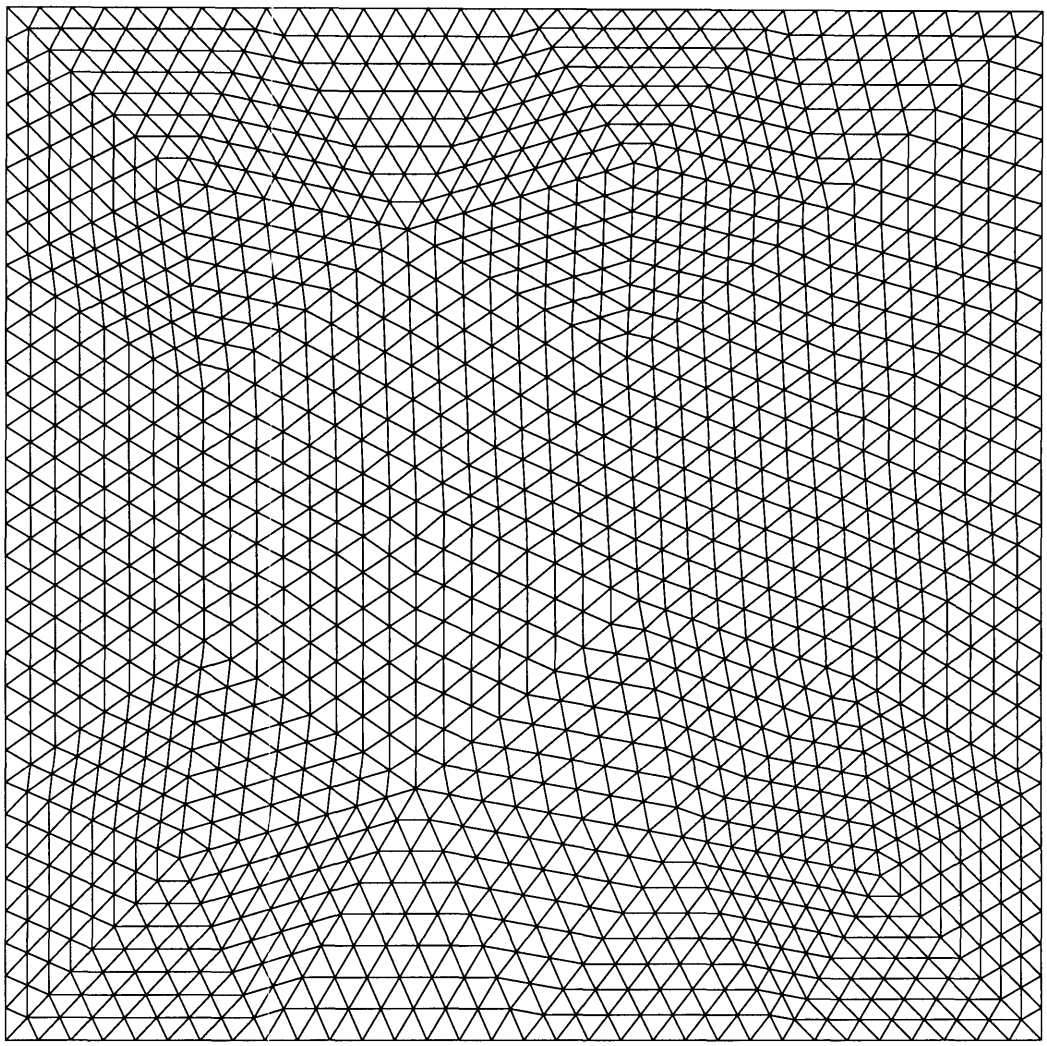

Figure 4-5: Grid with 2816 elements used in figure 4-6 and 4-7

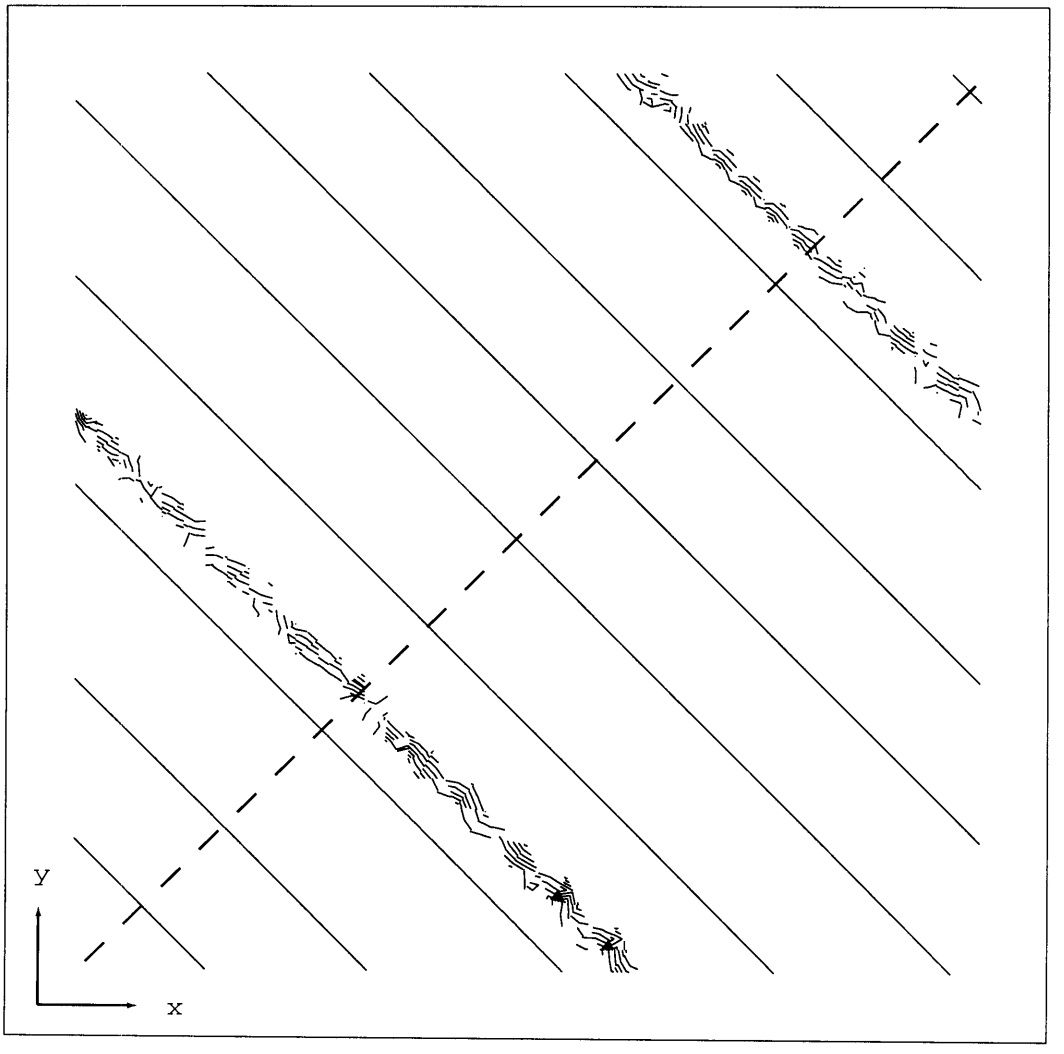

Figure 4-6: 2D Burgers equation, $u(x,0) = 0.5 + sin(\pi(x+y)/2)$ at $t = 1.5/\pi$. Contour plot of the solution. Dashed line represents the cut plane used in figure 4-7.

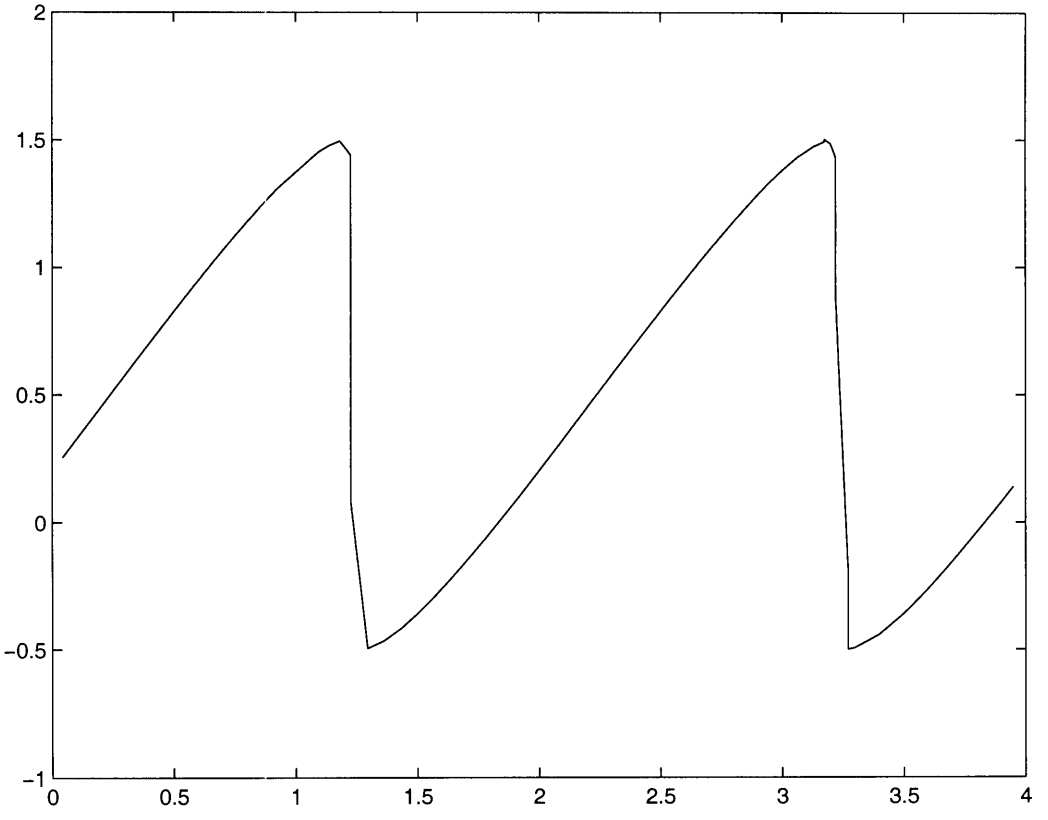

Figure 4-7: 2D Burgers equation, $u(x,0) = 0.5 + sin(\pi(x+y)/2)$ at $t = 1.5/\pi$. Cut of the solution at $x = y$. Grid with 2816 elements.

4.3 Conclusion

A higher order 2D Euler solver has been developed and the advantage of using higher order interpolations over h-refinement has been shown for smooth solutions. In order to increase the capabilities of the solver to transonic and supersonic problems, a slope limiter must be created.

A limiter based on the HWENO method introduced by Qiu *et al.* [10] has been developed for Runge-Kutta DG methods. The framework of the HWENO method was kept but a new method for computing the linear weights was created. This method is general and does not depend on the spatial dimension and of the order of the interpolants. Thus, assuming an appropriate set of stencils may be found, the method is applicable to any order and any dimension.

Using this method, a limiter for one dimensional non-uniform grids for interpolants of order $\mathbb{P}_2$ to $\mathbb{P}_5$ was created. In addition, a 2D limiter for $\mathbb{P}_2$ interpolants on unstructured grids was created. Both of these limiters maintain the order of accuracy of the solution in smooth regions and present good shock capturing capabilities. Moreover, the limiting procedure uses only the moments of the immediate neighbors of the cell which is being limited. This feature is an improvement on the WENO and the 2D HWENO limiter for structured grids proposed by Qiu [9, 14].

Although the limiting procedures are quite effective, as was discussed in section 4.2.2, the shock capturing capability would be highly enhanced through grid refinement and for maximum effectiveness, the limiting procedure should be linked with grid adaptation techniques.

Moreover, these limiting procedures are computationally expensive, especially in the 2D case. Although most of the computationally intensive work is solving the optimization problem for the linear weights which is solved only once, a reduction of the required number of stencils should be investigated.

Appendix A

Reconstruction polynomials for a 2D unstrucutred $\mathbb{P}_2$ limiter

A set of 18 reconstruction polynomials was identified making the LP-problem (3.9) feasible. Table A represents for $i = 0 \dots 17$, the moments which define the stencil S_i. The notation M_j represents the moment with respect to ϕ_j as defined in section 3.4.1. The ϕ_j are expressed in the basis (x, y) for the reference element T and in the basis (u, v) for its neighbors T_0, T_1 and T_2 as represented in in figure A-1.

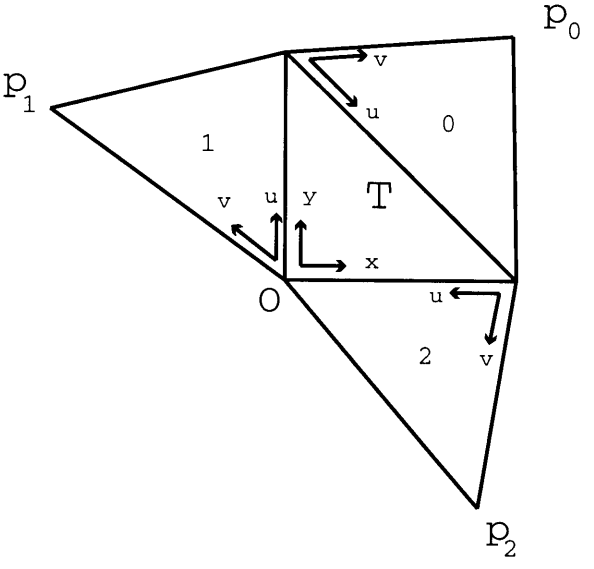

Figure A-1: The reference element T with it's three neighbors and the local basis associated with all cells.

i	T	T_0	T_1	T_2
0	M_0	M_0, M_1, M_2, M_3, M_4		
1	M_0		M_0, M_1, M_2, M_3, M_4	
2	M_0			M_0, M_1, M_2, M_3, M_4
3	M_0	M_0, M_1, M_2, M_3, M_5		
4	M_0		M_0, M_1, M_2, M_3, M_5	
5	M_0			M_0, M_1, M_2, M_3, M_5
6	M_0	M_0, M_1, M_2, M_4, M_5		
7	M_0		M_0, M_1, M_2, M_4, M_5	
8	M_0			M_0, M_1, M_2, M_4, M_5
9	M_0	M_0, M_1, M_2	M_0, M_2	
10	M_0		M_0, M_1, M_2	M_0, M_2
11	M_0	M_0, M_2		M_0, M_1, M_2
12	M_0	M_0, M_1, M_2		M_0, M_1
13	M_0	M_0, M_1	M_0, M_1, M_2	
14	M_0		M_0, M_1	M_0, M_1, M_2
15	M_0	M_0, M_1, M_2	M_0	M_0
16	M_0	M_0	M_0, M_1, M_2	M_0
17	M_0	M_0	M_0	M_0, M_1, M_2

Table A.1: Moments defining the reconstruction polynomials for the 2D limiter

Appendix B

Linear weights for 1D uniform meshes

We present the values of the linear weights for the 1D limiter on uniform meshes for interpolants of $\mathbb{P}_2$ to $\mathbb{P}_5$. For orders $\mathbb{P}_3$ and above, the linear weights are presented using rational approximations with a tolerance of 10^{-6}. The stencils defining P_i for all orders n and for all j when the j-th moment is reconstructed are described in section 3.3.2. The linear weights for $\mathbb{P}_2$ case are the same as those presented by Qiu *et al.* [10].

Moment	P_0	P_1	P_2
1	11/38	8/19	11/38
2	45/154	32/77	45/154

Table B.1: Linear weights for the 1D $\mathbb{P}_2$ limiter

Moment	P_0	P_1	P_2	P_3
1	94/633	703/2000	703/2000	94/633
2	103/626	105/313	105/313	103/626
3	204/1219	162/487	162/487	204/1219

Table B.2: Linear weights for the 1D $\mathbb{P}_3$ limiter

Moment	P_0	P_1	P_2	P_3	P_4
1	190/2781	173/704	164/441	173/704	190/2781
2	13/152	250/1009	1502/4505	250/1009	13/152
3	223/2397	225/911	447/1397	225/911	223/2397
4	224/2355	223/905	187/590	223/905	224/2355

Table B.3: Linear weights for the 1D $\mathbb{P}_4$ limiter

Moment	P_0	P_1	P_2	P_3	P_4	P_5
1	6/205	113/774	340/1047	340/1047	113/774	6/205
2	53/1285	160/983	177/598	177/598	160/983	53/1285
3	25/513	82/483	143/508	143/508	82/483	25/513
4	59/1124	145/842	321/1166	321/1166	145/842	59/1124
5	23/427	245/1418	512/1873	512/1873	245/1418	23/427

Table B.4: Linear weights for the 1D $\mathbb{P}_5$ limiter

Bibliography

[1] K. Fidkowski and D. Darmofal, Development of a Higher-Order Solver for Aerodynamic Applications, *42nd AIAA Aerospace Sciences Meeting and Exhibit*, AIAA 2004-0436, 2004.

[2] B. Cockburn and C.-W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, *Journal of Scientific Computing*, pp. 173–261, 2001.

[3] G. Jiang and C.-W. Shu, On a cell entropy inequality for discontinuous galerkin methods, *Mathematics of Computation*, Vol. 62, Number 162, pp. 531–538, 1994.

[4] S. Osher, Riemann solvers, the entropy condition, and difference approximation, *SIAM J. Numer. Anal.* 21, pp. 217-235, 1984.

[5] P. Roe, Approximate Riemann solvers, parametric vectors, and difference schemes, *Journal of Computational Physics*, Vol. 43, pp. 357–372, 1981.

[6] S. Gottlieb, C.-W. Shu, Total variation diminishing Runge-Kutta schemes, *Mathematics of Computation*, Vol. 67, Number 221, pp. 73–85, 1998.

[7] Q. Chen and I. Babuska, The optimal symmetrical points for polynomial interpolation of real functions in the tetrahedron, *Computer methods in applied mechanics and engineering*, 137, pp. 89–94, 1996.

[8] P. Solin, K. Segeth and I. Dolezel, Higher-Order Finite Element Methods, *Chapman & Hall/CRC*, 2003

[9] J. Qiu and C.-W. Shu, Runge-Kutta discontinuous Galerkin method using WENO limiters , submitted to *SIAM Journal on Scientific Computing.*

[10] J. Qiu, C.-W. Shu, Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: one-dimensional case. *Journal of Computational Physics* 193, pp. 115–135, 2003.

[11] A. Burbeau, P. Sagaut and Ch.-H. Bruneau, A Problem-Independent Limiter for Higher-Order Runge-Kutta Discontinuous Galerkin Methods, *Journal of Computational Physics* 169, pp. 111–150, 2001.

[12] C.-W. Shu, Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws *ICASE Report* No. 97–65, 1996.

[13] G.-S. Jiang, C.-W. Shu, Efficient Implementation of Weighted ENO Schemes, *Journal of Computational Physics* 126, pp. 202–228, 1996.

[14] J. Qiu, C.-W. Shu, Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method II: two-dimensional case. submitted to *Computers and Fluids*, 2004.

[15] C. Hu, C.-W. Shu, Weighted Essentially Non-oscillatory Schemes on Triangular Meshes, *Journal of Computational Physics* 150, pp. 97–127, 1999.

[16] J. Shi, C. Hu and C.-W. Shu, A Technique of Treating Negative Weights in WENO Schemes, *Journal of Computational Physics* 175, pp. 108–127, 2002