TÉCNICO LISBOA

Implementation of an arbitrary-order Finite-Volume Least-Squares WENO for inviscid Euler equations

Rafael Castro Mota

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Dr. Duarte Manuel Salvador Freire Silva de Albuquerque

Examination Committee

Chairperson: Prof. Fernando Lau

Supervisor:Dr. Duarte Manuel Salvador Freire Silva de Albuquerque

Member of the Committee: Prof. José Manuel Da Silva Chaves Ribeiro Pereira

August 2021

Dedicated to my friends and family.

Acknowledgments

I would like to thank the personnel at LASEF, in particular Prof. José Carlos Pereira, for the opportunity to develop my skills in computational fluid dynamics and expand my overall knowledge. I would also like to thank my supervisor Dr. Duarte Albuquerque for the help and guidance given throughout this Thesis and my colleagues, and friends, Élio Pereira and Liliana Sousa for the time spent reviewing a major part of the content of this work.

For my friends André Oliveira and José Rocha a warm salute is due. They made my days at college more enjoyable.

Finally, i would like to thank my close family for providing the monetary, social and emotional means that allowed for my academic education.

Resumo

Nesta tese é implementado e testado um esquema WENO de ordem de convergência arbitrária para a formulação de uma e duas dimensões das equações de Euler. As malhas usadas serão regulares e triangulares. Os esquemas WENO funcionam através de múltiplos conjuntos de dados para o mesmo ponto de interesse. A regressão distinta de cada um deles é feita através de um modelo polinomial. Os polinómios resultantes são depois combinados em um único. Como as oscilações não harmónicas da malha devem ser evitadas, cada polinómio recebe um peso decorrente do quanto oscilam nas imediações do ponto de interesse. Daí o nome WENO (Weighted Essentially Non-Oscillatory). Na construção destes polinómios será utilizado o método dos mínimos quadrados pois providencia flexibilidade para aplicações futuras. Nos capítulos introdutórios é dado um contexto para a importância desta linha de esquemas de alta ordem (1) e a base teórica necessária para a compreensão do conteúdo da Tese (2). São seguidos de um capítulo dedicado à implementação e discussão de resultados obtidos para o caso uni-dimensional (3) e outro de igual estrutura para o caso bi-dimensional (4). Na conclusão (5), é dada uma visão geral sobre as limitações e vantagens do esquema implementado.

Palavras-chave: WENO, Euler, Runge-Kutta, Riemann, Least-Squares, Shocks

Abstract

In this thesis, an arbitrary order Least-Squares-WENO (LS-WENO) scheme will be applied for both the one-dimensional and two-dimensional finite volume formulation of the Euler equations. WENO schemes work by defyning several data sets (stencils) for the same point of interest and then combining the resulting polynomial models into a single final polynomial. As spurious oscillations are to be avoided near discontinuities and shocks, each polynomial model receives a weight dependent on their oscillating behaviour, hence the name WENO (Weighted Essentially Non-Oscillatory). The regression method used will be the Least-Squares Method as it provides flexibility with unstructured grids. The introductory chapters give a context for this line of high-order schemes (1) as well as the necessary theoretical background (2). These are followed up by a chapter showcasing the implementation of the scheme developed for the uni-dimensional case and the discussion of several test cases. (3) and another, with the same structure, regarding the two-dimensional scenario (4. The concluding chapter (5) discusses the advantages and drawbacks of the developed scheme.

Keywords: WENO, Euler, Runge-Kutta, Riemann, Least-Squares, Shock

Contents

List of Figures

Nomenclature

Greek symbols

β	A set of wave fronts that discretizes the inner region of an expansion fan.
δ, ζ	Components in the coordinate system for the retangle triangle.
ϵ, η	Components in the auxiliary coordinate system.
γ	Specific heat ratio.
κ	Relative condition number.
Λ	Eigenvalues diagonal matrix.
ν	Eigenvector.
$\overline{\kappa}$	Average relative condition number.
ρ	Density ($Kg \cdot m^{-3}$).

Roman symbols

$\mathbb{A}$	Flux jacobian matrix for the Euler equations.
$\mathbb{H}$	Total number of terms in a polynomial model.
$\mathbb{J}$	Vector with the Riemann invariants of interest
$\mathbb{X}$	Eigenvector matrix.
$A(S_i)$	Area of a two-dimensional target cell (m^2).
$d(S_i)$	Lenght of a one-dimensional target cell (m).
d_{ref}	Reference dimension for the cells in a mesh (m).
E	Total energy density ($J \cdot m^{-3}$).
H	Total specific enthalpy (KJ $\cdot Kg^{-1}$).
J	Jacobian of a linear transformation.
L	Legendre polynomials.

M Mach number.

N Number of cells used to discretize a domain of interest.

N_β Number of waves used to discretize the expansion fan.

$N_{\mathbb{S}}$ Number of stencils used for each cell.

p Pressure (Pa).

Q Vector with the dependent variables of the Euler equations.

R Specific gas constant ($J \cdot kg^{-1} \cdot K^{-1}$).

r Degree of accuracy.

S Wave propagation speed ($m \cdot s^{-1}$).

S_{ij} A specific face of a target cell.

s_i Path along the boundary of a target cell.

T Temperature of the fluid (K).

a Speed of sound ($m \cdot s^{-1}$).

$\wp$ Degree of a polynomial.

SI Smoothness indicator.

u Particle velocity ($m \cdot s^{-1}$).

u_x, u_y Velocity cartesian components ($m \cdot s^{-1}$).

w Linear weight attributed to each stencils.

Subscripts

$\mathbb{S}$ Related to stencils.

C Compression.

c Centroid of a cell.

CD Contact discontinuity.

D Diagonal.

d Computational index for the cells that discretize a domain.

E Expansion.

g Computational index for the Gauss-Legendre points used.

h Computational index for a particular basis function of a polynomial model.

i Computational index for a particular target cell in the mesh.

j Computational index for a particular face of a cell in the mesh.

k Computational index for a particular entry on a dependent variables vector.

L Left.

m Computational index for a particular stencil for a given cell in the mesh.

n Computational index for a particular entry of a given stencil.

R Right.

ref Reference.

u Computational index for a particular step in a Runge Kutta method.

x, y Cartesian components.

Superscripts

* Star region in HLLC Riemann solver.

-1 Inverse.

$4 \rightarrow 3$ State 4 to state 3.

CW WENO with use of characteristic variables.

GL Gauss-Legendre.

LS Least-squares.

n Normal component.

nv Normal-tangential coordinate system.

T Transpose.

v Tangential component.

Acronyms

CFD Computational Fluid Dynamics.

GLQ Gauss-Legendre Quadrature.

HLL Harten Lax Leer.

HLLC Harten Lax Leer Contact.

RS Riemann Solver.

SFFS Supersonic Forward Facing Step.

SST Sod's Shock Tube.

Chapter 1

Introduction

Traditional numerical schemes for supersonic applications are tipically lower-order schemes.These schemes are easy to implement and very robust but they do, however, require very fine meshes to capture small details in the flow. High-order accuracy schemes allow for adequate discretization of finer smooth flow structures in coarser meshes. In the presence of sharp gradients gradients, discontinuities and shocks, however, typical polynomial regressing methods fail to produce coherent results. Important properties, such as monotonicity and positivity, are not preserved and the Gibbs phenomenon and associated unphysical oscillatory behaviour appear in any polynomial model that approximates data with such nuisances. These challenges normally limit commercial CFD code to second-order TVD schemes.

ENO (Essentialy Non-Oscillatory) schemes were the first attempt to bring high-order accuracy to discontinuity-ridden domains [1]. The scheme does this by choosing the smoothest set of data closest to the point of interest, avoiding sharp gradient discontinuities and shocks. The methodology of choosing the smoothest data set, altough effective, is time consuming, computationally inefficient and, therefore, limits the applicability of the scheme. In the year of nineteen-ninety-four (1994), the first WENO scheme was developed by Liu, Chan and Osher [2], as an attempt to improve the efficiency of ENO schemes.

In this Thesis, an arbitrary order Weighted-ENO scheme was applied for both the one-dimensional and two-dimensional finite volume formulation of the Euler equations. WENO schemes allow for high-order of accuracy in smooth regions of the domain while maintaing positivity and monotonicity near sharp gradients, discontinuities and shocks. They work by defining several data sets, called stencils, for the same point of interest and then combining the resulting approximated polynomial models. As spurious oscillations are to be avoided, each polynomial model receives a weight dependent on their oscillatory behaviour, hence the name WENO (Weighted Essentially Non-Oscillatory). As the underlying architecture does not need complicated algorithm for choice of data, stencils are only chosen once at the pre-processing stage of the CFD code. The methodologies for the one and two-dimensional schemes and respective numerical testing are presented in sections 3 and 4. Section 2 provides the necessary theoretical background for comprehension of the developed schemes.

The presented WENO scheme and respective developed code, some of which presented in appendix B, aim to serve as a basis for future work regarding high order methods capable of good performance

in transient and supersonic flow regimes. For future flexibility, the regression method used will be the Weighted-Least-Squares Method. All numerical coding was done in MATLAB® as it provides a user-friendly coding environment to develop and test new numerical techniques. Results for several test cases (Diffusion of a Gaussian, Sod's Shock tube, Sedovs cylindric Blast Wave, Supersonic Forward Facing Step) are presented and compared to traditional numerical schemes made to deal with discontinuities and shocks.

Chapter 2

Background

2.1 Gibbs phenomenom

The Gibbs phenomenon translates an inherent difficulty of approximating discontinuous functions by a sum of continuous ones. The phenomena was first reported for the Fourier series of a square wave. Overshoots and undershoots appear in the vicinity of a discontinuity and increasing the number of terms of the Fourier Series does not make the spurious oscillations disappear.

In CFD, high-order approximations of fluxes or other properties require polynomial regression and the Gibbs phenomena represents a major hindrance in maintaining physical coherency near discontinuities. Difficulty in assuring monotonicity preservation, and positivity of density or pressure, for example, is a major limiting factor in high-order schemes.

.

2.2 Least-Squares Method

The Least-Squares Method (LS) is a classic tool in regression analysis useful for approximating a regression model to a set of data ($Y(X), X$). The core reasoning behind the method is the minimization of the square of the error in the data points (residuals) through control of the coefficients in the regression model. Having a set of basis functions Φ_h, the regression model with $\mathbb{H}$ number of terms can be described by equation 2.1.

$$\sum_{h=1}^{H} C_h \Phi_h = C_1\Phi_1 + C_2\Phi_2 + ... + C_H\Phi_H \tag{2.1}$$

In each data point ($Y(X_n), X_n$), the respective residual is computed by using equation 2.2.

$$r_n = Y(X_n) - \sum_{h=1}^{H} C_h \Phi_h(X_n) \tag{2.2}$$

If the data set has N_S number of data points, one can present the residuals in matrix form, as shown

in the equations contained in 2.3.

$$r = Y^{LS} - D^{LS}C_f$$

$$r = \begin{bmatrix} r_1 \\ r_2 \\ ... \\ r_{NS} \end{bmatrix}; D^{LS} = \begin{bmatrix} \Phi_1(X_1) & \Phi_1(X_1) & ... & \Phi_H(X_1) \\ \Phi_1(X_2) & \Phi_1(X_2) & ... & \Phi_H(X_2) \\ & & ... & \\ \Phi_1(X_{N_S}) & \Phi_1(X_{N_S}) & ... & \Phi_{\mathbb{H}}(X_{N_S}) \end{bmatrix}; C_f = \begin{bmatrix} C_1 \\ C_2 \\ .. \\ C_{\mathbb{H}} \end{bmatrix}; Y^{LS} = \begin{bmatrix} Y(X_1) \\ Y(X_2) \\ ... \\ Y(X_{N_S}) \end{bmatrix} \tag{2.3}$$

It is very important to state that in a typical least squares method application the number of points to be interpolated is greater than the number of coefficients ($N_S > \mathbb{H}$) making the system of equations in 2.4 overdetermined.

$$\begin{bmatrix} \Phi_1(X_1) & \Phi_1(X_1) & ... & \Phi_H(X_1) \\ \Phi_1(X_2) & \Phi_1(X_2) & ... & \Phi_H(X_2) \\ & & ... & \\ \Phi_1(X_{N_S}) & \Phi_1(X_{N_S}) & ... & \Phi_{\mathbb{H}}(X_{N_S}) \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ .. \\ C_{\mathbb{H}} \end{bmatrix} = \begin{bmatrix} Y(X_1) \\ Y(X_2) \\ .. \\ Y(X_{N_S}) \end{bmatrix} \tag{2.4}$$

Having more data points than coefficients provides robustness and flexibility as the resulting model is not as susceptible to learning noise of a single data point (error in the value) and the number of points does not dictate the number of degrees of freedom. The typical Least Squares problem presents itself as shown in equation 2.5.

$$min(\sum_{n=1}^{K}(r_n)^2) \quad w.r.t \;\; C_f \tag{2.5}$$

In CFD, the model is only locally pertinent as it is normally used for approximating numerical fluxes and flow properties for a target cell. Consequently, the model should be more accurate for points closer to the area of interest.To provide this feature on the regression, the Weighted-Least-Square Problem, presented in equation 2.6, is used.

$$min(\sum_{n=1}^{K} W_n^{LS}(r_n)^2) \quad w.r.t \;\; C_f \tag{2.6}$$

Based on [3], the weight function in equation 2.7 will be one used for a p^{th} degree polynomial model centered on a generic point with coordinate vector X_0.

$$W_n^{LS} = \frac{1}{d(X_n)^p} \;\; with \; d = \sqrt{(X_1 - X_0)^2} \tag{2.7}$$

The weight function also has a reducing effect on the condition number of the matrices calculated (see [4]).

In matrix form, the problem in equation 2.6 can be described by equation 2.8.

$$min(r^T \cdot W^{LS} \cdot r) \quad w.r.t \;\; C_f \tag{2.8}$$

Equation 2.8 can be further simplified into equation 2.9.

$$min((Y^{LS} - DC_f)^T W^{LS} (Y^{LS} - DC_f)) \quad w.r.t \;\; C_f \tag{2.9}$$

Matrix W^{LS} is a diagonal matrix with the respective weight for the residual of each data point. The configuration of W^{LS} is shown in equation 2.10.

$$W^{LS} = \begin{bmatrix} W_1^{LS} & & \\ & \ddots & \\ & & W_{N_{LS}}^{LS} \end{bmatrix} \tag{2.10}$$

As the residuals depend linearly on the model coefficients, the minimum of the weighted squared residuals can be found by derivating with respect to C_f and equalling to zero, as depicted in equation 2.11.

$$\frac{d(r^T \cdot W^{LS} \cdot r)}{dC_f} = 0 \tag{2.11}$$

Equation 2.11 can be expressed as equation 2.12.

$$(D^{LS^T} \cdot W^{LS} \cdot D^{LS}) \cdot C_f = (D^{LS} \cdot W^{LS}) \cdot Y_s \tag{2.12}$$

In equation 2.13, the terms that only depend on the independent variables of the data set are be compiled into a single matrix (P).

$$P^{LS} = (D^{LS^T} \cdot W^{LS} \cdot D^{LS})^{-1} \cdot D^{LS} \cdot W^{LS} \tag{2.13}$$

Equation 2.13 can be used to transform 2.12 into equation 2.14.

$$C_f = P^{LS} \cdot Y_s \tag{2.14}$$

Again, it should be noted that matrix P is only dependent on the independent variables (X_n) of the data set.

2.3 Euler Equations

2.3.1 Compressible Euler Equations

Regarding fluid dynamics, the Euler equations govern compressible, adiabatic and inviscid flow. The equations for G number of dimensional spatial components have the following form described in equa-

tions 2.15, 2.16 and 2.17.

$$\frac{\partial \rho}{\partial t} + \sum_{j=1}^{G} \frac{\partial \rho u_i}{\partial x_i} = 0, (i = 1, ..., G) \tag{2.15}$$

$$\frac{\partial \rho u_i}{\partial t} + \sum_{j=1}^{G} u_j \frac{\partial \rho u_i}{\partial x_j} = \frac{\partial p}{\partial x_i}, (i = 1, ..., G) \tag{2.16}$$

$$\frac{\partial E}{\partial t} + \sum_{j=1}^{G} \frac{\partial u_j (E + p)}{\partial x_j} = 0, (i = 1, ..., G) \tag{2.17}$$

2.3.2 Compressible One-Dimensional Euler Equations

In equation 2.18, the conservative form for the one-dimensional compressible Euler equations is shown.

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{dx} = 0. \tag{2.18}$$

The dependent Euler variables vector (Q) and flux vector (F) are described in equation 2.19.

$$Q = \begin{bmatrix} \rho \\ \rho u_x \\ E \end{bmatrix} \qquad F(x) = \begin{bmatrix} \rho u_x \\ \rho {u_x}^2 + p \\ u_x (E + p) \end{bmatrix} \tag{2.19}$$

Finite Volume Formulation for the Compressible 1D Euler Equations

A one dimensional domain can be discretized by N equal lenght cells and immersed in a field of dependent variables Q, as illustrated by figure 2.1.

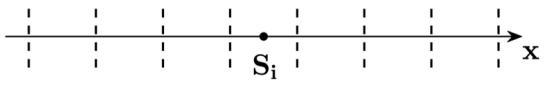

Figure 2.1: One dimensional domain discretized with equal lenght cells

Integrating the set of equations in 2.18 along the random target cell S_i, one gets equation 2.20

$$\int_{x(S_{i1})}^{x(S_{i2})} (\frac{\partial Q}{\partial t} + \frac{\partial F_x}{\partial x}) \ \partial x = 0 \tag{2.20}$$

Apllying the second theorem of calculus to the temporal term in equation 2.20 results in the mathematical expression in 2.21.

$$\int_{x(S_{i1})}^{x(S_{i2})} \frac{\partial Q}{\partial t} \ \partial x = \frac{\partial}{\partial t} \int_{x(S_{i1})}^{x(S_{i2})} Q \ \partial x \tag{2.21}$$

The cell averaged quantities in the target cell ($\overline{Q}(S_i)$ can be defined as demonstrated in equation 2.22.

$$\overline{Q}(S_i) = \frac{1}{d(S_i)} \int_{x(S_{i1})}^{x(S_{i2})} Q \;\; \partial x \tag{2.22}$$

Using equations 2.21 and 2.22, the integral in 2.20 can be expressed by equation 2.23.

$$d(S_i)\frac{\partial \overline{Q}}{\partial t} + \int_{(S_{i1})}^{x(S_{i2})} \frac{\partial F_x}{\partial x} \;\; \partial x = 0 \tag{2.23}$$

The former equation (2.23) can be further developed into equation 2.24.

$$d(S_i)\frac{\partial \overline{Q}}{\partial t} = -(F(x\big(S_{i2})) - F(x\big(S_{i1}))) \tag{2.24}$$

The time derivative of the cell averaged quantities ($\frac{\partial \overline{Q}}{\partial t}$) can then, consequently, be expressed as the sum of left and right fluxes , known as residual of the cell ($R(i,t)$), at that instance of time divided by the lenght of the cell ($d(S_i)$). The mathematical form of the previous phrase is shown in equation 2.25.

$$\frac{\partial \overline{Q}}{\partial t} = -\frac{1}{d(S_i)} R(S_i, t) \tag{2.25}$$

Characteristic Form for the Compressible 1D Euler Equations

Understanding the decoupled formulation of the Euler one-dimensional equations is useful for the correct comprehension of the use of characteristic variables in the implemented WENO schemes. Therefore, it will be explored in this section.

For decoupling the Euler one-dimensional equations, a Flux Jacobian ($\mathbb{A} = \frac{\partial F}{\partial Q}$) must be computed. The complete form for this matrix as a function of Euler dependent variables (Q_k) is shown in 2.26.

$$\frac{\partial F}{\partial Q} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)(\frac{Q_2}{Q_1})^2 & (3-\gamma)\frac{Q_2}{Q_1} & \gamma - 1 \\ -\gamma\frac{Q_2 Q_3}{Q_1} + (\gamma-1)(\frac{Q_2}{Q_1})^3 & \gamma\frac{Q_3}{Q_1} - \frac{3}{2}(\gamma-1)(\frac{Q_2}{Q_1})^2 & \gamma\frac{Q_2}{Q_1} \end{bmatrix} \tag{2.26}$$

In the effort of further simplifying equation 2.26, the total specific enthalpy (H) can be expressed by equation 2.27.

$$H = \frac{(E+p)}{\rho} = \frac{a^2}{\gamma - 1} + \frac{1}{2}u^2 \tag{2.27}$$

Thus, using equation 2.27, the Flux Jacobian Matrix can be expressed as equation 2.28.

$$\frac{\partial F}{\partial Q} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)u^2 & (3-\gamma)u & \gamma - 1 \\ \frac{1}{2}(\gamma-1)u^3 - Hu & H - (\gamma-1)u^2 & \gamma u \end{bmatrix} \tag{2.28}$$

By using eingenvalue decomposition, the Flux Jacobian Matrix can also be decomposed into the product of three matrices, as shown in equation 2.29.

$$\frac{\partial F}{\partial Q} = \mathbb{X}_R \cdot \Lambda \cdot {\mathbb{X}_R}^{-1} \tag{2.29}$$

Equations in 2.30 show how matrix X_R is composed by the right eigenvectors arranged in columns and matrix Λ by the respective eingenvalues set on the diagonal.

$$\mathbb{X}_R = \begin{bmatrix} 1 & 1 & 1 \\ u-a & u & u+a \\ H-ua & \frac{1}{2}V^2 & H+ua \end{bmatrix}; \qquad \Lambda = \begin{bmatrix} u-a & 0 & 0 \\ 0 & u & 0 \\ 0 & 0 & u+a \end{bmatrix} \tag{2.30}$$

Using equation 2.29, the spatial derivative of the Flux vector ($\frac{\partial F}{\partial x}$) can be transformed into equation 2.31.

$$\frac{\partial F}{\partial x} = \frac{\partial F}{\partial Q}\frac{\partial Q}{\partial x} = (\mathbb{X}_R \cdot \Lambda \cdot {\mathbb{X}_R}^{-1})\frac{\partial Q}{\partial x} \tag{2.31}$$

The former equation (2.31) can be used to rewrite the one-dimensional Euler system of equations as 2.32.

$$\frac{\partial Q}{\partial t} + (X_R \cdot \Lambda \cdot {X_R}^{-1})\frac{\partial Q}{\partial x} = 0 \tag{2.32}$$

Multiplying the left eigenvalue matrix ($\mathbb{X}_R^{-1}$) on the left of each side of equation 2.32, one can obtain equation 2.33.

$${\mathbb{X}_R}^{-1} \cdot \frac{\partial Q}{\partial t} + {\mathbb{X}_R}^{-1}(\mathbb{X}_R \cdot \Lambda \cdot {\mathbb{X}_R}^{-1})\frac{\partial Q}{\partial x} = 0 \Rightarrow {\mathbb{X}_R}^{-1} \cdot \frac{\partial Q}{\partial t} + \Lambda \cdot {\mathbb{X}_R}^{-1} \cdot \frac{\partial Q}{\partial x} = 0 \tag{2.33}$$

To this point, the Euler equations are still not in full decoupled form and will continue to not truly be as correct decoupling is only possible if the left eigenvector matrix (${X_R}^{-1}$) was deemed constant across the domain and time. In such a hypotethical scenario, the formulation in 2.34 would be mathematically correct.

$$\frac{\partial({\mathbb{X}_R}^{-1}Q)}{\partial t} + \Lambda \cdot \frac{\partial({\mathbb{X}_R}^{-1}Q)}{\partial x} = 0 \tag{2.34}$$

To circumvent the variation along time and space for the left eigenvectors, an approximated set of equations is used. A constant local approximation of the Jacobian Matrix is employed, as expressed in equation 2.35.

$$\frac{\partial F}{\partial Q} \approx \widetilde{\mathbb{A}} = \widetilde{X}_R \cdot \widetilde{\Lambda} \cdot \widetilde{X}_R^{-1} \tag{2.35}$$

With the enployment of the locally approximated Flux Jacobian matrix, equation 2.33 can be decoupled , taking the form of equation 2.36.

$$\frac{\partial(\widetilde{X}_R^{-1}Q)}{\partial t} + \Lambda \cdot \frac{\partial(\widetilde{X}_R^{-1}Q)}{\partial x} = 0 \Rightarrow \frac{\partial Z}{\partial t} + \Lambda \cdot \frac{\partial Z}{\partial x} = 0 \tag{2.36}$$

Each individual characteristic quantity in the set of local characteristic variables ($Z = \widetilde{\mathbb{X}_R}^{-1} Q$) remains constant along their respective characteristic curve. This concept is translated mathematically by 2.37.

$$\frac{\partial x}{\partial t} = u - a \rightarrow Z_1 = cte \quad \frac{\partial x}{\partial t} = u \rightarrow Z_2 = cte \quad \frac{\partial x}{\partial t} = u + a \rightarrow Z_3 = cte \tag{2.37}$$

The Z quantities are each related to two Riemman invariants of the one-dimensional Euler equations and, as they do not change along the characteristic curves, the relations in equation 2.37 can be established.

$$\begin{gathered} Z_1 = cte \Rightarrow \mathbb{J}_1 = u - \tfrac{2 \cdot a}{\gamma - 1} = cte, \ \mathfrak{s} = cte; \\ Z_2 = cte \Rightarrow \mathbb{J}_2 = u = cte, \ P = cte; \\ Z_3 = cte \Rightarrow \mathbb{J}_3 = u + \tfrac{2 \cdot a}{\gamma - 1} = cte, \ \mathfrak{s} = cte \end{gathered} \tag{2.38}$$

2.3.3 Compressible Two-Dimensional Euler Equations

In conservative form, the Compressible one-dimensional Euler equations take the configuration of 2.39.

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0. \tag{2.39}$$

The Euler dependent variables vector (Q), the flux vector for the xx axis direction (F) and yy axis direction (G) are described in equation 2.40.

$$Q = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ E(x) \end{bmatrix} \qquad F = \begin{bmatrix} \rho u_x \\ \rho {u_x}^2 + p \\ \rho u_x u_y \\ u_x(E + p) \end{bmatrix} \qquad G = \begin{bmatrix} \rho u_x \\ \rho u_x u_y \\ \rho {u_y}^2 + p \\ u_y(E + p) \end{bmatrix} \tag{2.40}$$

Finite Volume Formulation for the Compressible 2D Euler Equations

A two-dimensional domain, immersed in a field of properties Q, may be discretized by N number of two-dimensional cells, such as the four-sided one in figure 2.2.

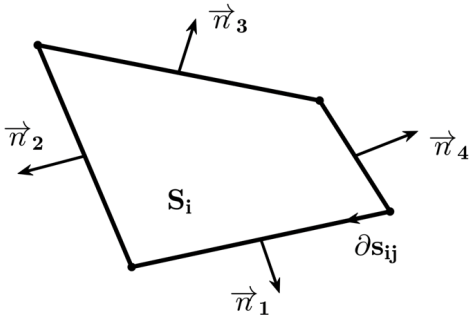

Figure 2.2: A four-sided two-dimensional target cell (S_i).

Integrating equation 2.39 on the target two-dimensional cell (S_i), one gets equation 2.41.

$$\iint_{S_i} (\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}) \partial S_i = 0 \tag{2.41}$$

Equation 2.41 can be further simplified into equation 2.42.

$$\iint_{S_i} (\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}) \partial x \partial y = 0 \Rightarrow \iint_{S_i} \frac{\partial Q}{\partial t} \partial x \partial y + \iint_{S_i} (\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}) \partial x \partial y = 0 \tag{2.42}$$

In equation 2.43, the Green theorem is presented, with arbitrary functions $L(x, y)$ and $M(x, y)$ possessing continuous partial derivatives on S.

$$\iint_S (\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y}) \partial x \partial y = \oint_s M \partial y + L \partial x \tag{2.43}$$

The Green theorem (equation 2.43) can be used to transform the integral of the fluxes in equation 2.42 into equation 2.44.

$$\iint_S (\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}) \partial x \partial y = \oint_{s_i} F \partial y - G \partial x \tag{2.44}$$

If the outter boundary (s_i) of the target cell is traversed in anti-clockwise fashion, as represented in figure 2.2 (for face j), the tangent to that path ($\overrightarrow{v}$) can be described by equation 2.45.

$$\overrightarrow{v} = (v_x, v_y) = (-n_y, n_x) \tag{2.45}$$

Equations in 2.46 are useful for further progress.

$$\partial x = v_x \cdot \partial C = -n_y \cdot \partial s_i \qquad \partial y = v_y \cdot \partial s_i = n_x \cdot \partial s_i \tag{2.46}$$

The same equations (equations in 2.46) can then be used to modify equation 2.44 into equation 2.47.

$$\oint_{s_i} F\partial y - G\partial x = \oint_{s_i} F \cdot n_x \cdot \partial s_i - G \cdot (-n_y) \cdot \partial s_i = \oint_{s_i} (F \cdot n_x + G \cdot n_y)\partial s_i \tag{2.47}$$

The time derivative for the Euler variables ($\frac{\partial Q}{\partial t}$) for equation 2.42 can be treated using the process previously described in equations 2.21 and 2.22 (section 2.3.2), resulting in equation 2.48.

$$\iint_{S_i} (\frac{\partial Q}{\partial t})\partial S = \frac{1}{A(S_i)} \frac{\partial \overline{Q}}{\partial t} \tag{2.48}$$

An integral around the boundary of the target cell can be described as a sum of the integral in each face. Therefore, it is possible to express equation 2.47 as equation 2.49

$$\oint_{s_i} (F \cdot n_x + G \cdot n_y)\partial s_i = \sum_{j=1}^{4} (\int_{s_{ij}} F \cdot n_x + G \cdot n_y)\partial s_{ij} = R(S_i, t) \tag{2.49}$$

Finally, the time derivative of the cell averaged properties can be expressed by the negative of the residual for the target cell at that given instance of time divided by the area of said cell ($A(S_i)$). Equation 2.50 reflects the final form of a finit volume formulation for the two-dimensional Euler equations.

$$\frac{\partial \overline{Q}}{\partial t} = -\frac{1}{A(S_i)} \cdot R(S_i, t) \tag{2.50}$$

Characteristic Form for the Compressible 2D Euler Equations

Although it is not possible to decouple the original two-dimensional Euler equations in the way demonstrated in section 2.3.2, a different set of differential equations can be derived for each face. By projecting the Euler equations onto an axis system based on the normal ($\vec{n}$) and tangent directions ($\vec{v}$) for a said face, the formulation in equation 2.51 is possible.

$$\frac{\partial Q^{nv}}{\partial t} + \frac{\partial F^n}{\partial n} + \frac{\partial F^v}{\partial v} = 0 \tag{2.51}$$

The vector for the projected Euler dependent variables (Q^{nv}) and the flux vectors for the tangent (F^v) and normal (F^n) directions are shown in equation 2.52.

$$Q^{nv} = \begin{bmatrix} \rho \\ \rho u_n \\ \rho u_v \\ E \end{bmatrix} \qquad F^n = \begin{bmatrix} \rho \\ \rho {u_n}^2 + p \\ \rho u_n u_v \\ u_n(E+p) \end{bmatrix} \qquad F^v = \begin{bmatrix} \rho \\ \rho u_n u_v \\ \rho {u_v}^2 + p \\ u_v(E+p) \end{bmatrix} \tag{2.52}$$

As a local approximation, considering the Q^{nv} quantities to not change along the tangent direction allows the previous system of equations 2.51 to be re-written as equation 2.53.

$$\frac{\partial Q^{nt}}{\partial t} + \frac{\partial F^n}{\partial n} = 0 \tag{2.53}$$

To the previous equation (2.53) one can apply a similar method to that of section 2.3.2.

The Flux Jacobian, for this case, is computed in equation 2.54.

$$\frac{\partial F^n}{\partial Q^{nv}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ (\gamma-1)H - {u_n}^2 - a^2 & (3-\gamma)u_n & (1-\gamma)u_v & \gamma-1 \\ u_n u_v & u_v & u_n & 0 \\ \frac{1}{2}u_n((\gamma-3)H - a^2) & H - (\gamma-1){u_n}^2 & (1-\gamma)u_n u_v & \gamma u_n \end{bmatrix} \tag{2.54}$$

The formula for the total specific enthalpy (H) is shown in equation 2.55.

$$H = \frac{(E+p)}{\rho} = \frac{a^2}{\gamma-1} + \frac{1}{2}V^2 \qquad V^2 = u_n^2 + u_v^2 \tag{2.55}$$

In equation 2.56, the Flux Jacobian is expressed as a product of three matrices (eigenvalue decomposition).

$$\frac{\partial F^n}{\partial Q^n} = \mathbb{X}_R \cdot \Lambda \cdot {\mathbb{X}_R}^{-1} \tag{2.56}$$

As in section 2.3.2, matrix X_R represents the right eigenvectors for the Flux Jacobian matrix arranged in columns and matrix Λ the diagonal matrix of the respective eingevalues (equation 2.57).

$$\mathbb{X}_R = \begin{bmatrix} 1 & 1 & 0 & 1 \\ u-a & u & 0 & u+a \\ u_v & u_v & 1 & u_v \\ H-ua & \frac{1}{2}V^2 & u_v & H+ua \end{bmatrix} \qquad \Lambda = \begin{bmatrix} u_n - a & 0 & 0 & \\ 0 & u_n & 0 & 0 \\ 0 & 0 & u_n & 0 \\ 0 & 0 & 0 & u_n + a \end{bmatrix} \tag{2.57}$$

Following an analogous procedure to that of section 2.3.2, equation 2.58 can be achieved.

$$\frac{\partial(\widetilde{\mathbb{X}}_R^{-1} Q^n)}{\partial t} + \Gamma \cdot \frac{\partial(\widetilde{\mathbb{X}}_R^{-1} Q^n)}{\partial n} = 0 \Rightarrow \frac{\partial Z^n}{\partial t} + \Gamma \cdot \frac{\partial Z^n}{\partial n} = 0 \tag{2.58}$$

2.4 Riemman Problem

A Riemman Problem normally is comprised of an initial value problem with piece-wise constant data separated initially by a single discontinuity in data and ruled by some sort of conservation equation.

2.4.1 Shock Tube

A shock tube, illustrated in Figure 2.3, is characterized by a tube initially filled with gases at different conditions, normally one with higher pressure than the other and separated by a diaphragma. At $t = 0$ the diaphragma is broken.

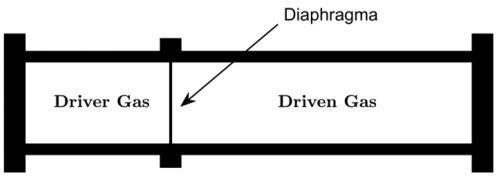

Figure 2.3: Shock Tube.

The setup is of importance to the development of low-cost supersonic testing and thus deserves some interest. This problem is tipically modelled by a Riemann problem governed by the one-dimensional Euler equations (see 2.3.2).

2.4.2 Sod's Shock Tube Problem

The initial left and right initial data can vary greatly but one common configuration is that of Sod's shock sube (SST) problem. The data setup is described by equation 2.59.

$$Q_1 = \begin{bmatrix} 1.0 \\ 0 \\ 2.5 \end{bmatrix} \qquad Q_4 = \begin{bmatrix} 0.125 \\ 0 \\ 0.25 \end{bmatrix} \tag{2.59}$$

Looking at the data, by intuition, the fluid to the right should compress and the fluid to the left should expand. Altough the fluids should reach the same pressure (P_2 and P_3), the density after this expansion is not necessarily the same and thus some sort of discontinuity in this quantity will certainly exist. One can theorise, at first, that a single wave of infinitesimal thickness for compression (compression shock) and another for expansion (expansion shock) will travel along the respective fluid, with respective constant speeds (S_C and S_E). Figure 2.4 serves as illustration for this wave configuration.

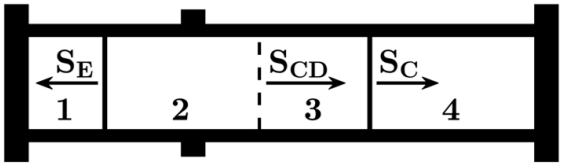

Figure 2.4: Initial supposition for the configuration of the wave structures in SST.

Compression Shock Relations

For the compression shock, one can apply a balance of mass, momentum and energy, tipycally refered to as Rankine-Hugoniot jump conditions, in a coordinate system moving with the shock. The velocities

for this coordinate system are denoted by a circumflex accent. The system of equations is presented in equation 2.60

$$\rho_3\hat{u}_3 = \rho_4\hat{u}_4; \qquad p_3 + \rho_3\hat{u}_3^2 = p_4 + \rho_4\hat{u}_4^2; \qquad \hat{u}_3(E_3 + p_3) = \hat{u}_4(E_4 + p_4) \tag{2.60}$$

In equation 2.61, the velocities in the coordinate system moving with the compression shock ($\hat{u}_4$ and $\hat{u}_4$) are expressed as a function of the particle velocities (u_3 and u_4) and the compression shock velocity (S_C) in the stationary coordinate system.

$$\hat{u}_3 = (u_3 - S_C) \qquad \hat{u}_4 = (u_4 - S_C) \tag{2.61}$$

Using the relations in equation 2.61 and knowing that the particle velocity in region four of the shock tube is null, the relations in equation 2.60 can be simplified, respectively, into the ones in equation 2.62.

$$\rho_3(u_3 - S_C) = -\rho_4 S_C; \qquad p_3 + \rho_3(u_3 - S_C)^2 = p_4 + \rho_4 {S_C}^2; \qquad (u_3 - S_C)(\hat{E}_3 + p_3) = -S_C(\hat{E}_4 + p_4) \tag{2.62}$$

Altough it is not still assured if the passage from state four to state two is isentropic or not, the isentropic relation in equation 2.27 can be applied to each side of the shock wave. The ratio of specific heats can also be assumed constant and the same for both gases. With further algebraic maipulation, the relations in equation 2.62 can be further developed into the ones in equation 2.63.

$$(u_3 - S_C) = -(\tfrac{\rho_4}{\rho_3})S_C; \quad \rho_4 S_C^2(\tfrac{\rho_4}{\rho_3} - 1) = p_4 - p_3; \quad (u_3 - S_C)(\tfrac{\gamma p_3}{\gamma - 1} + \tfrac{\rho_3(u_3 - S_C)^2}{2}) = -S_C(\tfrac{\gamma p_4}{\gamma - 1} + \tfrac{\rho_4 S_C^2}{2}) \tag{2.63}$$

Concerning only the second relation in equation 2.63, dividing both sides by γP_4 and using the algebraic expressions for mach number ($M = u/a$) and speed of sound in an ideal gas ($a = \sqrt{\gamma p/\rho}$) results in equation 2.64.

$$M_C^2(\frac{\rho_4}{\rho_3} - 1) = \frac{1}{\gamma}(1 - \frac{p_3}{p_4}) \tag{2.64}$$

Considering that there are no heat losses to the walls of the shock tube, adiabatic gas law, stated in equation 2.65, can be applied.

$$\frac{\rho_3}{\rho_4} = (\frac{p_3}{p_4})^{\frac{1}{\gamma}} \tag{2.65}$$

Combining equation 2.65 into equation 2.64 results in equation 2.66

$$M_C^2 = \frac{1}{\gamma} \cdot (1 - \frac{p_3}{p_4}) \cdot ((\frac{p_3}{p_4})^{-\frac{1}{\gamma}} - 1)^{-1} \tag{2.66}$$

Using the definition of Mach number and equation 2.66, shock speed can be expressed by equation 2.67.

$$S_C = a_4 \cdot \frac{1}{\gamma^{\frac{1}{2}}} \cdot (1 - \frac{p_3}{p_4})^{\frac{1}{2}} \cdot ((\frac{p_3}{p_4})^{-\frac{1}{\gamma}} - 1)^{-\frac{1}{2}} \tag{2.67}$$

Equation 2.67 and the first relation from equation 2.63 can be used to obtain an expression, shown in equation 2.68, for the particle velocity in the upstream of the shock.

$$u_3 = a_4 \cdot (1 - \frac{p_3}{p_4}^{-\frac{1}{\gamma}}) \cdot \frac{1}{\gamma^{\frac{1}{2}}} \cdot (1 - \frac{p_3}{p_4})^{\frac{1}{2}} \cdot ((\frac{p_3}{p_4})^{-\frac{1}{\gamma}} - 1)^{\frac{1}{2}} \tag{2.68}$$

The use of adiabatic gas law in equation 2.65 and the definition of sound speed in an ideal gas allows for equation 2.69.

$$\frac{a_4}{a_3} = \sqrt{(\frac{P_4}{P_3})^{\frac{\gamma-1}{\gamma}}} \tag{2.69}$$

With employment of equation 2.69 into equation 2.68, the Mach number ($M_3 = u_3/a_3$) for the fluid in the upstream of the moving shock can be expressed as equation 2.70.

$$M_3 = (\frac{P_3}{P_4})^{\frac{1-\gamma}{2\gamma}} \cdot (1 - \frac{P_3}{P_4}^{-\frac{1}{\gamma}}) \cdot \frac{1}{\gamma^{\frac{1}{2}}} \cdot (1 - \frac{P_3}{P_4})^{\frac{1}{2}} \cdot ((\frac{P_3}{P_4})^{-\frac{1}{\gamma}} - 1)^{-\frac{1}{2}} \tag{2.70}$$

Compression Shock Hypothesis

Before one goes further and applies the previously computed relations between states 3 and 4, the compression shock hypothesis should be put to the test.

The Second Law of Thermodynamics states that the total entropy of an isolated system (such as the shock tube) can never decrease with time. Assuming a perfect gas, equation 2.71 expresses the change in entrophy from the passage of state 3 into state 4 ($(\Delta\mathfrak{z})_{4\to3}$).

$$(\Delta\mathfrak{z})_{4\to3} = R \cdot \ln(\frac{p_3}{p_4}^{\frac{1}{\gamma-1}} \cdot \frac{\rho_3}{\rho_4}^{\frac{\gamma}{\gamma-1}}) \tag{2.71}$$

Assuming that there is no heat transfer on the walls of the shock tube, the adiabatic gas law in equation 2.65 can be used to simplify equation 2.71 into equation 2.72.

$$(\Delta\mathfrak{z})_{4\to3} = R \cdot \ln(\frac{p_3}{p_4}^{\frac{2}{\gamma-1}}) \tag{2.72}$$

Knowing that pressure downstream of the moving shock is inferior to the one upstream ($p_4 > p_3$), it becomes apparent, when looking at equation 2.72, that entropy increases from state 4 to state 3 and that the compression shock hypothesis does not violate the Second Law of Thermodynamics. A moving shock configuration for compression is further strenghtened by the intersection of characteristic curves associated with the Z_3 quantity in the vicinity of the shock wave, as illustrated in Figure 2.5.

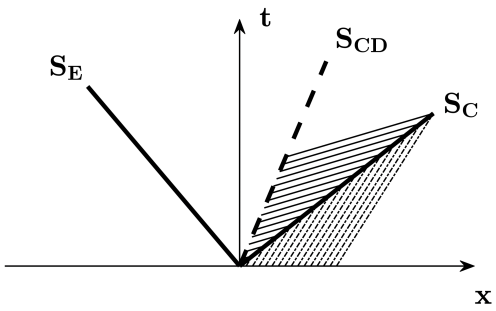

Figure 2.5: Z_3 characteristic curves from states three (fine solid line) and four (fine intermittent line) intersecting on the shock wave.

The local characteristic quantity Z_3 from each side (states 3 and 4) travels along a constant characteristic curve, as explained in section 2.3.2 and again stated by equation 2.73.

$$\frac{dx}{dt} = u + a \Rightarrow Z_3 = constant \tag{2.73}$$

Through the relations obtained in section 2.4.2, it can be proved that the information for the Z_3 characteristic quantity travels faster downstream of the moving shock. Equation 2.74 translates this statement mathematically.

$$u_3 + a_3 > u_4 + a_4 \tag{2.74}$$

In consequence of equation 2.74, the upstream and downstream Riemman invariants associated with the Z_3 would intersect in the vicinity of the moving shock. One of these intersections, stated in equation 2.75, directly translates to a discontinuity in the Euler dependent variables.

$$Z_3 = cte \Rightarrow u + \frac{2 \cdot a}{\gamma - 1} = cte \tag{2.75}$$

The counter-proposition of a single moving shock would be that of a series of infinetisimal isentropic compression waves ($\frac{p_3}{p_4} \longrightarrow 1$). That hypothesis gets easily shunned as the compression waves would all coalesce due to the sound speed increasing with each successive adiabatic compression. Consequently, equation 2.67 forces the propagation speed for each wave front to be greater than that of the one downstream to it.

Relations for the contact discontinuity

Applying the Rankine-Hugoniot jump conditions to the contact discontinuity on a solidary axis coordinate system, one gets the system described by eqaution 2.76.

$$\rho_2\hat{u}_2 = \rho_3\hat{u}_3; \qquad p_2 + \rho_2\hat{u}_2^2 = p_3 + \rho_3\hat{u}_3^2; \qquad \hat{u}_2(\hat{E}_2 + p_2) = \hat{u}_3(\hat{E}_3 + p_3) \tag{2.76}$$

The fact that the particle velocities in the coordinate system moving with shock can be described as a function of the ones in the stationary one along with the speed of the contact discontinuity ($\hat{u}_2 = (u_2 - S_{CD})$ and $\hat{u}_3 = (u_3 - S_{CD})$) allows for some algebraic manipulation that combined with the equality in downstream and upstream pressures ($p_2 = p_3$) results in the simplification of the system in equation 2.76 into the relations in equation 2.77.

$$u_2 = u_3; \qquad u_3 = S_{CD} \tag{2.77}$$

Impossibility of an expansion shock

In analogy with equation 2.72, the entropy change for an adiabatic and inviscid flow in the expansion shock can be described by equation 2.78.

$$(\Delta\mathfrak{s})_{1\to2} = R \cdot \ln[\frac{p_2}{p_1}^{\frac{2}{\gamma-1}}] \tag{2.78}$$

When looking at equation 2.78, it becomes clear that as pressure in the downstream of the expansion shock is greater than that in the upstream ($p_1 > p_2$), the change in entropy is forcefully negative. This conclusion violates the Second Law of Thermodynamics and, thus, serves as proof that modelling an expansion wave as a moving shock physically is incoherent and that an expansion process must be isentropic. Furthermore, when looking at equation 2.79, obtained by the same process required for equation 2.66, the shock speed mach number (M_E for the expansion) tends to zero as the the ratio between upstream and downstream pressures tends to one ($P_2/P_1 \to 0$) and turns negative to any ratio lower than one. This impossibility leads to the conclusion that a moving shock wave for the Euler equations can only compress.

$$M_C^2 = \frac{1}{\gamma} \cdot (1 - \frac{p_3}{p_4}) \cdot ((\frac{p_3}{p_4})^{-\frac{1}{\gamma}} - 1)^{-1} \tag{2.79}$$

Another wave configuration must be sought for the expansion process. An expansion fan with infinetisimal pressure jumps would respect the isentropy condition as the ratio between pressure would tend to one. Figure 2.6 illustrates the new proposed wave structure configuration.

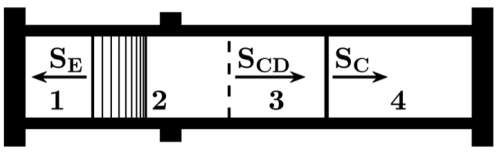

Figure 2.6: Final configuration for the wave structures in SST.

Relations for the expansion wave

As the expansion process is isentropic, one can use isentropic flow equations to establish dependence between the states one and two. Equation 2.80 takes into account the quiescence of the fluid in state one and develops an expression for the mach number upstream of the expansion fan (M_2).

$$u_1 = M_1 = 0 \Rightarrow \frac{p_2}{p_1} = (1 + \frac{\gamma+1}{2}M_2^2)^{-\frac{\gamma}{\gamma-1}} \Rightarrow M_2 = (\frac{2}{\gamma+1})^{\frac{1}{2}}((\frac{p_2}{p_1})^{\frac{1-\gamma}{\gamma}} - 1)^{\frac{1}{2}} \tag{2.80}$$

The relation between sound speeds in equation 2.69 can be used along equation 2.80 to achieve an expression for particle speed in state two (u_2). This process is demonstrated by equation 2.81.

$$a_2 = a_1 \cdot \frac{a_2}{a_1} = a_1 \cdot (\frac{p_2}{p_1})^{\frac{\gamma-1}{2\gamma}} \Rightarrow u_2 = a_1 \cdot (\frac{p_2}{p_1})^{\frac{\gamma-1}{2\gamma}}(\frac{2}{\gamma+1})^{\frac{1}{2}}((\frac{p_2}{p_1})^{\frac{1-\gamma}{\gamma}} - 1)^{\frac{1}{2}} \tag{2.81}$$

Altough not mathematically proven here (see [5]), the multiple infinitesimal expansion waves coincide with local Z_1 characteristic curves and their propagation speed is the same ($u-a$). As the speed of sound (a) decreases and velocity (u) increases along the expansion fan (from state four to state one), the infinitesimal wave speed should decrease, giving a diverging character to the expansion fan, illustrated by figure 2.7.

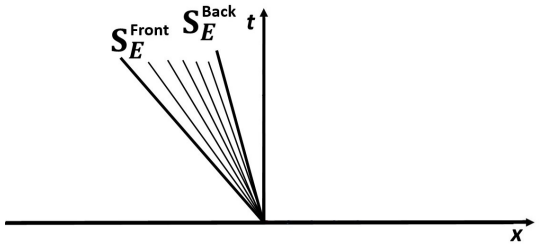

Figure 2.7: Waves structures, including the expansion fan, development in x coordinate with time (t).

As the particle velocity in state 1 is null, the front of the expansion fan propagates at the local sound speed (a_1).

Pressure in States 2 and 3

The pressures in states two and three have been proven in section 2.4.2 to be equal and can be calculated trough the use of previous obtained relations for the moving shock (section 2.4.2) and expansion fan (section 2.4.2). Equation 2.82 is obtained by using equations 2.68 and 2.81 and taking into account the relation in equation 2.77 that states the equality of the particle velocities at the upstream and downstream of the contact discontinuity ($u_2 = u_3$).

$$a_1 \cdot (\frac{p_2}{P_1})^{\frac{\gamma-1}{2\gamma}}(\frac{2}{\gamma+1})^{\frac{1}{2}}((\frac{P_2}{P_1})^{\frac{1-\gamma}{\gamma}} - 1)^{\frac{1}{2}} = a_4 \cdot (1 - \frac{P_2}{P_4}^{-\frac{1}{\gamma}}) \cdot \frac{1}{\gamma^{\frac{1}{2}}} \frac{(1-\frac{P_2}{P_4})^{\frac{1}{2}}}{((\frac{P_2}{P_4})^{-\frac{1}{\gamma}} - 1)^{\frac{1}{2}}}; \tag{2.82}$$

Equation 2.82 can be solved iteratively for p_2, which can then be used to compute all other required quantities in states two and three.

Quantities across the expansion fan

The expansion fan can be approximately discretized by a finite number of expansion waves that exist inside the expansion fan region with propagation speeds between those of the expansion front and the back ($S_E^{Front} > S_E(\beta) > S_E^{Back}$). By equally dividing the range of propagation speed into N_β number of waves speeds, one gets the model illustrated in figure 2.8.

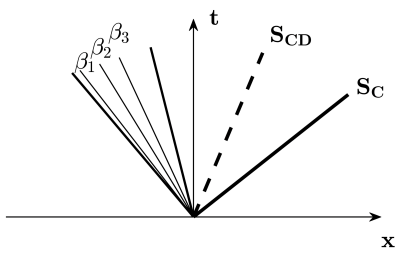

Figure 2.8: Detail over the discretization of the inner region of the expansion fan with $N_\beta = 3$ number of waves.

As stated in [5], through the expansion fan, the $\mathbb{J}_3$ Riemann invariant remains constant, although its characteristic curve gets altered. Therefore, for some point $(x, t^{'})$ inside the expansion wave, the $\mathbb{J}_3$ Riemann invariant is the same as the one in state four. Equation 2.83 translates this equality.

$$\mathbb{J}_3 = a_4 + \frac{2 \cdot u_4}{\gamma - 1} = a(x^{'}, t^{'}) + \frac{2 \cdot u(x^{'}, t^{'})}{\gamma - 1} \tag{2.83}$$

As each discrete infinetisimal expansion wave (β) propagates a $\mathbb{J}_1$ Riemann invariant and intersects the same $\mathbb{J}_3$ Riemann invariant throughout its path, the relation in equation 2.84 may be established.

$$\begin{cases} a(\beta) + \frac{2 \cdot u(\beta)}{\gamma - 1} = cte \\ a(\beta) - \frac{2 \cdot u(\beta)}{\gamma - 1} = cte \end{cases} \Rightarrow u(\beta) = cte; \;\; a(\beta) = cte \tag{2.84}$$

The propagation speed of a infinetisimal expansion wave is the same as the characteristic speed of the Riemann invariant that it propagates ($u^{'} - a^{'}$). Therefore, for a prescribed constant propagation speed of a given discrete infinitesimal expansion wave ($S_E(\beta)$), the system in equation 2.85 serves to discover the values of particle speed and local sound speed along that wave's path.

$$a(\beta) + \frac{2 \cdot u(\beta)}{\gamma - 1} = a_4; \quad u(\beta) - a(\beta) = S_E(\beta) \tag{2.85}$$

If one wishes to discover the local sound and particle speed at a given point and time, he can use equation 2.86 to discover the propagation speed of the discrete expansion wave that intersects it and

the system in equation 2.85 for the wanted quantities.

$$x^{'} = x_0 + S_E(B) \cdot t^{'} \tag{2.86}$$

From the local sound and particle speed, all other quantities can be calculated by resorting to insentropic relations.

The reflection of shock and expansion waves on the walls of the shock tube will not be covered as the work done throughout the thesis does not require it.

2.5 Approximate Riemann Solvers

Computing analytical solutions of Riemann problems is often too strenuous to provide sufficient computacional efficiency in numerical schemes that require an adequate solution. As such, approximate solvers are used instead. In this section, three types of approximate Riemman Solvers will be discussed and then compared on section 3.1.6 to decide on which to use for the proposed WENO scheme.

2.5.1 Harten Lax Van-Leer (HLL)

In the HLL Riemann solver, the complicated wave structure of the Sod-like configuration is substituted by two travelling waves: one for representing the front of the expansion fan and the other the shock wave. Both are treated mathematically as shock waves. These two waves separate an averaged state (Q_{HLL}) in the middle from the left and right initial states.The cell interface is represented at $x = 0$, so the flux of interest is the one that crosses the t-axis.

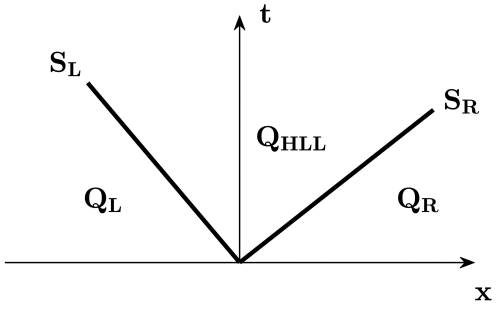

Figure 2.9: Possible configuration for the wave structures in the HLL approximate Riemann solver.

The intermediate state (Q_{HLL}), represented in equation 2.87, is a spatial average of the analytic solution ($Q(x,t)$) at any given time, between the two waves.

$$Q_{HLL} = \frac{1}{t(S_R - S_L)} \int_{tS_L}^{tS_R} Q(x,t)\partial x \tag{2.87}$$

A balance of mass, momentum and energy (Rankine-Hugoniot) at the wave-fronts gives the relations in equation 2.88.

$$F_{HLL} = F_R + S_R(Q_{HLL} - Q_R)\ ; \qquad F_{HLL} = F_L + S_L(Q_{HLL} - Q_L) \tag{2.88}$$

From the previous two relations in 2.88, equation 2.89 can be obtained.

$$Q_{HLL} = \frac{(F_R - F_L) + S_L Q_L - S_R Q_R}{S_L - S_R} \tag{2.89}$$

Substituing the previous expression for Q_{HLL} (2.89) in any of the relations in equation 2.88 results in equation 2.90.

$$F_{HLL} = \frac{F_L S_R - F_R S_L + S_L S_R (Q_R - Q_L)}{S_R - S_L} \tag{2.90}$$

It is not ensured that the flux at the t-axis, which is the one of interest, will be the HLL one. For example, if either one the left or right state is supersonic, the configuration for the wave structure differs from the one in figure 2.9. Figure 2.10 illustrates the two other possible wave configurations of interest.

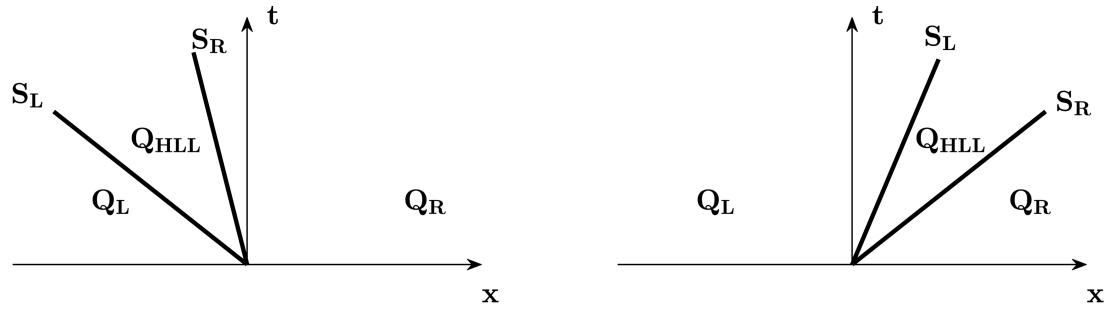

Figure 2.10: Configuration for the wave structures for a HLL approximate Riemann solver in case of (left) left-going supersonic flow and (right) right-going supersonic flow.

With the proper upwinding, the complete HLL flux is defined by equation 2.91.

$$F(S_{i+j}) = \begin{cases} F_L & , S_L \geq 0 \\ F_{HLL} = \frac{F_R S_L - F_L S_R - S_L S_R (Q_L - Q_R)}{S_R - S_L} & , S_L \leq 0 \cap S_R \geq 0 \\ F_R & , S_R \leq 0 \end{cases} \tag{2.91}$$

It is important to note that, as stated in equation 2.92, for this approximated Riemann Solver, the flux computed at the t-axis is not equivalent to the one computed directly using the averaged state Euler variables (Q_{HLL}).

$$F_{HLL} \neq F(Q_{HLL}) \tag{2.92}$$

Other important characteristics of this approximate Riemann solver are that the entropy condition is not respected because expansion fans are substitued by a single shock and, therefore, entropy can diminish (violating the second law of thermodynamics). The contact discontinuity is not taken into account resulting in difusive behaviour for this wave structure, when applied to numerical schemes.

2.5.2 Harten Lax-van Leer Contact (HLLC)

The HLLC approximate Riemann solver is an improvement of the HLL as the contact discontinuity is restored by adding a third party to the wave structure, shown in figure 2.11.

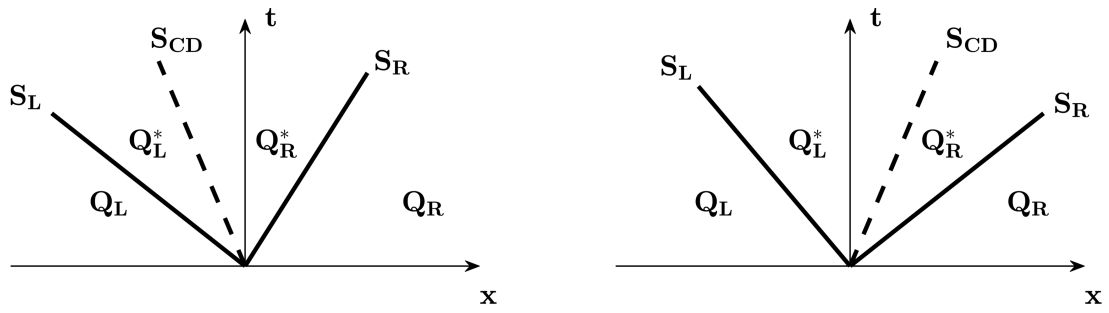

Figure 2.11: Two possible configurations for the wave structure in the HLLC Riemann solver.

Similarly to the HLL Riemann solver, the Q_R^* and Q_L^* states are a spatial average of the analytic solution between their respective waves. The integral form for the two states is shown in equation 2.93.

$$Q_R^* = \frac{1}{t(S_R - S_{CD})} \int_{tS_{CD}}^{tS_R} Q(x,t)\partial x \; ; \qquad Q_L^* = \frac{1}{t(S_{CD} - S_L)} \int_{tS_{CD}}^{tS_R} Q(x,t)\partial x \tag{2.93}$$

Knowing that all previous equations from the HLL Riemann Solver still hold true, one can rewrite Q_{HLL} as a function of the Q_R^* and Q_L^* averaged states. In equation 2.94, the integral of the exact solution ($Q(x,t)$) between the left and right shocks (S_L and S_R) is decomposed into a sum of integrals between the left shock and contact discontinuity and the contact discontinuity and the right shock.

$$\int_{tS_L}^{tS_R} Q(x,t) \;\; \partial x = \int_{tS_L}^{tS_{CD}} Q(x,t) \;\; \partial x + \int_{tS_C D}^{tS_R} Q(x,t)\partial x \tag{2.94}$$

Comparing equations 2.94, 2.87 and the ones in 2.93, one can obtain equation 2.95

$$Q_{HLL} = \frac{(S_{CD} - S_L)}{(S_R - S_L)} \cdot Q_L^* + \frac{(S_R - S_{CD})}{(S_R - S_L)} \cdot Q_L^* \tag{2.95}$$

Equations 2.95 and 2.89 can be used to obtain equation 2.96.

$$(S_{CD} - S_L)Q_L^* + (S_R - S_{CD})Q_R^* = (F_L - F_R) + S_R Q_R - S_L Q_L \tag{2.96}$$

By applying Rankine-Hugoniot jump-conditions on the different waves, the consistency relations in equation 2.97 are computed.

$$F_R^* = F_R + S_R \cdot (Q_R^* - Q_R) \; ; \quad F_L^* = F_R^* + S_{CD} \cdot (Q_L^* - Q_R^*) \; ; \quad F_L^* = F^L + S_L \cdot (Q_L^* - Q_L) \tag{2.97}$$

From Rankine-Hugoniot conditions at the contact discontinuity (S_{CD}) (second relation in equation 2.97), which has equal left and right pressure, the relations in equation 2.98 can be obtained.

$$p_L^* = p_R^* \qquad u_L^* = u_R^* \qquad S_{CD} = u_R^* \tag{2.98}$$

The first equalities in equation 2.98 can be used in conjunction with momentum conservation in the consistency relations (2.97) to define the contact discontinuity wave speed (S_{CD}) as a function of left

and right undisturbed state quantities (Q_L and Q_R), which are prescribed. Equation 2.99 shows the final relation between the forementioned quantities.

$$S_{CD} = \frac{p_R - p_L + \rho_L u_L(S_L - u_L) - \rho_R u_R(S_R - u_R)}{\rho_L(S_L - u_L) - \rho_R(S_R - u_R)} \tag{2.99}$$

Using equations 2.97, 2.98 and 2.96, one can define the averaged states that exist between the shock wave and the expansion wave front. Taken from [5], equation 2.100 gives a mathematical expression for these averaged states.

$$Q_K^* = \rho_K \frac{S_K - u_k}{S_K - S_{CD}} \begin{bmatrix} 1 \\ S_{CD} \\ Q_K(:) \\ \frac{E_K}{\rho_K} + (S_{CD} - u_K)[S_{CD} + \frac{p_K}{\rho_K(S_K - u_K)}] \end{bmatrix} \tag{2.100}$$

The index K in equation 2.100 can represent either the state either between the shock wave and the contact discontinuity ($K = R$) or between the contact discontinuity and the expansion front ($K = L$). The third matrix entry ($Q_K(:)$) represents a random passive scalar quantity. For two dimensional schemes, tangential velocity at the face is often treated like a passive scalar.

The fluxes in the star region can be completely defined by using equation 2.100 along with the first and third relations for the consistency conditions in equation 2.97. These relations are repeated in equation 2.101.

$$F_L^* = F_L + S_L(Q_L^* - Q_L)\,; \qquad F_R^* = F_R + S_R(Q_R^* - Q_R) \tag{2.101}$$

As with the HLL Riemann solver, the flux of interest is the one on the t-axis. The wave structure can have different configurations, and therefore, the solver needs to again take into account some different scenarios. It should be stated, however, that neither the HLL and HLLC Riemann solvers are equipped from default to deal with any random wave configuration Figure 2.12 illustrates the two remaining wave configurations, not shown in figure 2.11, that are encompassed in a typical HLLC Riemann solver.

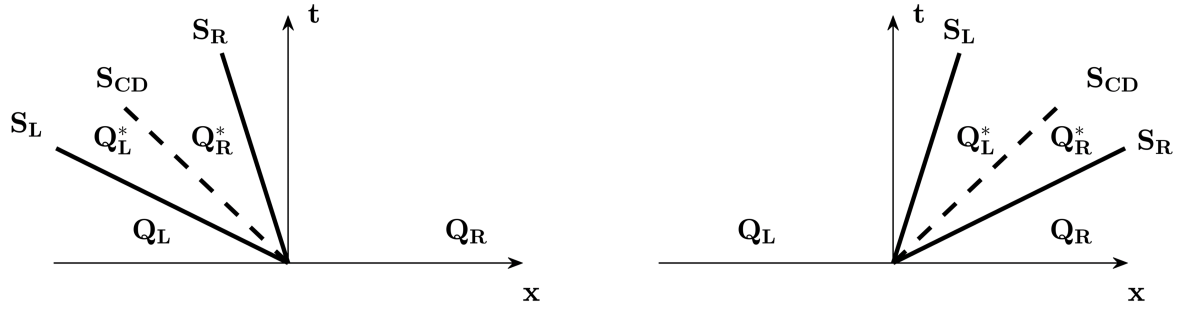

Figure 2.12: Wave structures configurations for (left) right-going supersonic flow and (right) right-going supersonic flow in the HLLC Riemann solver.

The the HLLC flux is defined in equation 2.102.

$$F(S_{ij}) = \begin{cases} F_L & , S_L \geq 0 \\ F_L^* & , S_L \leq 0 \cap S_{CD} \geq 0 \\ F_R^* & , S_R \geq 0 \cap S_{CD} \leq 0 \\ F_R & , S_R \leq 0 \end{cases} \tag{2.102}$$

As with the HLL solver, the HLLC solver does not, by default, preserve entropy across expansion regions.

2.5.3 Wave Speed Estimates

As the correct wave speeds are impossible to calculate without the analytical solution. Wave speed estimates are required for both the HLL and HLLC approximate Riemann solvers. The simplest form of wave estimates comprise of the eingenvalues of the characteristic form of the euler equations in the left and right states:

$$\begin{aligned} S_L &= min(u_L - a_L, u_R - a_R) \\ S_R &= max(u_L + a_L, u_R + a_R) \end{aligned} \tag{2.103}$$

The estimate used in this thesis will be pressure based as it is found by [5] to be sufficiently accurate while preserving simplicity.

Pressure-Based Speed Estimates

In this kind of wave speed estimates, an approximation for the pressure in the star region is first given and then the wave speeds are calculated. The method here described is the same as that of Toro (see [5]).

$$p_R^* = p_L^* = \frac{1}{2}(p_L + p_R) - \frac{1}{2}(u_R - u_L)\overline{\rho a} \tag{2.104}$$

$$\overline{\rho} = \tfrac{1}{2}(\rho_L + \rho_R) \qquad \overline{a} = \tfrac{1}{2}(a_L + a_R) \tag{2.105}$$

The wave speed estimates are given by:

$$S_L = u_L - a_L q_L \qquad S_R = u_R - a_R q_R \tag{2.106}$$

$$q_K = \begin{cases} 1 & , p^* \leq p_K \\ [1 + \frac{\gamma+1}{\gamma}(p^*/p_K - 1)]^{\frac{1}{2}} & , p^* > p_K \end{cases} \tag{2.107}$$

The parameter q_K is set on basis of checking if the wave front tries to emulate a rarefaction fan front or a shock wave. If the pressure on the star region (p^*) is less than that of said side (p_K), the wave emulates the front of a rarefaction fan and will propagate with a speed of $u_L - a_L$.On the other hand, if the pressure in the star region is greater than that of said side, a shock wave will be present. In this scenario, Rankine-Hugoniot jump conditions along with the pressure estimate can be used to determine the shock-wave propagation speed, summarized in the above expression.

2.5.4 Riemann Solver of Roe

The Roe approach for solving a Riemann problem for the Euler equations consists of substituting the one-dimensional set of equations by an approximate linearized version, presented in section 2.3.2, and then solving the approximated problem exactly. Equation 2.108 defines this methodology.

$$\begin{cases} \frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} = 0 \\ Q(x,0) = Q_L \quad , x < 0 \\ Q(x,0) = Q_R \quad , x \geq 0 \end{cases} \quad \approx \quad \begin{cases} \frac{\partial Q}{\partial t} + \mathbb{A}\frac{\partial Q}{\partial x} = 0 \\ Q(x,0) = Q_L \quad , x < 0 \\ Q(x,0) = Q_R \quad , x \geq 0 \end{cases} \tag{2.108}$$

Matrix $\widetilde{\mathbb{A}}(Q_L, Q_R)$ in equation 2.108 is an approximation for the exact Flux Jacobian ($J = \frac{\partial F}{\partial Q}$), and must respect a set of conditions:

- The system must remain hyperbolic. This means that matrix $\widetilde{\mathbb{A}}(Q_L, Q_R)$ must have only real eigenvalues ($\tilde{\lambda}_1 \leq ... \leq \tilde{\lambda}_m \ \in \mathbb{R}$) and a complete set of linearly independent eigenvectors ($\tilde{\nu}_1, \tilde{\nu}_2, ..., \tilde{\nu}_m$).
- Matrix $\widetilde{\mathbb{A}}$ must have consistency with the exact flux jacobian ($\widetilde{\mathbb{A}}(Q, Q) = J(Q)$).
- Conservation across discontinuities must be preserved ($F(Q_R) - F(Q_L) = \widetilde{A}(Q_R - Q_L)$).

For the complete Roe solver all these criteria were fulfilled through the use of Roe averaged values (for more details see [5]). The complete form for the approximated Flux Jacobian is shown in equation 2.109.

$$\tilde{\hat{\mathbb{A}}} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)\tilde{u}^2 & (3-\gamma)\tilde{u} & \gamma - 1 \\ \frac{1}{2}(\gamma-1)\tilde{u}^3 - \tilde{H}\tilde{u} & \tilde{H} - (\gamma-1)\tilde{u}^2 & \gamma\tilde{u} \end{bmatrix} \tag{2.109}$$

The necessary Roe averaged quantities, denoted by a tilde, are computed with the right (Q_R) and left (Q_L) states, as shown in equation 2.110.

$$\tilde{u} = \frac{\rho_L u_L + \rho_R u_R}{\rho_L + \rho_R} \qquad \tilde{H} = \frac{\rho_L H_L + \rho_R H_R}{\rho_L + \rho_R} \qquad \tilde{a} = ((\gamma-1)(\tilde{H} - \tfrac{1}{2}\tilde{u}^2))^{\frac{1}{2}} \tag{2.110}$$

Using the same methodology as section 2.3.2, the characteristic form for the new differential equation can be obtained. This process requires calculating the right eigenvectors and eigenvalues for the approximated Flux Jacobian ($\widetilde{\mathbb{A}}$). Equation 2.111 displays the right eigenvectors arranged in columns, in matrix $\widetilde{R}$), and the respective eigenvalues in a diagonal matrix ($\tilde{\Gamma}$).

$$\tilde{R} = \begin{bmatrix} 1 & 1 & 1 \\ \tilde{u} - \tilde{a} & \tilde{u} & \tilde{u} + \tilde{a} \\ \tilde{H} - \tilde{u}\tilde{a} & \frac{1}{2}\tilde{V}^2 & \tilde{H} + \tilde{u}\tilde{a} \end{bmatrix} \qquad \tilde{\Gamma} = \begin{bmatrix} \tilde{u} - \tilde{a} & 0 & 0 \\ 0 & \tilde{u} & 0 \\ 0 & 0 & \tilde{u} + \tilde{a} \end{bmatrix} \tag{2.111}$$

The fully decoupled form for the approximated set of equations is shown in 2.112.

$$\frac{\partial(\widetilde{R}^{-1}Q)}{\partial t} + \Gamma \cdot \frac{\partial(\widetilde{R}^{-1}Q)}{\partial x} = 0 \qquad \equiv \qquad \frac{\partial \widetilde{Z}}{\partial t} + \tilde{\Lambda} \cdot \frac{\partial \widetilde{Z}}{\partial x} \tag{2.112}$$

Each initial value of Z is then propagated with a speed equal to the respective eigenvalue. As the left and right states of a given quantity Z are different but propagate at the same speed, the discontinuity between both side propagates at that same speed. Figure 2.13 illustrates the concept.

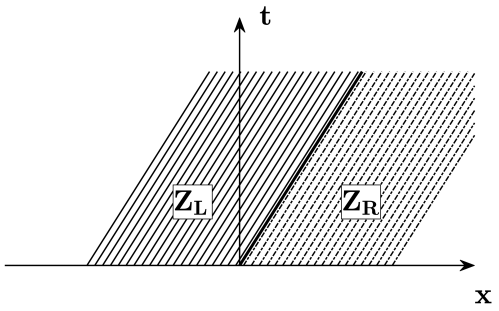

Figure 2.13: Characteristic curves for a given quantity Z in the approximated problem with the discontinuity between the two states highlighted by a thicker black line

The values for the characteristic variables at the cell interface can be easily known and are completely defined by equation 2.113.

$$\tilde{Z}_1(0,t) = \begin{cases} \tilde{Z}_{L_1}, \tilde{u} - \tilde{a} \geq 0 \\ \tilde{Z}_{R_1}, \tilde{u} - \tilde{a} < 0 \end{cases} \quad ; \quad \tilde{Z}_2(0,t) = \begin{cases} \tilde{Z}_{L_2}, \tilde{u} \geq 0 \\ \tilde{Z}_{R_2}, \tilde{u} < 0 \end{cases} \quad ; \quad \tilde{Z}_3(0,t) = \begin{cases} \tilde{Z}_{L_3}, \tilde{u} + \tilde{a} \geq 0 \\ \tilde{Z}_{R_3}, \tilde{u} + \tilde{a} < 0 \end{cases} \tag{2.113}$$

The original variables at the face of interest ($x = 0$) can then be obtained, by converting the computed characteristic quantities back to the original Euler variables, as done in equation 2.114:

$$\tilde{Q}(0,t) = \tilde{R}\tilde{Z}(0,t) \tag{2.114}$$

The intercell flux cannot be directly calculated from these quantities (equation 2.114), as propper upwinding is not ensured. For example, if the left state is supersonic, the eigenvalues of the approximated problem are not guaranteed to all be positive. A consistency condition must then be used. A possible choice for the exact solution is presented in equation 2.115.

$$\int_0^{TS_R} Q(x,t) \;\; \partial x = T(F(0,t) - F_R + S_R Q_R) \tag{2.115}$$

The computed flux ($T(\tilde{F}(S_{ij}))$ from the approximate problem should also respect the similar consistency condition in equation 2.116.

$$\int_0^{TS_R} \tilde{Q}(x,t) \partial x = T(\tilde{F}_{i+\frac{1}{2}} - F_R + S_R Q_R) \tag{2.116}$$

One can use the solution for the approximated problem in equation 2.113 and establish the useful

approximation shown in equation 2.117.

$$\int_0^{TS_R} Q(x,t)\partial x \approx \int_0^{TS_R} \tilde{Q}(x,t)\partial x \tag{2.117}$$

Using equations 2.117, 2.116 and 2.114 the intercell flux from the original Riemman problem can be expressed as equation 2.118.

$$F(S_{i+\frac{1}{2}}) = F_R - \widetilde{A}(Q_R - \tilde{Q}_{i+\frac{1}{2}}) \tag{2.118}$$

A more computationally-friendly approach is present in [5].

2.6 Explicit Runge-Kutta Time Integration

The Runge-Kutta family of explicit methods are commonly used for temporal discretization in ordinary diferential equations such as the finite volume formulation of the Euler equations. Equation 2.119 states the typical formulation for the time of the averaged quantities in a target cell (S_i) as a function of the cell averaged quantities of the other cells discretizing the domain.

$$\frac{d\overline{Q}(S_i)}{dt} = \frac{1}{A(S_i)} \cdot R_i(\overline{Q}(S_1,t), \overline{Q}(S_2,t).., \overline{Q}(S_N,t)) \tag{2.119}$$

The exact residual (R) term is usually substituted by an approximation ($\tilde{R}$) using a particular flux scheme such as Godunov or WENO.

In CFD, it is important to know the cell averaged quantities in discrete instances of time and proper time integration of the residual term is required. As exact analytical integration of the residual is often strenuous or impossible, some sort of approximated explicit time integration can be used.

It is also of interest to state that one seeks appropriate aproximate integration of the finite volume formulation with the aproximated residual function ($\tilde{R}$) and not that of the original FV formulation (R). In the case of the flux scheme gathering the necessary conditions (monotonocity,convergence,stability, etc.) the approximate residual should match the exact residual, as the spatial discretization gets finer.

For simplicity, the averaged quantities of the cells discretizing the domain at a given time will be represented by the term $\overline{Q}_t$, as defined in equation 2.120.

$$\overline{Q}(t) = \overline{Q}(S_1,t), \overline{Q}(S_2,t), \ldots, \overline{Q}(S_N,t) \tag{2.120}$$

Given an initial field of quantities ($Q(x,t_0)$) and respective cell averaged values $\overline{Q}(t_0)$, the next time-step cell averaged quantities for a given target cell ($\overline{Q}(S_i, t_0+\Delta t)$) can be expressed by equation 2.121.

$$\overline{Q}(S_i, t_0+\Delta t) = \overline{Q}(S_i,t_0) + \int_{t_0}^{t_0+\Delta t} \frac{d\overline{Q}(S_i,t)}{dt}\, \partial t \approx \overline{Q}(S_i,t_0) + \frac{1}{A(S_i)} \int_{t_0}^{t_0+\Delta t} \tilde{R}_i(\overline{Q}(t_0))\, \partial t \tag{2.121}$$

Runge-Kutta methods of a given order of accuracy (RK^r) allow for approximate integration with com-

putational applicability. Equation 2.122 shows the approximation of time integration of the approximate residual by such methods.

$$\int_{t_0}^{t_0+\Delta t} \tilde{R}_i(\overline{Q}(t_0))\ \partial t \approx RK^r(\overline{Q}(t), \Delta t) \tag{2.122}$$

The simplest form of explicit time integration is the Euler Method, equivalent to the first-order Runge Kutta Method. This variant uses only the information of the time derivative at the initial state and assumes it as constant throughout the time step. An order of accuracy for this method can be obtained by deriving an expression for the the exact and approximate values of the averaged property for a given target cell in the next time iteration ($\overline{Q}(S_i, t_0 + \Delta t)$). In the context of this process, the exact value can be expressed through a Taylor expansion over the previous time iteration (t_0), as expressed by equation 2.123.

$$\overline{Q}(S_i, t_0 + \Delta t) = \overline{Q}(S_i, t_0) + \frac{1}{A(S_i)}(\frac{d\overline{Q}(S_i, t_0)}{dt} \cdot \Delta t + O(\Delta t^2)) \tag{2.123}$$

The Euler method approximates the forementioned cell averaged quantities ($\overline{Q}(S_i, t_0 + \Delta t)$) by the expression in equation 2.124.

$$\overline{Q}(S_i, t_0 + \Delta t) \approx \overline{Q}(S_i, t_0) + \frac{1}{A(S_i)}(R_i(\overline{Q}(t_0)) \cdot \Delta t) \tag{2.124}$$

In equation 2.125, the expression in equation 2.124 is subtracted from that of equation 2.123 so the truncation error (τ), and, thus, the order of accuracy of the Euler method can be known.

$$\tau = \frac{1}{A(S_i)} O(\Delta t^2) \tag{2.125}$$

Figure 2.14 illustrates the procedure of a first-order Runge Kutta method.

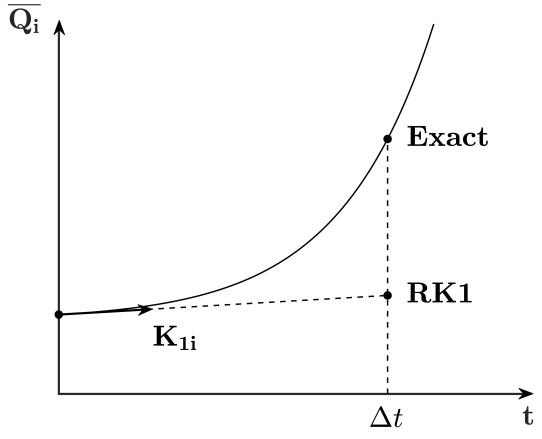

Figure 2.14: Procedure for for a first-order of accuracy Runke-Kutta method.

More sophisticated Runge Kutta methods include intermediate consecutive data points ($\overline{Q}, t_0 + \beta\Delta t$) in which the approximated residual function is evaluated . Each data point is chosen using information from all other previous data points. A linear combination of the residual at each data point is then used for an estimate of the next time-step cell averaged quantities for the target cell ($\overline{Q}(S_i, t_0 + \Delta t)$).

In a generic way, a Runge Kutta method with r number of data points has the structure expressed in equation 2.126.

$$\boxed{\overline{Q}(S_i, t_0) + \Delta t)^{RK^r} = \overline{Q}(S_i, t_0) + \frac{\Delta t}{A(S_i)} \cdot \sum_{u=1}^{\text{ı}} b_u \cdot K_{iu}}$$

$$K_{i1} = R_i(\overline{Q}(t_0)) \Rightarrow k_1 = [\tfrac{K_{11}}{A(S_1)}, \tfrac{K_{21}}{A(S_2)}, \ldots, \tfrac{K_{N1}}{S_N}];$$

$$K_{i2} = R_i(\overline{Q}(t_0) + \beta_2 \cdot \Delta t \cdot (\alpha_{21} \cdot k_1)) \Rightarrow k_2 = [\tfrac{K_{12}}{A(S_1)}, \tfrac{K_{22}}{A(S_2)}, \ldots, \tfrac{K_{N2}}{A(S_N)}]; \tag{2.126}$$

$$K_{i3} = R_i(\overline{Q}(t_0) + \beta_3 \cdot \Delta t \cdot (\alpha_{31} \cdot k_1 + \alpha_{32} \cdot k_2)) \Rightarrow k_3 = [\tfrac{K_{13}}{A(S_1)}, \tfrac{K_{23}}{A(S_2)}, \ldots, \tfrac{K_{N3}}{A(S_N)}];$$

$$\ldots$$

$$K_{\text{ı}} = R_i(\overline{Q}(t_0) + \beta_{\text{ı}} \cdot \Delta t \cdot (\alpha_{\text{ı}1} \cdot k_1 + \alpha_{p2} \cdot k_2 + ... + \alpha_{\text{ı}(\text{ı}-1)} \cdot k_{\text{ı}-1}))$$

The values of the different control parameters α, β and b can be computed in order to reduce the truncation error of the scheme. A second-order Runge-Kutta is derived in order to exemplify the procedure.

Equation 2.127 translates the method mathematically.

$$\boxed{\overline{Q}(S_i, t_0 + \Delta t)^{RK^2} = \overline{Q}(S_i, t_0) + \frac{\Delta t}{A(S_i)} \cdot \sum_{u=1}^{2} b_u \cdot K_{iu}}$$

$$K_{i1} = R_i(\overline{Q}_{t_0})) \Rightarrow k_1 = [\tfrac{K_{11}}{A(S_1)}, \tfrac{K_{21}}{A(S_2)}, ..., \tfrac{K_{N1}}{A(S_N)}]; \tag{2.127}$$

$$K_{i2} = R_i(\overline{Q}_{t_0} + \beta_2 \cdot \Delta t \cdot (\alpha_{21} \cdot k_1))$$

Figure 2.15 illustrates the procedure for the second-order of accuracy Runge Kutta method.

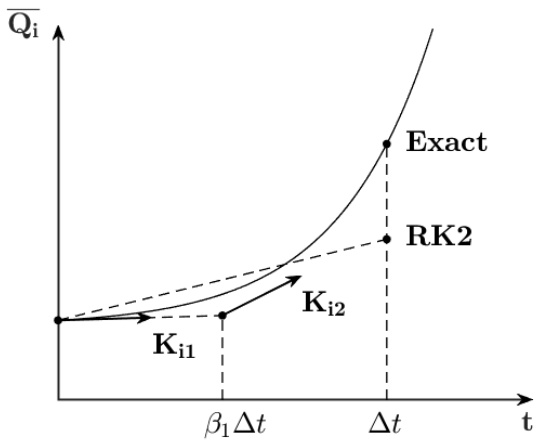

Figure 2.15: Procedure for a second-order of accuracy Runge Kutta method

To acess the dependence of the truncation error on the above mentioned parameters (α and β, etc),

one should compare the exact value of the target cell averaged quantities in the next tiem iteration and the approxiamtion given by the second-order Runge Kutta scheme. For that purpose, we need to state the two values in comparable terms.

One can write K_{i2} as a Taylor expansion over $R_i(\overline{Q}_{t_0})$ and construct equation 2.128.

$$K_{i2} = R_i(\overline{Q}(t_0)) + \beta_2 \cdot \Delta t \cdot \alpha_{21} \cdot k_{1d} \cdot \frac{\partial R_i(\overline{Q}(t_0))}{\partial \overline{Q}(S_d)} + \beta_2 \cdot \Delta t \cdot \frac{\partial R_i(\overline{Q}(t_0))}{\partial t} + \Psi(\Delta t^2) \tag{2.128}$$

The approximated value of the next time-step cell averaged property can be expressed by equation 2.129.

$$\overline{Q}(S_i, t_0 + \Delta t)^{RK2} = \overline{Q}(S_i, t_0) + \frac{\Delta t}{A(S_i)} \cdot (b_{i1} \cdot K_{i1} + b_{i2} \cdot K_{i2}) \tag{2.129}$$

Using the equations in 2.128 and 2.127, one can further expand equation 2.129 into equation 2.130.

$$\begin{aligned} \overline{Q}(S_i, t_0 + \Delta t)^{RK2} &= \overline{Q}(S_i, t_0) + \tfrac{\Delta t}{A(S_i)} \cdot (b_{i1} + b_{i2}) \cdot R_i(\overline{Q}(t_0)) + \dots \\ &\cdots + \Delta t^2 \cdot b_{i2} \cdot \beta_2 \cdot (\alpha_{21} \cdot k_{1d} \cdot \tfrac{\partial R_i(\overline{Q}(t_0))}{\partial \overline{Q}(S_d)} + \cdot \tfrac{\partial R_i(\overline{Q}(t_0))}{\partial t}) + \Omega(\Delta t^3) \end{aligned} \tag{2.130}$$

The exact value of $\overline{Q}(S_i, t_0 + \Delta t)$ can, again, be expressed as a Taylor expansion over $\overline{Q}(S_i, t_0)$, as shown in equation 2.131.

$$\overline{Q}(S_i, t_0 + \Delta t) = \overline{Q}(S_i, t_0) + \Delta t \cdot \frac{d\overline{Q}(S_i, t_0)}{dt} + \frac{1}{2} \cdot \Delta t^2 \cdot \frac{d^2\overline{Q}(S_i, t_0)}{dt^2} + \Theta(\Delta t^3) \tag{2.131}$$

Using equation 2.131, the time derivative in equation 2.131 can be substituted and equation 2.132 obtained.

$$\overline{Q}(S_i, t_0 + \Delta t) = \overline{Q}(S_i, t_0) + \frac{\Delta t}{A(S_i)} \cdot R_i(\overline{Q}(t_0)) + \frac{1}{2} \cdot \frac{\Delta t^2}{A(S_i)} \cdot \frac{dR_i(\overline{Q}(t_0))}{dt} + \Theta(\Delta t^3) \tag{2.132}$$

The full time derivative of the residual in equation 2.132 ($\frac{dR_i(\overline{Q}(t_0)}{dt}$), can be expressed by a summation of partial derivatives. The process is described in equation 2.133.

$$\frac{dR_i}{dt} = \frac{\partial R_i}{\partial t} + \frac{d\overline{Q}(S_d)}{dt} \cdot \frac{\partial R_i}{\partial \overline{Q}(S_d)} = \frac{\partial R_i}{\partial t} + R_d \cdot \frac{\partial R_i}{\partial \overline{Q}(S_d)} = \frac{\partial R_i}{\partial t} + k_{1d} \cdot \frac{\partial R_i}{\partial \overline{Q}(S_d)} \tag{2.133}$$

Equation 2.133 allows equation 2.132 to developed into equation 2.134.

$$\overline{Q}(S_i, t_0 + \Delta t) = \overline{Q}(S_i, t_0) + \frac{\Delta t}{A(S_i)} \cdot R_i(Q_{t_0}) + \frac{1}{2} \cdot \frac{\Delta^2}{A(S_i)} \cdot (\frac{\partial R_i(\overline{Q}(t_0))}{\partial t} + k_{1d} \cdot \frac{\partial R_i(\overline{Q}_{t_0})}{\partial \overline{Q}(S_d)}) + \Theta(\Delta t^3) \tag{2.134}$$

Subtracting the approximated next-iteration target cell averaged quantites, as expressed by equation 2.130, from the exact values, expressed by equation 2.134, the dependence of the truncation error on

the control parameters can be obtained. Equation 2.135 shows the forementioned process.

$$\tau = \overline{Q}(S_i, t_0 + \Delta t) - \overline{Q}(S_i, t_0 + \Delta t)^{RK2} \Rightarrow$$

$$\Rightarrow \tau = \frac{\Delta t}{A(S_i)}(1 - b_{i1} - b_{i2}) \cdot R_i(\overline{Q}_{t_0}) + \frac{\Delta^2}{A(S_i)} \cdot (\frac{1}{2} - b_{i2} \cdot \alpha_{21} \cdot \beta_2) \cdot k_{1d} \cdot \frac{\partial R_i(\overline{Q}_{t_0})}{\partial \overline{Q}(S_d)}) + (\Theta(\Delta t^3) - \Omega(\Delta t^3)) \tag{2.135}$$

The truncation error can therefore be diminuished by eliminating terms in equation 2.135. The system in equation 2.136 can be used.

$$1 - b_1 - b_2 = 0, \qquad \frac{1}{2} - b_2 \cdot \alpha_{21} \cdot \beta_2 = 0 \tag{2.136}$$

One solution, among infinite choices, to the system in equation 2.136 is shown in equation 2.137.

$$\alpha_{21} = \beta_2 = 1 \qquad b_1 = b_2 = \frac{1}{2} \tag{2.137}$$

The Runge Kutta method used troughout this thesis is the third-order Runge Kutta, which can be derived by a similar procedure. The mathematical structure is shown in equation 2.138 and is derived from equation 2.126.

$$\overline{Q}(S_i, t_0 + \Delta t) = \overline{Q}(S_i, t_0) + \frac{\Delta t}{A(S_i)} \cdot \frac{1}{6}(K_{i1} + 4K_{i2} + 1K_{i3}) \tag{2.138}$$

2.7 MUSCL scheme

The Monotonic Upstream-centred Scheme for Conservation Laws is a second-order scheme that uses second-order TVD schemes for computing left and right states in the interface of neighbouring cells.

The second order piece-wise approximation for the field in each cell comes from the use of downwind slopes:

$$u_i^{2^{nd}order}(x) = u_i + \frac{x - x_i}{x_{1+i} - x_i}(u_{i+1} - u_i) \tag{2.139}$$

The first order piece-wise approximation for the field in each cell comes from a first-order cell centred approximation:

$$u_i^{1^{st}order}(x) = u_i \tag{2.140}$$

The monotone scheme with the flux limiter is, therefore:

$$u_i^{2^{nd}orderTVD}(x) = u_i + \Phi(r_i)\frac{x - x_i}{x_{1+i} - x_i}(u_{i+1} - u_i) \tag{2.141}$$

For a regular one dimensional mesh, the left and right piece-wise approximations are used to extrap-

olate a right and left state at each interface in the following manner:

$$u^L_{i+\frac{1}{2}} = u_i + 0.5\Phi(r_i)(u_{i+1} - u_i); \qquad u^R_{i+\frac{1}{2}} = u_{i+1} - 0.5\Phi(r_i)(u_{i+2} - u_{i+1});$$
$$u^L_{i-\frac{1}{2}} = u_{i-1} + 0.5\Phi(r_i)(u_{i-1} - u_{i-2}); \qquad u^R_{i-\frac{1}{2}} = u_i - 0.5\Phi(r_i)(u_{i+1} - u_i); \tag{2.142}$$

The extrapolated states can then be introduced into an approximate Rieman-Solver or used in Riemann-Solver-Free schemes.

2.8 Godunov scheme

The Godunov scheme is a first-order scheme that preserves monotonicity near discontinuities while providing better results near shocks and other flow structures by employing an approximated Riemman solver in each FV cell interface. Using as left and right states the average cell values of the two adjacent cells, a Riemann problem is set and the intercell flux calculated (see [5]):

$$\begin{matrix} Q_L = \overline{Q_i} \\ Q_R = \overline{Q_{i+1}} \end{matrix} \Rightarrow F_{i+\frac{1}{2}} = RS(Q_L, Q_R) \tag{2.143}$$

The original Godunov scheme (see [5]), uses first-order time integration. For motives of equal comparition with other schemes, this thesis will use the term to describe any scheme that uses similar flux calculation to the one in 2.143 along with an arbitrary explicit time integration.

Chapter 3

1D WENO - Implementation and Results

.

3.1 Implementation

In this chapter, a proposed architecture for a WENO scheme is dissected and ultimately used to approximate the residual function of each target cell. The residual for the finite volume formulation of the one-dimensional Euler equations is presented in section 2.3.2.

Computing the next-iteration cell averaged quantities implies knowing the numerical fluxes for each target cell. An approppriate approximation of adequate flow variables is, thereby, required. For an arbitrary order of accuracy (r), a $\wp^{th}$-degree polynomial model can be used to approximate an useful quantity. By consequence, a data set and a regression method are needed. The polynomial model used for the proposed one-dimensional WENO scheme is presented in section 3.1.1. Near discontinuities and shock waves, proper useful regression tends to be unsuccessfull as the resulting polynomials exhibit the Gibbs phenomenon (see section 2.1). For this reason, WENO schemes use directionally biased data sets, called stencils, for each target cell. Stencil creation methodology scheme is presented in section 3.1.2. The polynomial model used for regressing usefull quantities is a convex combination of the models obtained for each stencil. The weight given to each polynomial is dependent on how much they oscillate and thus a much greater importance is given for smoother data sets. This process filters out polynomials that present non-coherent oscillations. The regression method used was the Least-Squares method (section 2.2) with the process for seting up the required matrices being explained in section 3.1.3. The process of computing the full WENO ploynomial for a said quantity is described in section 3.1.4

For the interface between the one-dimensional cells, the necessary quantities receive two different values: one coming from the target cell WENO polynomial and the other from the one for the face neighbouring cell. A Riemann solver is used to solve this discrepancy while maintaining physical coherence.

3.1.1 Polynomial Model

In section 2.2, the polynomial model for use in the Least-Squares Method was defined as a linear combination of basis functions. For the proposed one-dimensional WENO, the Legendre polynomial functions (L_h) were chosen as the basis functions. The first three polynomials are given in equation 3.1.

$$L_0(x) = 1; \quad L_1(x) = x; \quad L_2(x) = \tfrac{1}{2}(3x^2 - 1); \quad L_3(x) = \tfrac{1}{2}(5x^3 - 3x) \tag{3.1}$$

Legendre polynomials are orthogonal over the [-1,1] interval, respecting the relation in equation 3.2.

$$\int_{-1}^{1} L_a(x)L_b(x)dx = \frac{1}{2b+1} \cdot \delta_{ab}; \quad \delta_{ab} = \begin{cases} \delta_{ab} = 0, & a \neq b \\ \delta_{ab} = 1, & a = b \end{cases} \tag{3.2}$$

The property expressed in equation 3.2 dictates that, when multiplied by L_0, every other function has a null integration value in the $[-1, 1]$ interval. As such, the resulting polynomial models can easily benefit from having the same cell averaged value as the target cell. The linear mapping in equation 3.3 can be applied to ensure that, in a new transformed space (ϵ), the target cell extends from $\epsilon = -1$ to $\epsilon = 1$ (standard cell S_{st}). The new coordinate system has its origin in the center of the transformed target cell and the rest of the domain is also passed to this auxiliary coordinate system.

$$\epsilon = J \cdot x - x_c(S_i); \quad J = \frac{1}{2 \cdot d(S_i)} \tag{3.3}$$

Figure 3.1 illustrates the process of mapping a random target cell (S_i) to the standard cell (S_{st}).

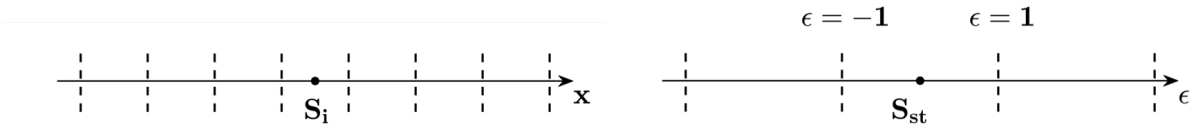

Figure 3.1: Mapping a random target cell S_i (left) to the one dimensional standard one-dimensional cell S_{st} (right).

The usage of an auxiliary system ensures that, for the one-dimensional regular case, the condition number of the matrices used in the Least-Squares Methos is independent on the size of the original mesh. This property is demonstrated in section 3.2.1.

It is also important to note that linear mapping respects equation 3.4 and does not change cell averaged values.

$$\overline{Q} = \frac{1}{d(S_i)} \cdot \int_{S_i} Q(x)\ \partial x = \frac{1}{2} \int_{-1}^{1} Q(\epsilon)\ \partial \epsilon \tag{3.4}$$

The $\wp^{th}$-degree regression model for a generic quantity $U(\epsilon)$ is represented by equation 3.5 with the constants C serving as the control variables of the polynomial model.

$$U(\epsilon) = \overline{U}(S_i) + \sum_{h=1}^{p} C_h L_h(\epsilon) \tag{3.5}$$

3.1.2 Stencil Creation

Section (2.1) states that large jumps in data values may provoke oscillations in approximating polynomials. Unlike ENO schemes, which find the smoothest data closest to the target cell, WENO schemes work with different stencils and a combination of the respective resulting polynomial models. A typical one-dimensional WENO scheme uses three different stencils ($\mathbb{S}_m$) for each target cell: a left biased stencil, a central biased stencil and a right biased stencil. Figure 3.2 illustrates a possible configuration for the three different stencils.

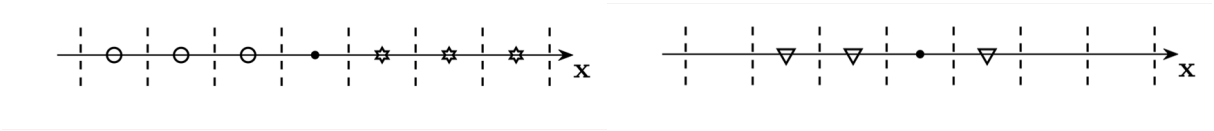

Figure 3.2: Right (stars), left (circles) and central (triangles) biased stencils for a given target cell (black dot).

In the context of this thesis and in accordance with [6], to each stencil was given a number of data equal to the amount of degrees of freedom (C_h) of the polynomial model used plus one. Equation 3.6 expresses this choice.

$$N_{\mathbb{S}} = p + 1 \tag{3.6}$$

The computacional implementation of stencils for the one-dimensional case is fairly simple as Boundary conditions were not used as data. Consequently, the creation of data stencils is independent on the type of data used. This approach was chosen because the gain in accuracy from using boundary conditions as data was found to be diminute when compared to the extra computational strain. For the creation of right-biased stencils, a sucessive use of right neighbours forms the basis of the procedure. The left-biased stencil is created analogously. The process is repeated until each stencil has sufficient data points. The central stencil is created using the left and right stencils and filtering out the data farthest to the target cell.

If either a right or a left stencil fails to be completed, as is the case when close to domain borders, it is simply discarded. This choice to avoid completing the stencil with discontinuous data, hence introducing more oscillations into the final WENO polynomial model.

3.1.3 Least-Squares Problem

The Least-Squares Method and the necessary matrices used are presented and discussed in section 2.2. This section concerns the setup of the necessary matrices for an appropriate Least-Squares Problem.

$\mathbf{D^{LS}}$ Matrix

For this thesis, the polynomial models were made to approximate the cell averaged values contained in a stencil. As such, the $\mathbf{D^{LS}}$ matrix was setup with the cell averaged values of the basis function in each entry of said stencil ($\overline{L}_h(\mathbb{S}_{imn})$). Equation 3.7 shows how each basis function is analitically integrated. The integration is done in the auxilliary coordinate system of the given target cell, where each cell as a lenght of two.

$$\overline{L}_h(\mathbb{S}_{imn}) = \frac{1}{2}\int_{\mathbb{S}_{imn}} L_{ih}(\epsilon)\partial\epsilon \tag{3.7}$$

The D^{LS} matrix can be constructed as described in equation 3.8.

$$D^{LS}_{im} = \begin{bmatrix} \overline{L}_1(\mathbb{S}_{im1}) & \overline{L}_2(\mathbb{S}_{im1}) & ... & \overline{L}_{\wp}(\mathbb{S}_{im1}) \\ \overline{L}_1(\mathbb{S}_{im2}) & \overline{L}_2(\mathbb{S}_{im2}) & ... & \overline{L}_{\wp}(\mathbb{S}_{im2}) \\ \overline{L}_1(\mathbb{S}_{im3}) & \overline{L}_2(\mathbb{S}_{im3}) & ... & \overline{L}_{\wp}(\mathbb{S}_{im3}) \\ & & & \\ ... & ... & ... & ... \\ \overline{L}_1(\mathbb{S}_{imN_S}) & \overline{L}_2(\mathbb{S}_{imN_S}) & ... & \overline{L}_{\wp}(\mathbb{S}_{imN_S}) \end{bmatrix} \tag{3.8}$$

Each D^{LS} matrix may be computed only once at $t = 0$, since it only depends on the independent variables of the set. The same goes for the diagonal weight matrix W^{LS} and, consequently, for the P^{LS} matrix (see section 2.2).

$\mathbf{Y^{LS}}$ Vector

The ultimate goal of a polynomial reconstruction in a WENO scheme is to be used to extrapolate useful quantities. There are two main currents of what data should a WENO scheme regress: some use the original Euler variables (Q) or similar quantities, while others use local characteristic variables (Z) for an approximate decoupled version of the one-dimensional Euler equations (see section 2.3.2). The main advantage in the use of characteristic variables lies in the smoother data that it provides, resulting in less osscilating polynomials. It is, however, a far more computationally strenuous alternative as it requires added processes.

Original Euler Variables

For some original Euler dependent variable (k), the Y^{LS} vector is decribed in equation 3.9.

$$Y^{LS}_{imk} = \begin{bmatrix} \overline{Q}_k(\mathbb{S}_{im1}) - \overline{Q}_k(S_i) & ... & \overline{Q}_k(\mathbb{S}_{imNs}) - \overline{Q}_k(S_i) \end{bmatrix}^T \tag{3.9}$$

Characteristic Euler Variables

For each interface of a cell, a local approximated decoupled formulation of the 1D Euler equations is set, as shown in equation 3.10. The detailed process of decoupling the one-dimensional Euler equations is described in section 2.3.2.

$$\frac{\partial(\widetilde{\mathbb{X}}_R^{-1}Q)}{\partial t} + \widetilde{\Lambda} \cdot \frac{\partial(\widetilde{\mathbb{X}}_R^{-1}Q)}{\partial x} = 0 \tag{3.10}$$

The approximated right eigenvectors matrix ($\widetilde{\mathbb{X}}_R$) and eigenvalue diagonal matrix ($\widetilde{\Lambda}$) used in equation 3.10 are detailed in equation 3.11 and calculated trough the use of Roe averaged values (denoted by a tilde).

$$\widetilde{\mathbb{X}}_R^{-1}{}_{ij} = \begin{bmatrix} 1 & 1 & 1 \\ \tilde{u} - \tilde{a} & \tilde{u} & \tilde{u} + \tilde{a} \\ \tilde{H} - \tilde{u}\tilde{a} & \frac{1}{2}\tilde{u}^2 & \tilde{H} + \tilde{u}\tilde{a} \end{bmatrix} \qquad \tilde{\Lambda}_{ij} = \begin{bmatrix} \tilde{u} - \tilde{a} & 0 & 0 \\ 0 & \tilde{u} & 0 \\ 0 & 0 & \tilde{u} + \tilde{a} \end{bmatrix} \tag{3.11}$$

The Roe averaged quantities of interest are calculated with the use of the target cell averaged quantities ($\overline{Q}_L$) and those of the face neighbouring cell ($\overline{Q}_R$). Equation 3.12 demonstrates how the Roe averaged values are computed.

$$\tilde{u}_{ij} = \frac{\rho_L u_L + \rho_R u_R}{\rho_L + \rho_R} \qquad \tilde{H}_{ij} = \frac{\rho_L H_L + \rho_R H_R}{\rho_L + \rho_R} \qquad \tilde{a}_{ij} = ((\gamma - 1)(\tilde{H} - \tfrac{1}{2}\tilde{u}^2))^{\frac{1}{2}} \tag{3.12}$$

After obtaining the locally coherent decoupled equations, the data contained in a given stencil for the said target cell must be converted to characteristic variables, as shown in equation 3.13.

$$\overline{Z}_(\mathbb{S}_{imn}) = \tilde{R}_{ij}\overline{Q}(\mathbb{S}_{imn}) \tag{3.13}$$

For the use of local characteristic varaibles, the dependent variables vector (Y^{LS}) of the Least-Square Problem consists on the difference between the cell averaged value of a given characteristic variable in a given stencil entry and that of the target cell. Equation 3.14 translates the previous phrase mathematically.

$$Y^{LS}_{imjk} = \begin{bmatrix} \overline{Z}_1(\mathbb{S}_{im1}) - \overline{Z}_1(S_i) & \dots & \overline{Z}_3(\mathbb{S}_{imNs}) - \overline{Z}_3(S_i) \end{bmatrix}^T \tag{3.14}$$

The extra index (j) is present to alert to the fact that, differently to the original WENO, each interface produces different polynomial models. With the use of the original Euler variables, a polynomial model suffices for the entire target cell. Each interface has a different right eigenvalue matrix and different characteristic variables and, thus, needs a specific polynomial model.

Weight Diagonal Matrix ($\mathbf{W^{LS}}$)

A weight function is required for maintaining a well-conditioned Least-Square Problem (see section 2.2) and to emphasize data closer to the target cell. The weight function for the one-dimensional WENO scheme is shown in equation 3.15 and is used in [3] and [4].

$$W_{imnn}^{LS} = (\frac{1}{|\epsilon(\mathbb{S}_{imn})|})^{\wp} \tag{3.15}$$

The $\mathbf{W^{LS}}$ matrix is presented in equation 3.16.

$$W_{im}^{LS} = \begin{bmatrix} (\frac{1}{|\epsilon(\mathbb{S}_{im1})|})^{\wp} & & \\ & \ddots & \\ & & (\frac{1}{|\epsilon(\mathbb{S}_{imN_{\mathbb{S}}})|})^{\wp} \end{bmatrix} \tag{3.16}$$

3.1.4 WENO Polynomial Model

For a said target cell, polynomial models for each stencil and quantity are obtained and, as some stencils may intersect discontinuites and/or shocks, the Gibbs phenomenon may occur in the respective approximated polynomial models. The WENO methodology uses a convex combination of polynomial models obtained for each stencil in which a larger consideration is given to less oscillating polynomials. Thus, filtering-out unwanted polynomial models.

In equation 3.17, the WENO full polynomial model for a characteristic local variable (M^{CW}) is expressed as convex combination of polynomial models for said face (j), stencil (m) and local characteristic variable (k).

$$M_{ijmk}^{CW} = W_{ij1k}^{CW} \cdot M_{ij2k}^{C} + W_{ij2k}^{CW} \cdot M_{ij2k}^{C} + ... \tag{3.17}$$

If the quantities approximated by the polynomials models are the original Euler variables, index j may be omitted as the WENO polynomial models is valid for the entire target cell and not just for a determined cell face.

The weight function, presented in equation 3.18, gives higher importance to polynomial models that present less oscillations within the target cell.

$$W^{CW} = \frac{\Gamma}{\sum_{e=1}^{3} \Gamma_e}; \quad \Gamma = \frac{w}{(\iota + SI)^r} \tag{3.18}$$

A linear weight (w) is used as an additional control parameter. Central stencils are usually given a higher value ($w = 10^3$) as they are more accurate in smoother areas of the domain (see [4]) and the directional stencils given a lower value ($w = 1$). The smoothness indicator of the polynomial model (SI), given in equation 3.19, serves as metric for how much a certain polynomial model oscillates in the confines of the target cell.

$$SI = \sum_{i=1}^{p} \int_{S_{st}} (\frac{d^i}{dx_t{}^i} \cdot M(\epsilon, \eta))^2 \, \partial S_{st} \tag{3.19}$$

The value of ι in equation 3.18 is usually very small and is used to avoid divison by zero.

3.1.5 Flux Calculation

The computation of numerical fluxes directly implies knowing the dependent variables or equivalent on the interfaces. Equation 3.20 shows how these can be extrapolated using the polynomial models that approximated the original Euler variables (M).

$$Q(\epsilon(S_{ij})) \approx \Big[M_{ij1}(\epsilon(S_{ij})) \qquad ... \qquad M_{ij3}(\epsilon(S_{ij}))\Big]^T \tag{3.20}$$

As show in equation 3.21, the use of WENO polynomials that regressed local characteristic variables (M^C) is also a viable option.

$$Z(\epsilon(S_{ij})) \approx \Big[M^{CW}_{ij1}(\epsilon(S_{ij})) \qquad ... \qquad M^{CW}_{ij3}(\epsilon(S_{ij}))\Big] \tag{3.21}$$

The use of said models (equation 3.21) implies that, on the points of interest, the characteristic variables must be converted to original Euler variables. Equation 3.22 explains the process of convertion.

$$Q(\epsilon(s), \eta(s)) \approx \widetilde{R}_{ij} \cdot Z((\epsilon(s), \eta(s))) \tag{3.22}$$

For the same face, two different sets for the same quantities are calculated. One data set is extrapolated from polynomials associated with the target cell (Q_L) and the other from those of the respective face neighbouring cell (Q_R). Equation 3.23 demonstrates how a Riemann problem can be setup and an approximate solver used to resolve this artificial discontinuity while maintaing physical coherency..

$$F(\epsilon(S_{i\pm\frac{1}{2}})) \approx RS(Q_L, Q_R) \tag{3.23}$$

On section 3.1.6, benchmark testing will be used to choose from amongst three popular approximated Riemann solvers (Riemann Solver of Roe, HLL and HLLC).

The residual for each cell on the current time-step is then calculated, as given by equation 3.24.

$$R_i = (F(x\big(S_{i+\frac{1}{2}}\big)) - F(x\big(S_{i-\frac{1}{2}}\big)) = (F(\epsilon\big(S_{i+\frac{1}{2}}\big)) - F(\epsilon\big(S_{i-\frac{1}{2}}\big)) \tag{3.24}$$

3.1.6 Comparing Approximated Riemann Solvers

In the intent of choosing an appropriate riemann solver, a comparison in terms of acuracy and computational efficiency between the three solvers presented in section 2.5 was made. The Godunov first-order scheme in conjunction with the respective Riemann solvers was used to solve the SST problem at $t = 0.2s$. The L_2 norm, defined in equation 3.25, of the errors in cell averaged density was used for comparing the three.

$$||e||_2 = \sqrt{\frac{1}{N} \cdot ||e||} = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} |e_i - e_i^{exact}|} \tag{3.25}$$

Figure 3.3 shows the values of the L_2 norm of the errors in cell averaged density at $t = 0.2s$ for four levels of mesh refinement and respective solver run-time.

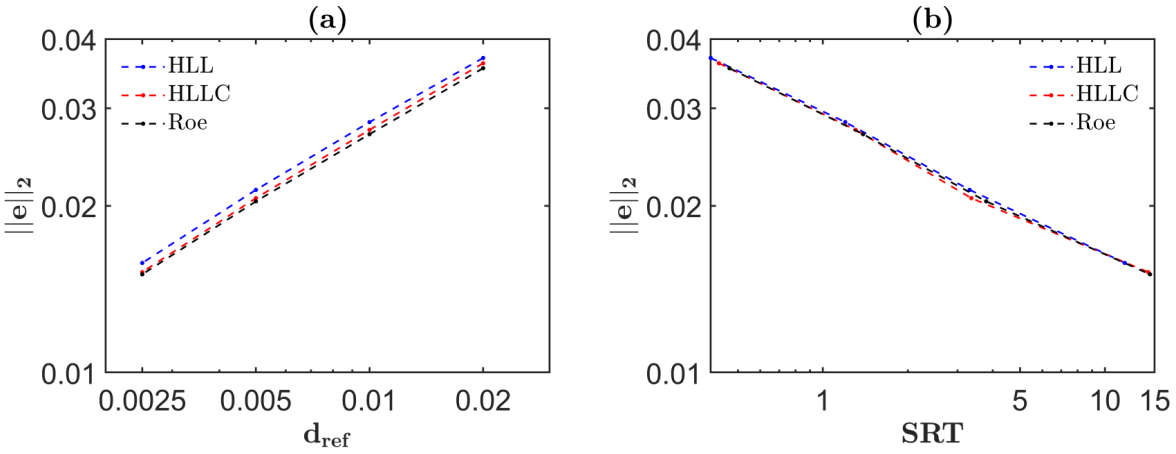

Figure 3.3: L_2 norm of the errors in cell average density at $t = 0.2s$ as a function of (a) average cell size (d_{ref}) and (b) solver run-time (SRT).

The Solver of Roe and HLLC have similar performance in terms of accuracy and computational efficiency .The HLL solver has comparable computational efficiency to the other two, but accuracy-wise it suffers from not being able to solve the left and right states of the contact discontinuity. Figure 3.4 illustrates the slighlty more diffusive behaviour of the HLL riemann solver in the vicinity of the contact discontinuity. All three Riemann solvers have similar performance in the other parts of the domain, including the shock wave.

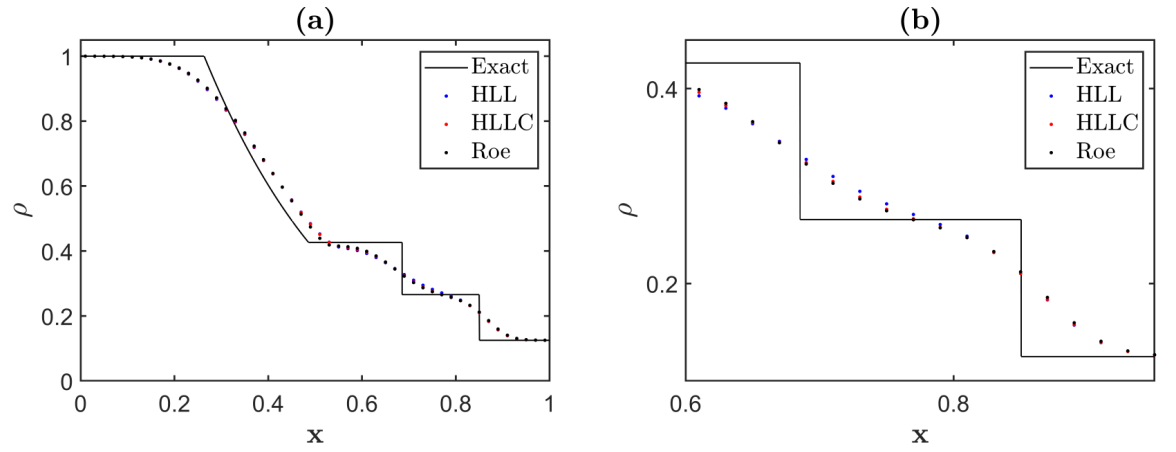

Figure 3.4: L_2 norm of the errors in cell average density at $t = 0.2s$ as a function of (a) average cell size (d_{ref}) and (b) solver run-time (SRT)

Given this data, the decision was made to follow [4] and choose the HLLC Riemman solver. The choice of wavespeeds is pressure-based, presented in section 2.5.3, which is described in [5] to be sufficiently robust.

3.1.7 Flux limiting for the 1D-WENO scheme

For some test cases, the proximity of discontinuities and shocks did not allow the WENO methodology to properly work, as all stencils, for a given target cell, intersected these flow structures. The use of local characteristic variables mitigates the non-solution-coherent oscillations that appear, but do not completely get rid of them. As such, a flux limiting strategy similar to [7] was implemented for the one-dimensional WENO schemes. The quantities extrapolated on the face from the WENO models are checked for positivity (pressure, density and energy) and for monotonicity (the value on the face must be between the left and right cell values). If the previous conditions are not fulfilled, all the quantities on the face (Q_L and Q_R) are then extrapolated using a MUSCL scheme with a min-mod flux limiter ($\Phi_{min-mod}$). The structure of the full flux scheme is given by equation 3.26, in which a function (t_w) is used to toggle between WENO and MUSCL procedures.

$$Q = (1 - t_w) \cdot Q^{MUSCL} + t_w \cdot Q^{WENO} \tag{3.26}$$

3.1.8 Time integration

The time integration method used is the third-order Runge Kutta method described in section 2.6 . The next time iteration averaged values ($Q(S_i, t_0+)$) were calculated as expressed by 3.27

$$\overline{Q}(S_i, t_0+) = \overline{Q}(S_i, t_0) + \frac{\Delta t}{d(S_i)} \cdot \frac{1}{6}(K_{i1} + 4K_{i2} + 1K_{i3}) \tag{3.27}$$

The terms K_{i1}, K_{i2} and K_{i3} in equation 3.27 are defined in equation 3.28:

$$K_{i1} = R_i(\overline{Q}_{t_0}) \qquad K_{i2} = R_i(\overline{Q}_{t_0} + 0.5 \cdot k_1 \cdot \Delta t) \qquad K_{i3} = R_i(\overline{Q}_{t_0} + 0.5 \cdot k_2 \cdot \Delta t) \tag{3.28}$$

In accordance with [4], the time step (Δt), expressed in equation 3.29, was calculated in such a way that preserved the stability of the temporal scheme.

$$\Delta t = \mu \frac{\min(d(S_i))}{2 \cdot \max|\lambda|} \tag{3.29}$$

The CFL number was given the value of 0.4. In the one dimensional case, the term $d(S_i)$ is computed as the real lenght of a one-dimensional cell. The term $\max|\lambda|$, in equation 3.29, is the fastest characteristic speed in all the domain, at that particular time-iteration.

3.2 Test Cases

3.2.1 Diffusion Equation and Gaussian

In order to prove that the proposed WENO schemes produced arbitrary high order of accuracy when in presence of smooth quantities, a test case present in [7] was implemented. The linear diffusion equation

with unitary diffusivity and a Gaussian source term, as shown in equation 3.30, was used as the ruling differential equation. This test case also served to study the conditioning of the matrices used.

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial^2 t} + S^{ce}(x) \qquad S^{ce}(x) = \frac{\partial^2}{\partial^2}(5 \cdot e^{-x^2}) \tag{3.30}$$

The steady-state analytical solution of equation 3.30 is presented in equation 3.31.

$$U(x, \infty) = 5 \cdot e^{-x^2} \tag{3.31}$$

The cell averaged steady state solution was injected into a one-dimensional domain discretized by N one-dimensional cells. Appropriate code was developed and left to iterate untill it converged. The achieved steady-state solution was then compared with the steady-state analytical solution using the L_2 error norm described in 3.1.6. The time integration method was a first-order Runge-Kutta and the time step was set in such a way that time integration played a diminute part in overall error ($\Delta t = 1.25 \cdot 10^{-04} s$, 8000 iterations). Following the directives in [7], a null flux was imposed on the boundaries .

To ensure that the use of the Least-Squares Method in the numerical scheme gave the proposed order of accuracy, the derivative of the poynomial models resultant from only the central stencils were used to calculate the intercell fluxes. The error metrics decay with mesh refinement is presented in figure 3.5 (subfigure (a)). The WENO combination of the three different polynomial models can be subjected to the same procedure. The order of accuracy for the full WENO one-dimensional polynomials was evaluated. The results are show in figure 3.5 (subfigure (b)).

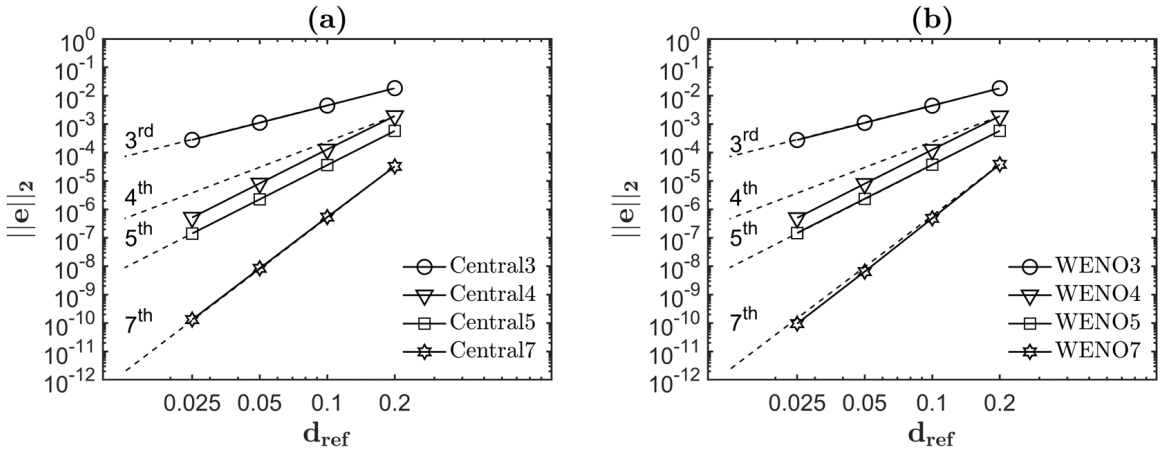

Figure 3.5: L_2 norm of the errors in quantity U as a function of mesh reference size (d_{ref}) for (a) Central and (b) WENO schemes.

From the graphs in image 3.5, one can come to the conclusion that the WENO scheme behaves very similarly to the Central scheme, suggesting that, in regions of smooth quantities, the developed WENO scheme correctly favours the central polynomial models, as advertised in 3.1.3. The WENO and Central schemes retained the desired order of accuracy and no mentionable penalty could be observed

for this test case. All schemes followed or exceeded the proposed order of accuracy. Strangely, the fourth-order Central and WENO schemes (third-order for the first derivative) gave a better performance than the expected error decay. The correct functioning of the code developed for application of the Least-Squares Method was, therefore, corroborated.

To evaluate if the P matrices used were properly conditioned, an average condition number (κ) was computed for each scheme and mesh used throughout this section. Subfigure (a) in 3.6 showns the average condition number of the matrices used as a function of average cell size (d_{ref}).

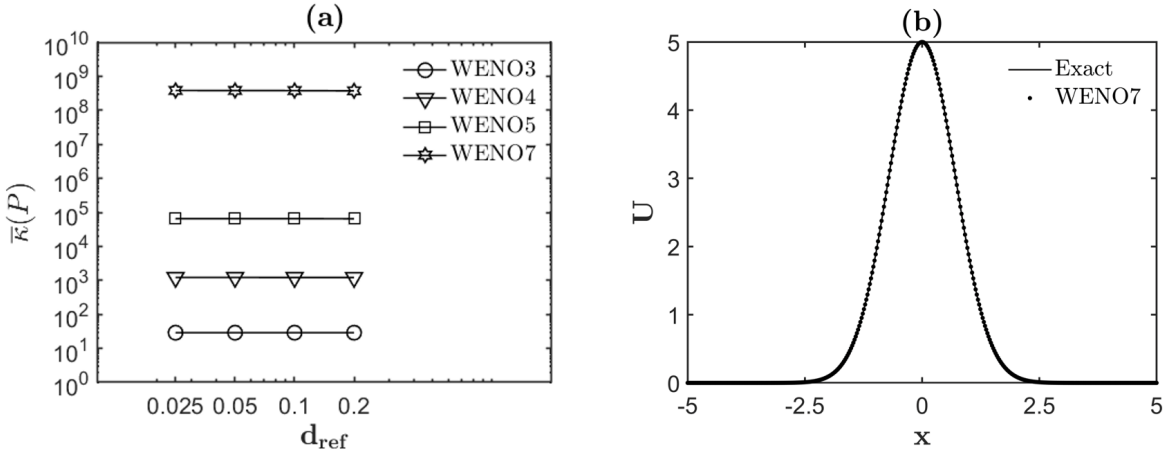

Figure 3.6: (a) - Condition number (κ) of the P matrices versus mesh reference size (d_{ref}). (b) - Cell averaged values of U obtained from the seventh-order WENO scheme at $t = 1s$.

From figure 3.6, the prediction made in section 3.1.1 was confirmed. It stated that, with the use of transformed space, the condition number of the P matrices would be independent on the mesh used. Another important conclusion is that the average condition number increases exponentially with the desired order of accuracy (r) of the WENO scheme. For higher order interpolations some added measures might need to be taken to prevent degradation of the results. The P matrices were found to be sufficiently conditioned for the error range of the test cases carried out in this Thesis.

The results given by the seventh-order WENO scheme (WENO7) are shown in subfigure (b) in 3.6 to serve as an illustration of the several results obtained.

3.2.2 Burguers Equation and Shocks

One of the important functions of the proposed one-dimensional WENO scheme is proper attribution of weights to the obtained polynomial models. As such, a test case for the Burguers equation was taken from [7]. The governing differential equation is expressed in equation 3.32) with use of a variable U.

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial t} = 0 \tag{3.32}$$

The initial point-wise values of property U are stated in equation 3.33 and the initial cell averaged values of U were computed using a mid-point rule integration with a thousand points in each cell.

$$U(x,0) = \begin{cases} max[0,(x-2.5)(-x+3.5)] & \text{if } x-2.5 \geq 0 \\ max[0,(x-2.5)(x-1.5)] & \text{if } x-2.5 < 0 \end{cases} \tag{3.33}$$

The boundaries received a null Neumann boundary condition. The time integration method used was an explicit Euler (first order Runge Kutta, see section 2.6) with an appropriate time-step ($\Delta t = 0.0025s$, 8000 iterations) that ensured that time discretization errors were of a much lower order of magnitude than those of spatial discretization. The reference solution used as the exact one is resultant from a first-order upwind scheme with a domain discretized by ten thousand cells.

For this particular test case, the main characteristic that should be evaluated is the absence or presence of oscillations near discontinuities. These confirm or discredit the appropriate choice of smooth polynomials in the WENO scheme. Four WENO schemes of different target order of accuracy (third, fourth, fifth and seventh) were used to compute the solution at $t = 0.25s$. The results obtained for each with a discretized domain of four hundred cells are presented in figures 3.7 and 3.8. The error metrics for this test case are not shown as the smooth region of the initial conditions is only a quadratic function.

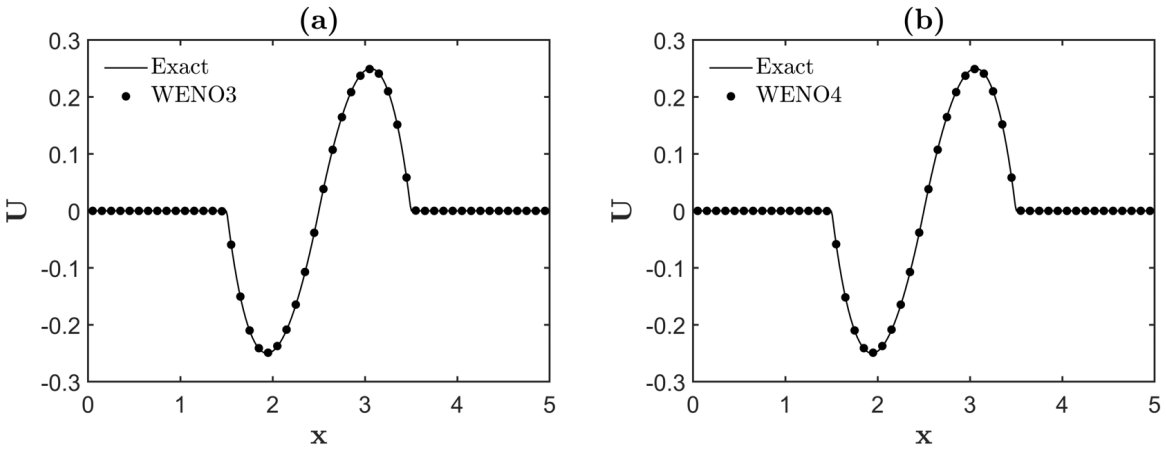

Figure 3.7: U cell averaged values as a function of x coordinate for (a) third-order WENO and (b) fourth-order WENO schemes.

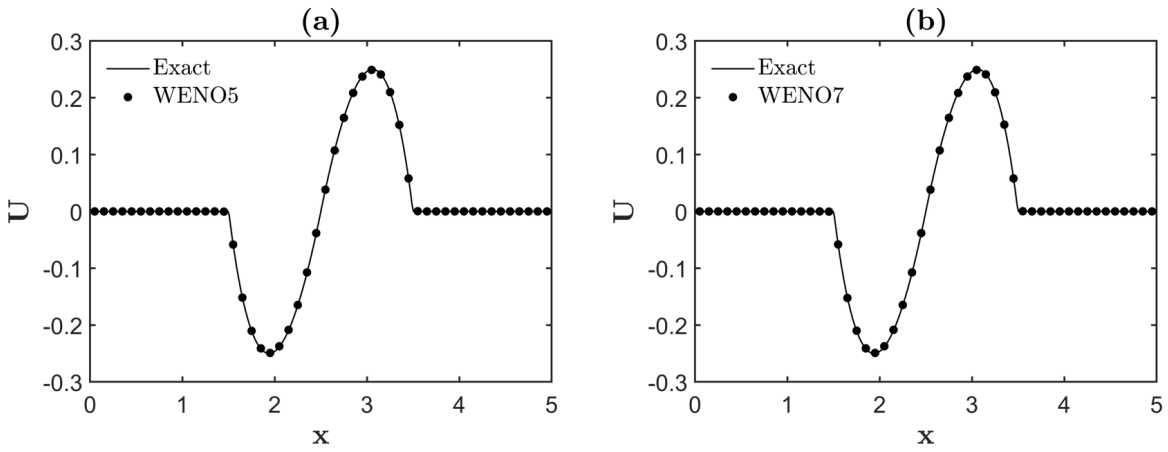

Figure 3.8: U cell averaged values as a function of x coordinate for (a) fifth-order WENO and (b) seventh-order WENO schemes.

All four WENO schemes showed no spurious oscillations and followed the exact solution very well. From the results, the code responsible for the full WENO polynomials was validated as it succesfully filtered-out polynomial models that exhibited the Gibbs phenomenon.

3.2.3 Sod's Shock Tube

As the developed WENO scheme needs to be sufficiently robust for handling shock waves, the Sod's Shock Tube test case, presented in section 2.4.2 was implemented. Very demanding in nature as there are always various descontinuities in proximity, it gives a very good screen. The low values used present another dificulty as any minor wiggles in the polynomial models may cause non-positivity of pressure, energy and density. These, in turn, can cause a failure in the HLLC Rieman solver as these quantities appear square-rooted (see section 2.5.2). The results and conclusions presented in this section impacted the implementation of the two-dimensional version of the WENO scheme. The accuracy and computacional efficiency of the schemes were also compared to wide-spread schemes used for dealing with shock waves and discontinuities, such as the one-dimensional Godunov and MUSCL schemes. The ruling differential equations are the Euler set presented in 2.3.2 with the initial ($t = 0s$) point-wise values of the Euler variables defined in equation 3.34.

$$Q(x,y) = \Big[\rho, \quad \rho u, \quad E\Big] = \begin{cases} \Big[1, \quad 0, \quad 2.5\Big], & x \leq 0.5 \\ \\ \Big[0.1, \quad 0, \quad 0.125\Big], & x > 0.5 \end{cases} \tag{3.34}$$

The initial cell-averaged values of the Euler variables were set by directly prescribing the respective states to the cells at either the left or right side of the initial discontinuity (diaphragama). The time-step was set for each level of mesh refinement by achieving temporal convergence with the Godunov scheme. The temporal scheme used was the third-order Runge-Kutta time scheme described in 2.6. The left and

right boundaries were given Neumann boundary conditions. The two methodologies for approximated quantities, described in 3.1.3, were implemented.

WENO - Original Euler Variables

Results for the component-wise WENO scheme were not computed as the aproximated polynomials contained too much oscillations. The close proximity of large jumps in Euler variables did not allow for proper functioning of the scheme. Positivity was not preserved for density, energy and pressure face values and, as a result, the Riemann Solver could not compute propper numerical fluxes. Altough several attempts were made, including given a higher weight to smoother interpolations, the Gibbs phenomenon was not sufficiently mitigated and, eventually, there would be one polynomial that failed to produce adequate values and ruin the solution.

WENO - Characteristic Euler Variables

The WENO scheme in conjunction with the use of local characteristic variables was used to solve the SOD test case at $t = 0.2s$ and evaluate if the proposed WENO scheme could maintain positivity and monotonicity and provide advantageous levels of accuracy. The third, fourth, fifth and seventh-order schemes were implemented and compared with a first-order Godunov and a MUSCL scheme with a min-mod slope limiter. Both were taken from [5]. The min-mod slope/flux limiter is described in equation 3.35).

$$\Phi_{min-mod}(r_i) = max[0, min(1, r_i)]; \quad \lim_{r_i \to \infty} \Phi_{min-mod}(r_i) = 1 \tag{3.35}$$

As a means of comparing the accuracy and computational efficiency of the proposed WENO schemes to the MUSCL and Godunov schemes, the L_2 norm (see section 3.1.6) for the error in cell averaged density at $t = 0.2s$ was calculated for four different levels of mesh refinement. The exact solution for density at $t = 0.2s$ was taken from [5] and the cell averaged density values computed using the midpoint rule with ten thousand sample points in each cell. As a way to evalue both accuracy and computational efficiency, figure 3.9 presents the L_2 norm for the error in cell averaged density at $t = 0.2s$ as a function of average cell size (d_{ref}) and solver run-time (SRT).

From subfigure (a) in 3.9 several comments can be drawn. The first ones may be that all the WENO schemes provided remarkably better accuracy than the more popular schemes and that the MUSCL scheme provided better accuracy than the Godunov scheme. For the three lowest levels of refinement, higher order translated into better accuracy. However, the same statement is not valid for the highest level of refinement, in which the seventh-order WENO produced the worst accuracy of the WENO schemes. The reason for this event becomes clear when looking at the density plots in figures 3.10,3.11 and 3.12. Computed for the highest level of refinement, these figures appear to indicate that the oscillatory behaviour of the WENO schemes drastically increases with the order of accuracy. Higher order polynomials require added sufficient physical separation between shocks and discontinuities in order to work properly. If all stencils encounter discontinuities, WENO methodology does not properly ensure

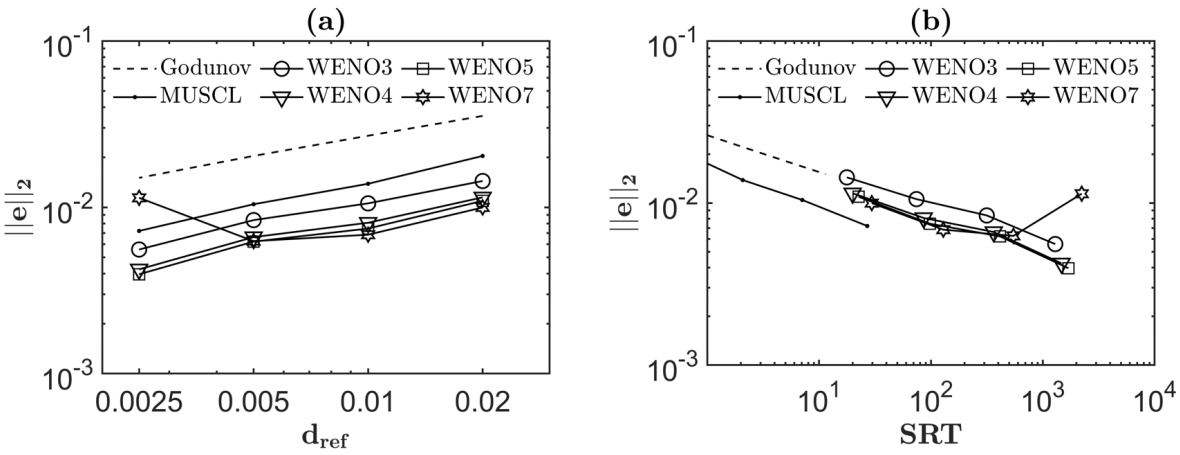

Figure 3.9: L_2 norm for the error in cell averaged density at $t = 0.2s$ as a function of (a) average cell size (d_{ref}) and (b) solver run-time (SRT) (without flux limiting).

monotonicity. The oscillatory behaviour of WENO schemes can also increase with mesh refinement, as discontinuities and shocks produce sharper edges in cell averaged values. Figure 3.16 illustrates this previous remark.

In terms of computational performance, one can understand why second-order TVD schemes are more widely used than high-order schemes for supersonic inviscid flow as, for the same magnitude of error, the MUSCL scheme takes far less time to provide a solution. Between WENO schemes, the fourth-order CWENO gave a better computational efficiency than the other three. All schemes provided better overall performance than the Godunov scheme.

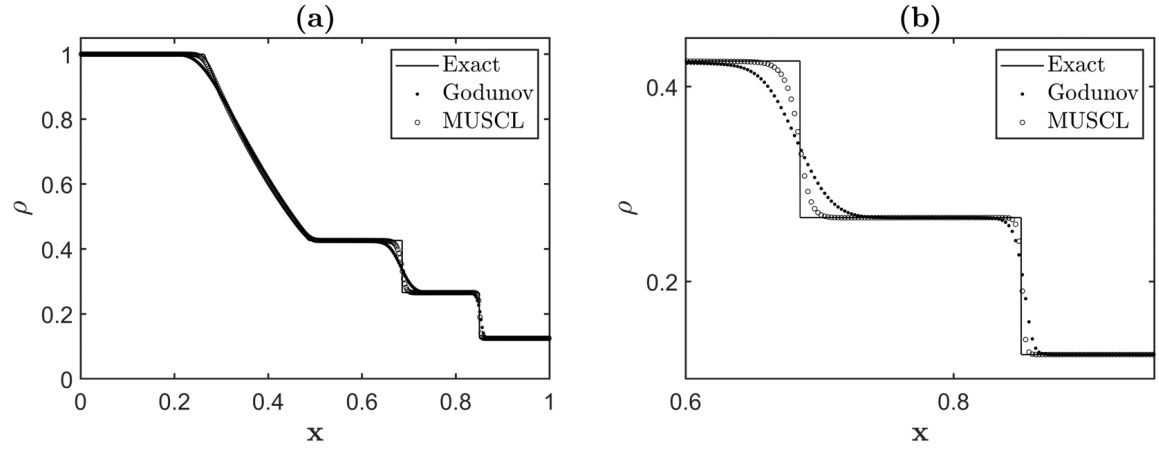

Figure 3.10: Density as a function of the x-coordinate for the Godunov and MUSCL schemes at $t = 0.2s$ with (b) detail over the contact discontinuity and shock wave.

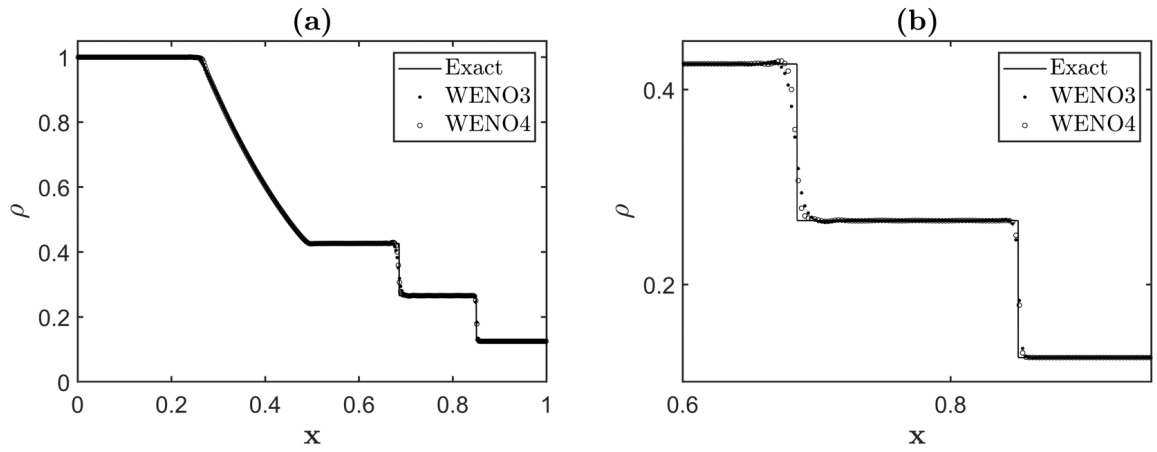

Figure 3.11: Density as a function of the x-coordinate for the third and fourth-order WENO schemes at $t = 0.2s$ with (b) detail over the contact discontinuity and shock wave (without flux limiting).

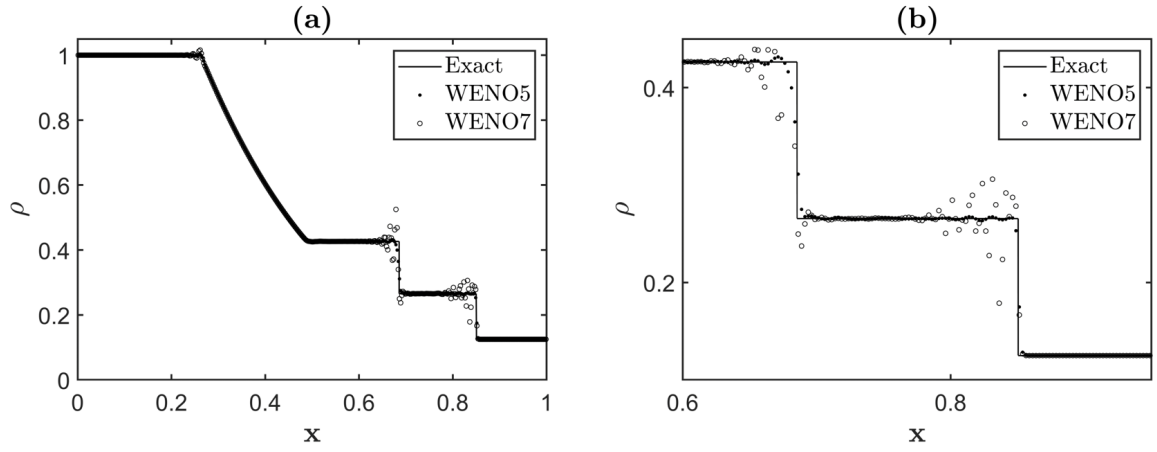

Figure 3.12: Density as a function of the x-coordinate for the fifth and seventh-order WENO schemes at $t = 0.2s$ with (b) detail over the contact discontinuity and shock wave (without flux limiting).

The two previous figures suggest that the third-order WENO scheme had the least amount of non-solution-coherent oscillations but, as all other WENO schemes, could not preserve the monotononicity of the exact solution near the shock and contact discontinuity. In spite of the oscillatory behaviour, the third, fourth and fifth-order WENO schemes managed to provide sharper jumps in cell averaged quantities near the shock and contact discontinuity. Predictably, they also offered substancially better discretization of the expansion wave. The seventh-order WENO provided a oscillation ridden solution near the the shock and contact discontinuity. The MUSCL scheme provided satisfatory performance near the contact discontinuity and shock, as the surrounding area is of low complexity, and maintained the monotonocity of the exact solution throughout the domain. It should be noted that this test case favoured the MUSCL scheme from the start as it has multiple close discontinuities in quantities and an expansion fan that is almost a straight line with respect to $\rho(x)$. The Godunov scheme provided the worst overall results.

In order to satisfy the monotonicity preserving condition, the proposed WENO schemes were given appropriate flux limiting. The flux limiting methodology for the one-dimensional WENO schemes is

described in 3.1.7. For the WENO schemes with flux limiting, the L_2 norm for the error in cell averaged density at $t = 0.2s$ as a function of average cell size (d_{ref}) and as a function of solver run-time (SRT) is shown in figure 3.13.

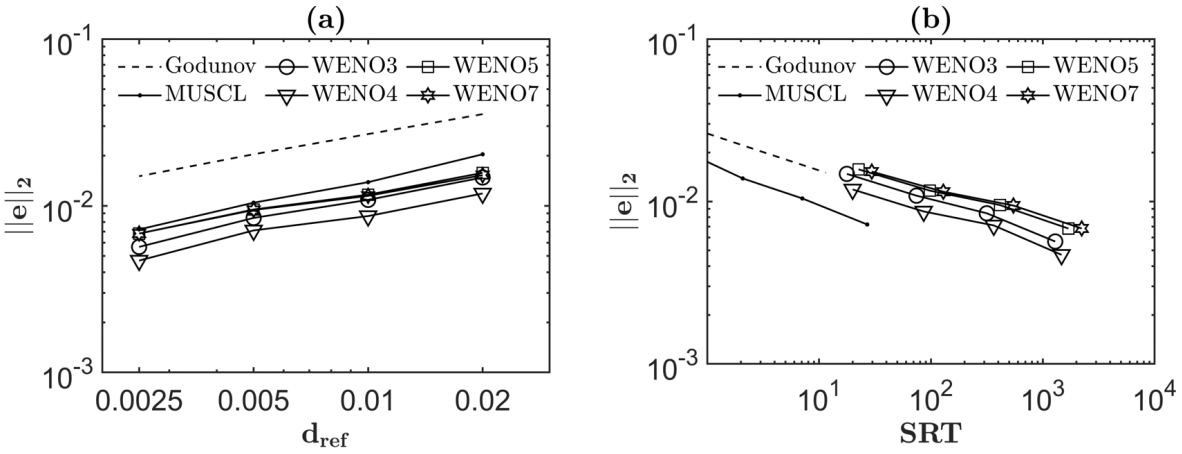

Figure 3.13: L_2 norm for the error in cell averaged density at $t = 0.2s$ as a function of (a) average cell size (d_{ref}) and (b) solver run-time (SRT) (with flux limiting).

For the results in figure 3.13, it can be stated that the overall accuracy of the WENO schemes decreased with use of flux limiting, indicating that forcing monotonicity came at a trade-off in acuracy. The fifth and seventh-order WENO schemes, which showed greater non-solution-coherent oscillatory behaviour, suffered the most, with larger portions of the domain reverting to second and first-order accuracy. All WENO schemes provided better accuracy than the MUSCL and Godunov schemes. The fourth-order WENO scheme provided the best overall accuracy, followed by the third-order WENO scheme.

As accuracy for the WENO schemes suffered from the flux limiting strategy, and the time to compute the necessary polynomials, either they were used or not, was mostly the same, the computational efficiency of the WENO schemes also decreased. The Godunov scheme surpassed all but the fourth-order WENO scheme. The MUSCL scheme provided the best computational efficiency.

The non-oscillattory behaviour of the computed cell averaged density at $t = 0.2s$ for the WENO schemes can be seen in figures 3.14 and 3.17.

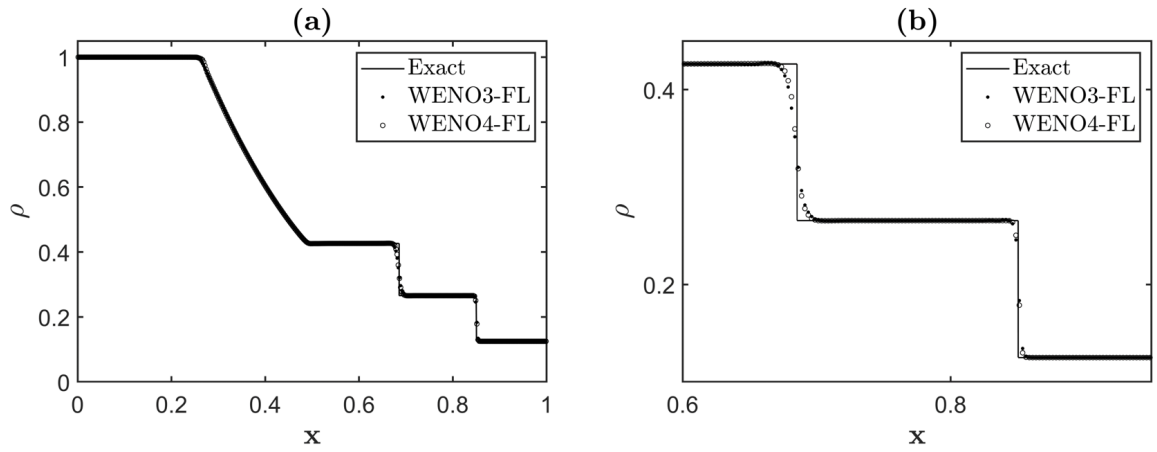

Figure 3.14: Density as a function of the x-coordinate for the third and fourth-order WENO schemes at $t = 0.2s$ with (b) detail over the contact discontinuity and shock wave (with flux limiting).

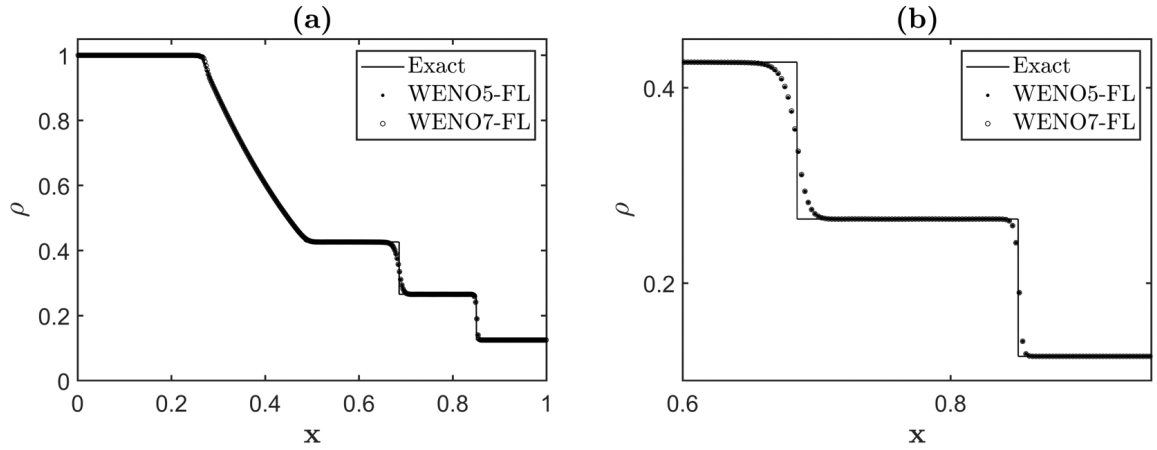

Figure 3.15: Density as a function of the x-coordinate for the fifth and seventh-order WENO schemes at $t = 0.2s$ with (b) detail over the contact discontinuity and shock wave (with flux limiting).

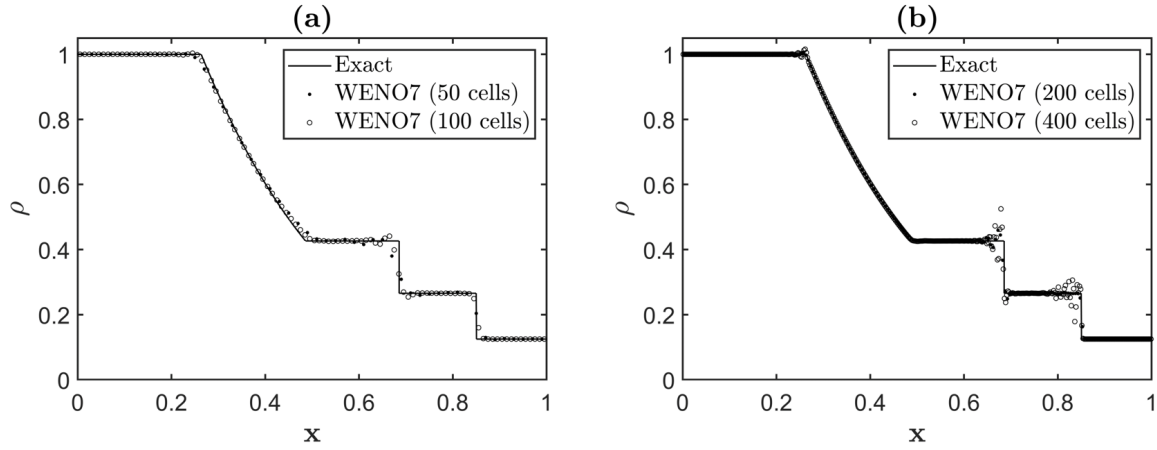

Figure 3.16: Results in density at $t = 0.2s$ obtained from the sevent-order WENO scheme for the four levels of mesh refinement.

The reason why characteristic quantities provide smoother polynomial regression is evident when

comparing values of Z_1 and Z_2 (Z_3 is equal to Z_1 across the domain) to density, momentum and energy (original Euler quantities) values across the domain. Figure 3.17 illustrates this comparison.

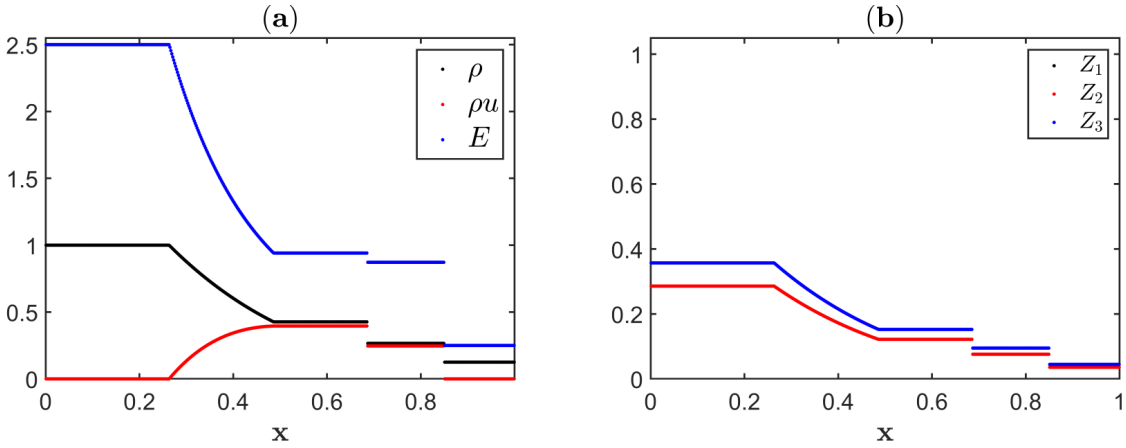

Figure 3.17: (a) Original and (b) local characteristic variables for the SOD test case at $t = 0.2s$.

In every wave structure, the jumps in local characteristic quantities are much lower than those of the original Euler varaiables. Consequently, approximating these quantities results in smoother polynomials.

Chapter 4

2D WENO - Implementation and Results

4.1 Implementation

Throughout this chapter, a proposed version of a WENO scheme for a two-dimensional finite-volume formulation of the Euler equations is presented and its inner-workings explained. The scheme is used to approximate the residual of the finite-volume formulation, present in section 2.3.3, of a given target cell (S_i) on a domain discretized by N number of cells. In regions of the discretized domain where data is smooth, the scheme provides high-order of accuracy while maintaining monotonicity and positivity around shocks and discontinuities.

The polynomial model used is presented in section 4.1.1 of this chapter. The regression method used is the Least-Squares Method which is explained with detail in section 2.2 .

For a given target cell, several polynomials of equal degree, each one resultant from a different collection of data (stencils), are computed. The procedure of stencil creation is explained in section 4.1.2 of this chapter.

The setup of the Least-Squares Problem in the presented WENO scheme is discussed in section 4.1.3 of this chapter.

A final polynomial for a given target cell (WENO polynomial) is calculated using a combination of the polynomials resultant from each stencil. As the Gibbs phenomenon is to be mitigated in this final polynomial, more emphasis is given to polynomials that present less oscillations. This procedure forms the basis of the proposed non-oscillatory behaviour of WENO schemes as the scheme filters out computed polynomials that do not present monotonic cell face values. The procedure is detailed in section 4.1.4

The resulting polynomials for certain types of data (be it pressure, density, etc.) are used to extrapolate adequate quantities on the cells faces. A Riemann solver is then used to resolve the numerical inter-cell fluxes given the two intersecting extrapolated values: one from the polynomial models centred on the target cell, the others from the polynomial models centred on the neighbouring cell (see section 4.1.5)

4.1.1 Polynomial Model

The polynomial model is introduced in section 2.2 as a linear combination of basis functions. For the proposed two-dimensional WENO methodology, the basis function were chosen to be modified orthogonal Taylor functions of up to $\wp^{th}$ degree. The standard form for the chosen basis functions is shown in equation 4.1.

$$a, b \in \mathbb{N}; \quad a + b \leq \wp : \qquad B(x, y) = x^a y^b + Cte \tag{4.1}$$

As previously (see section 3.1.1), the basis functions are required to have a null integration value in the target cell (S_i). This can be achieved by defining the constant Cte by the process in equation 4.2.

$$Cte = -\int_{S_i} x^a y^b \, \partial S_i \tag{4.2}$$

If left alone, the previous requirement would imply different basis functions , as happens in [4], for every target cell and subsequent integration of every binomial in each cell. A strenuous route that dwindles applicability of an already demanding scheme. Having in mind that this thesis only aims to apply WENO schemes to regular and triangular meshes, a computationally friendlier way of ensuring that the integration value is null is through the usage of auxiliary coordinate systems (ϵ, η). Each cell is tranformed into a standard cell (S_{st}), and the rest of the mesh is linearly mapped. In their respective auxiliary coordinate system, all cells have the same basis functions. This procedure can also be useful for minimizing scalling effects, as stated in [4]. The complete form for the basis functions is described in equation 4.3.

$$B(\epsilon, \eta) = \epsilon^n \eta^m - \int_{S_{st}} \epsilon^a \eta^b \, \partial S_{st} \tag{4.3}$$

By making the auxiliary coordinate system origin coincide with the centroid of the standard cell, the polynomial model is also cell-centred. Analytic integration of the binomials in the standard cell is straight-forward and does not require quadrature methods. The complete polynomial model for an arbitrary quantity U in a auxiliary coordinate system is expressed in equation 2.107.

$$M(\epsilon, \eta) = \overline{U}(S_i) + \sum_{h=1}^{\frac{1}{2}(\wp+1)(\wp+2)-1} C_h B_h(\epsilon, \eta) \tag{4.4}$$

Once again, linear mapping ensures that a random cell averaged quantity ($\overline{U}$) remains the same for both the original and auxiliary coordinate systems and, thus, respects the property in equation 4.5.

$$\overline{U}(S_i) = \frac{1}{A(S_i)} \int_{S_i} U(x, y, t) \, \partial S_i = \frac{1}{A(S_{st})} \int_{S_{st}} U(\epsilon, \eta, t) \, \partial S_{st} \tag{4.5}$$

Auxiliary Coord. Transformation for Triangular Elements

Each triangular target cell gets subjected to the linear transformation in equation 4.6 and mapped into an equilateral triangle with unitary side.

$$\begin{bmatrix} x \\ y \end{bmatrix} = J \begin{bmatrix} \epsilon \\ \eta \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix}, J = \begin{bmatrix} 2x_1 - 3x_c + x_3 & \sqrt{3}(x_3 - x_c) \\ 2y_1 - 3y_c + y_3 & \sqrt{3}(y_3 - y_c) \end{bmatrix} \tag{4.6}$$

Figure 4.1 illustrates a random target cell getting mapped into the standard triangle along with the vertex coordinates that are used to define the jacobian (J in equation 4.6) of the linear transformation.

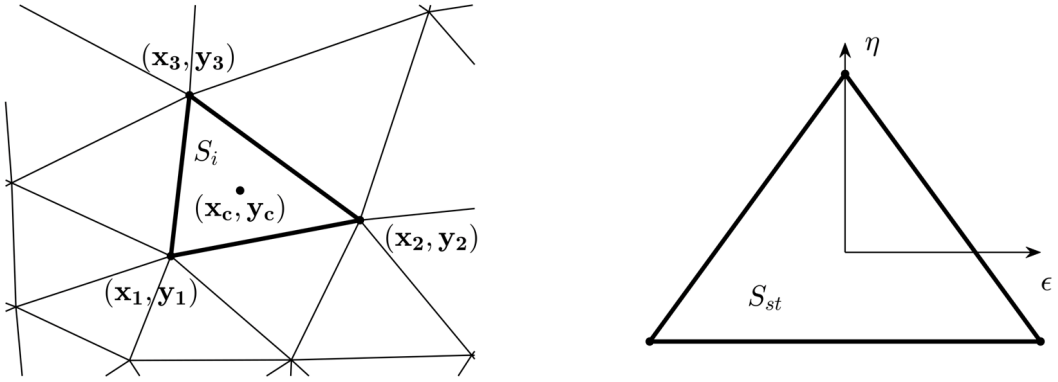

Figure 4.1: Mapping a random target cell (S_i) into the standard triangle (S_{st}).

Auxiliary Coord. Transformation for Square Elements

Each square target cell gets subjected to the linear transformation in equation 4.7 and mapped into square with unitary side.

$$\begin{bmatrix} x \\ y \end{bmatrix} = J \begin{bmatrix} \epsilon \\ \eta \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix}, J = \begin{bmatrix} |x_2 - x_1| & 0 \\ 0 & |y_2 - y_1| \end{bmatrix} \tag{4.7}$$

Figure 4.2 illustrates a random target cell getting mapped into the standard square along with the vertex coordinates that are used to define the jacobian (J in equation 4.7) of the linear transformation.

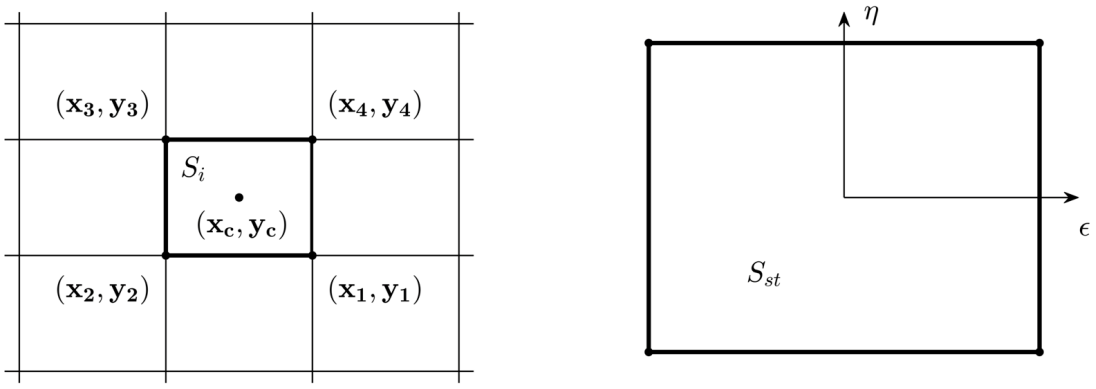

Figure 4.2: Mapping a random square target cell (S_i) into the standard square (S_{st}).

4.1.2 Stencil Creation

Each target cell must have a collection of data from which to regress the polynomial model. With this purpose, stencils ($\mathbb{S}$) are defined for each target cell. In the context of this thesis, a stencil is a collection of cells from which the required data is taken, as given by equation 4.8.

$$\mathbb{S}_{im} = \bigcup_{n=1}^{N_S} S_n \tag{4.8}$$

As aforementioned, WENO schemes require various polynomial regressions for propper functioning around discontinuities. A central and various directional stencils are for this reason used. It is also important that the data present in one diretional stencil is not present in the others as to minimize the probability of a discontinuity or shock intersecting various stencils. In this thesis, the methodology in [4] is followed and stencil search areas are defined through the angle formed by the center of the target cell and two of its vertices (see figure 4.3).

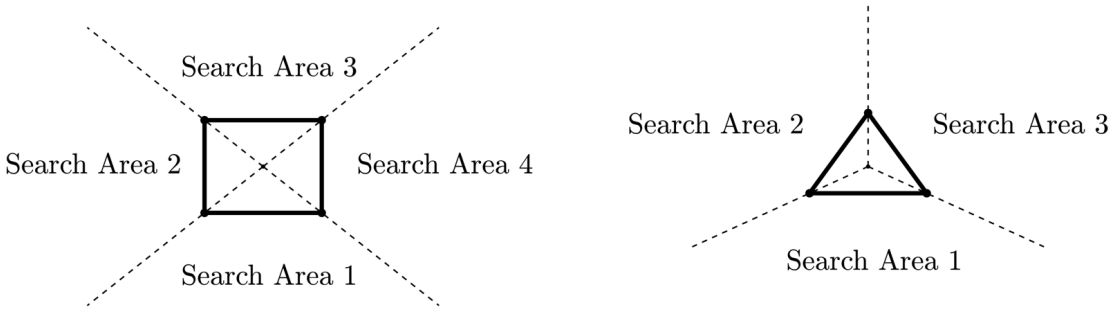

Figure 4.3: Stencil search areas for a square (left) cell and a triangular cell (right).

The number of stencils that each target cell has is dependent on its number of faces. A triangular cell has four stencils: one central and three directionals. A square cell has five stencils: one central and four

directionals. For the construction of the directional stencils, each neighbouring cell of the target cell is added to the stencil concerning the search area in which they are located. The neighbouring cells of the most recently added cells that are not already in a stencil receive the same treatment. The procedure is done until the desired number of cells in each stencil is reached (see figures 4.4).

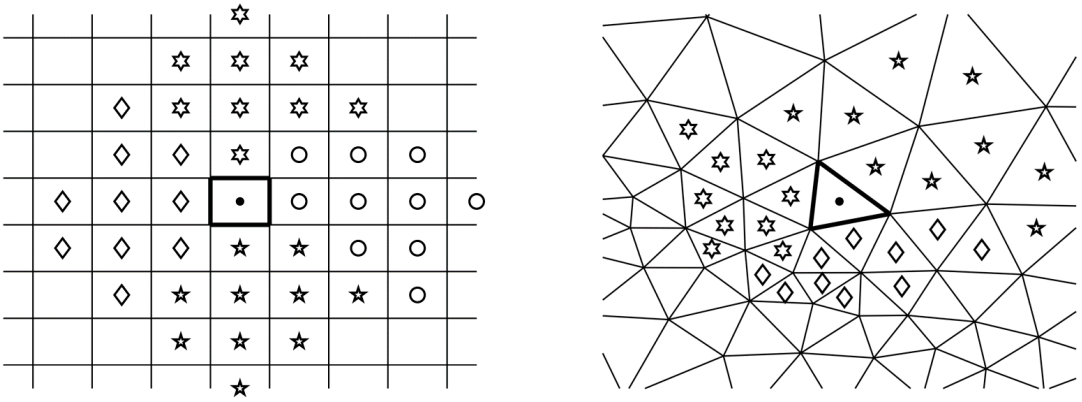

Figure 4.4: Diretional stencils for a square cell (left) and a triangular cell (right).

The number of cells (data) used in each stencil will be double the number of degrees of freedom in the polynomial model, as stated in [4] and [8]. This alternative was chosen as it provides flexible and robust behaviour for general meshes. Altough this thesis only concerns triangular and square meshes, it aims to expose the drawbacks and advantages of WENO schemes in general CFD supersonic applications. In equation 4.9 the number of data in each stencil (N_S) is given as a function of the the degree of the polynomial model used (p).

$$N_S = (\wp + 1)(\wp + 2) - 2 \tag{4.9}$$

The central stencil is made up of the closest cells to the target cell taken from the various diretional stencils (see figure 4.5).

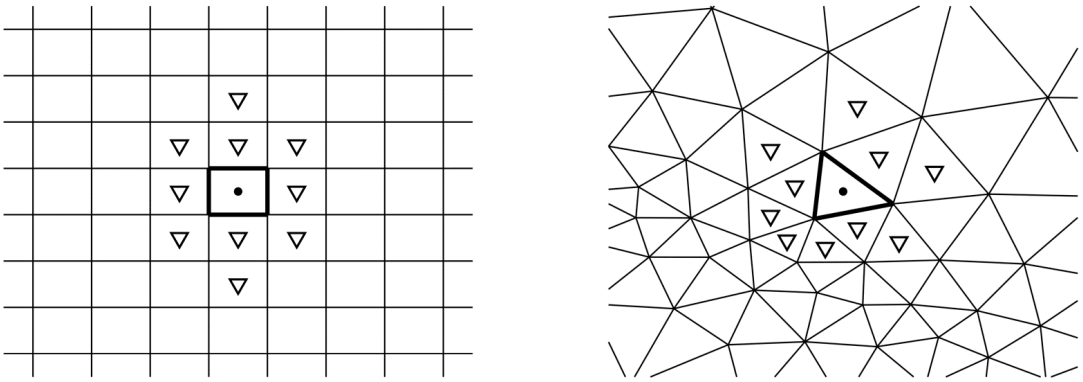

Figure 4.5: Central stencil for a square cell (left) and a triangular cell (right).

Stencil Creation Near To Boundaries

Next to boundaries, if a diretional stencil fails to be completed it is simply discarded (see figure 4.6). This option was chosen over completing the failed stencil with data from other stencils. Usage of data from other stencils may ensue in giving more weight to polynomial models resulting from intersection with discontinuties and shocks.

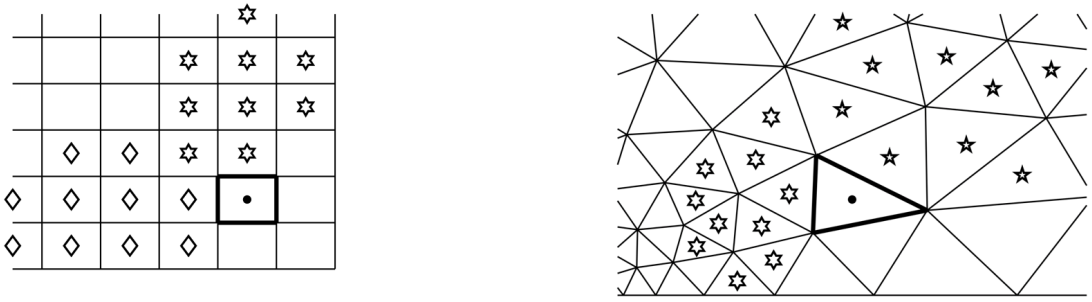

Figure 4.6: Stencils near boundaries for a square cell (left) and a triangular cell (right).

4.1.3 Least-Square Problem for Characteristic Variables

Having the necessary data sets (cells in central and diretional stencil) for a given cell and quantity, the Least-Squares Method can be used to approximate the polynomial model in section 4.1.1 to each one. A Least-Squares Problem and the respetive matrices must be setup. The procedure is described in this section.

$\mathbf{D^{LS}}$ Matrix

The Least-Square Problem is set up by approximating the average value of the polyomial model in each stencil cell ($\mathbb{S}_{imn}$) to that of the stencil cell itself. As the average value of the polynomial model is the sum of cell average of each basis functions ($\overline{B}_h(\mathbb{S}_{imn})$) multiplied by the respective coefficient (C_h), the D^{LS} matrix can be constructed using the averaged value of each basis function in the stencil cells. Equation 4.10 mathematically translates the process.

$$\overline{B}_h(\mathbb{S}_{imn}) = \frac{1}{A(\mathbb{S}_{imn})} \int_{\mathbb{S}_{imn}} B_h \; \partial \; \mathbb{S}_{imn} \tag{4.10}$$

Integration can be done in either the physical coordinate system or the auxiliary coordinate system, as linear mapping does not change cell averages.

The computation of the integral in equation 4.10 for triangular and square cells is described in the next two respective portions of this section.

$\mathbf{D^{LS}}$ Matrix - Integration in triangular cells: Integration in triangular cells is done through the use of a Gauss-Legendre quadrature (GLQ). The integration points (δ_g, ζ_g) and weights (W^{GL}) are defined for a unitary rectangle triangle (S_{ret}) and are taken from [9]. One can linearly map those same points into the corresponding ones in any cell in the stencil and fhe values of the basis functions at those points known $(B_h(\epsilon, \eta))$. The mathematical procedure is described in equations 4.11, 4.12 and 4.13.

$$\overline{B}_h(\mathbb{S}_{imn}) = \frac{1}{A(\mathbb{S}_{imn})} \int_{\mathbb{S}_{imn}} B_h(\epsilon_g, \eta_g)\, \partial\mathbb{S}_{imn} = \frac{1}{0.5} \int_{S_{ret}} B_h(\delta_g, \zeta_g)\, \partial S_{ret} \tag{4.11}$$

$$\Rightarrow \frac{1}{0.5} \int_{S_{ret}} B_h(\delta_g, \zeta_g)\, \partial S_{ret} = \frac{1}{0.5} \sum_{g=1}^{4} W_g^{GL} \cdot B_h(\delta_g, \zeta_g) \tag{4.12}$$

$$\Rightarrow \frac{1}{0.5} \sum_{g=1}^{4} W_g^{GL} \cdot B_h(\delta_g, \zeta_g) = \frac{1}{A(\mathbb{S}_{imn})} \sum_{g=1}^{4} W_g^{GL} \cdot B_h(\epsilon_g, \eta_g) \tag{4.13}$$

For this thesis, the Gauss-Legendre quadrature points are passed from the unitary rectangle triangle to the auxiliary coordinate system and not to the original coordinate system. For a random triangular cell in the target cells stencil, the Gauss-Legendre integration points for the unitary rectangle triangle are subjected to the linear transformation described in equation 4.14.

$$\begin{bmatrix} \delta_g \\ \zeta_g \end{bmatrix} = \begin{bmatrix} \epsilon_3 - \epsilon_1 & \epsilon_2 - \epsilon_1 \\ \eta_3 - \eta_1 & \eta_2 - \eta_1 \end{bmatrix} \begin{bmatrix} \epsilon_g \\ \eta_g \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \eta_1 \end{bmatrix} \tag{4.14}$$

Figure 4.7 illustrates the process of tranforming the unitary retangle triangle into a random triangle on a given stencil. The integration points are, thereby, passed onto this random triangle.

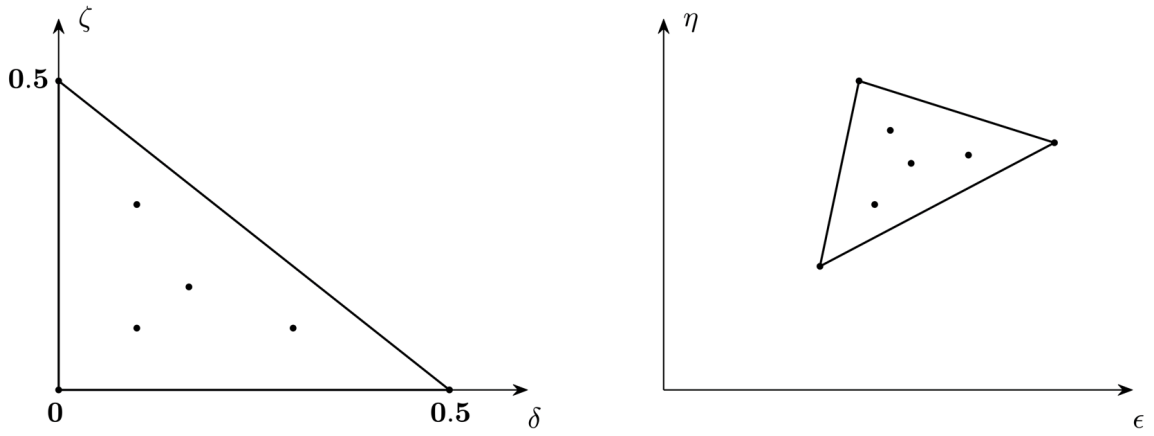

Figure 4.7: Mapping the unitary retangle triangle (left) and respetive GLQ points (black dots) into a random triangle (right).

$\mathbf{D^{LS}}$ Matrix - Integration in square cells: Altough integration in square elements within uniform meshes can be done analytically (as expressed in equation 4.15), for futher flexibility of the developed

code, the square cells are divided onto two triangles (S_{t1} and S_{t2}) and then each one subjected to the integration method previously described.

$$\overline{B}_h(\mathbb{S}_{imn}) = \frac{1}{A(S_n)} \int_{S_n} B_h(\epsilon_g, \eta_g)\ \partial S_n = \int_{y_c-\frac{1}{2}}^{y_c+\frac{1}{2}} \int_{x_c-\frac{1}{2}}^{x_c+\frac{1}{2}} B_h(\epsilon_g, \eta_g)\ \partial x \partial y \tag{4.15}$$

Figure 4.8 illustrates the process of tranforming the unitary retangle triangle into one of the two triangles that compose a random square cell on a given stencil. The integration points are, thereby, passed onto this random triangle.

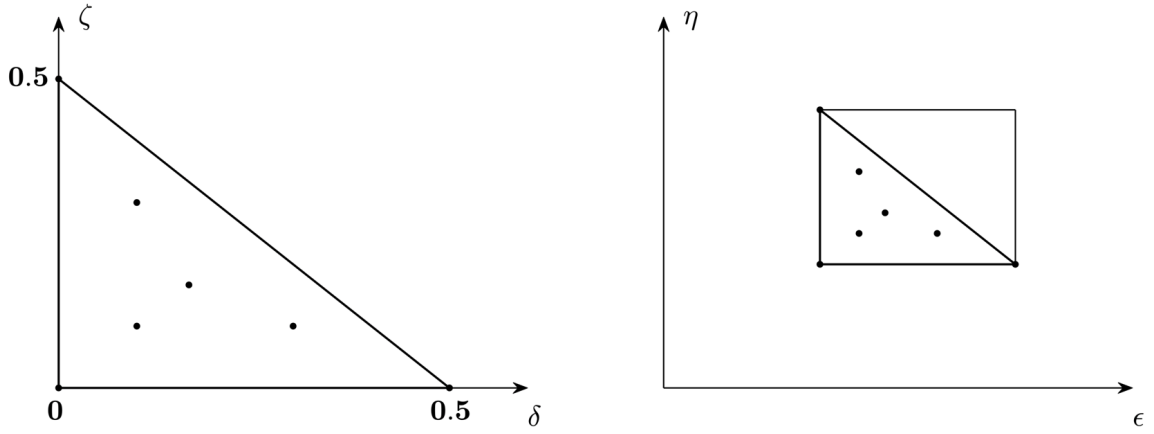

Figure 4.8: Mapping the unitary retangle triangle (left) and respetive GLQ points (black dots) into one of two triangles that compose a square cell (right).

After having the average of each triangle (S_{t1} and S_{t2}) that composes the square cell (S_n), the average of a given basis function $\overline{B}_h$ can be given by equation 4.16.

$$\overline{B}_h(\mathbb{S}_{imn}) = \frac{1}{A(\mathbb{S}_{imn})}\left(\int_{S_n} B_h(\epsilon_g, \eta_g)\ \partial S_{t1} + \int_{S_n} B_h(\epsilon_g, \eta_g)\ \partial S_{t2}\right) \tag{4.16}$$

$\mathbf{D^{LS}}$ Matrix - Final form: Finally, matrix D^{LS} can be constructed with the integrals of the basis functions on the stencil cells. The structure of the matrix is given by equation 4.17.

$$D_{im}^{LS} = \begin{bmatrix} \overline{B}_1(\mathbb{S}_{im1}) & \overline{B}_2(\mathbb{S}_{im1}) & ... & \overline{B}_K(\mathbb{S}_{im1}) \\ \overline{B}_1(\mathbb{S}_{im2}) & \overline{B}_2(\mathbb{S}_{im2}) & ... & \overline{B}_K(\mathbb{S}_{im2}) \\ \overline{B}_1(\mathbb{S}_{im3}) & \overline{B}_2(\mathbb{S}_{im3}) & ... & \overline{B}_K(\mathbb{S}_{im3}) \\ ... & ... & ... & ... \\ \overline{B}_1(\mathbb{S}_{imN_S}) & \overline{B}_2(\mathbb{S}_{imN_S}) & ... & \overline{B}_K(\mathbb{S}_{imN_S}) \end{bmatrix} \tag{4.17}$$

$\mathbf{Y^{LS}}$ Vector

As discussed in section 3.2.3, characteristic quantities provide smoother polynomials and, as such, are now used as the dependent variables of the data set. In each interface of a cell, the approximated

formulation of the two-dimensional Euler equations (discussed in section 2.3.3) is used. These equations are again shown in equation 4.18.

$$\frac{\partial Q_{n_{ij}}}{\partial t} + \frac{\partial F_{n_{ij}}}{\partial n_{ij}} = 0 \tag{4.18}$$

An approximated decoupled version of these equations, detailed in equation 4.19, is set on each face.

$$\frac{\partial(\widetilde{\mathbb{X}}_R^{-1} Q_n)}{\partial t} + \widetilde{\Lambda} \cdot \frac{\partial(\widetilde{\mathbb{X}}_R^{-1} Q_n)}{\partial n} = 0 \tag{4.19}$$

The approximated right eigenvectors matrix ($\widetilde{\mathbb{X}}_R$) and eigenvalue diagonal matrix ($\widetilde{\Lambda}$) are calculated through the use of Roe averaged quantities (denoted by a tilde). The structure of these matrices is shown in equation 4.20.

$$\widetilde{\mathbb{X}}_{Rij} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ \tilde{u}_n - \tilde{a} & \tilde{u}_n & 0 & \tilde{u}_n + \tilde{a} \\ \tilde{u_v} & \tilde{u_v} & 1 & \tilde{u_v} \\ \tilde{H} - \tilde{u}\tilde{a} & \frac{1}{2}\tilde{V}^2 & \tilde{u_v} & \tilde{H} + \tilde{u}_n\tilde{a} \end{bmatrix} \quad \tilde{\Lambda}_{ij} = \begin{bmatrix} \tilde{u}_n - \tilde{a} & 0 & 0 & 0 \\ 0 & \tilde{u}_n & 0 & 0 \\ 0 & 0 & \tilde{u}_n & 0 \\ 0 & 0 & 0 & \tilde{u_n} + \tilde{a} \end{bmatrix} \tag{4.20}$$

The Roe averaged quantities are, in turn, calculated with data from the target cell averaged quantities ($\overline{Q}_L$) and those of the neighbouring cell ($\overline{Q}_R$) that shares the common interface through the process in equation 4.21.

$$\tilde{u}_{ij} = \frac{\rho_L u_L + \rho_R u_R}{\rho_L + \rho_R}; \quad \tilde{H}_{ij} = \frac{\rho_L H_L + \rho_R H_R}{\rho_L + \rho_R}; \quad \tilde{a}_{ij} = ((\gamma - 1)(\tilde{H} - \tfrac{1}{2}\tilde{u}^2))^{\frac{1}{2}}; \quad \tilde{V}_{ij}^2 = \tilde{u}_n^2 + \tilde{u}_v^2 \tag{4.21}$$

For each interface, the data contained in each stencil must be converted to characteristic variables, according to the established approximate decoupled equations in equation 4.20. The methodology is shown in equation 4.20)).

$$\overline{Z}(\mathbb{S}_{imn}) = \tilde{R}_{ij}\overline{Q}(\mathbb{S}_{imn}) \tag{4.22}$$

The dependent variables vector (Y^{LS}) of the Least-Square Problem will be the difference between the cell average of a given characteristic variable (index *k*) in a given stencil entry and that of the target cell (see equation 4.23).

$$Y_{imjk}^{LS} = \begin{bmatrix} \overline{Z}_k(\mathbb{S}_{im1}) - \overline{Z}_k(S_i) & \dots & \overline{Z}_k(\mathbb{S}_{imNs}) - \overline{Z}_k(S_i) \end{bmatrix}^T \tag{4.23}$$

The process here described is also used in [4].

Weight Diagonal Matrix ($\mathbf{W^{LS}}$)

The use of a weight diagonal matrix is justified in [4], [3] and in section 2.2. The weight function used for the two dimensional is a version of the one present in the one-dimensional case (see section 3.1.3) and is described by equation 4.24.

$$W_{imnn}^{LS} = \frac{1}{\left(\epsilon(\mathbb{S}_{imn})^2 + \eta(\mathbb{S}_{imn})^2\right)}^{\frac{\wp}{2}} \tag{4.24}$$

The formulation of the $\mathbf{W^{LS}}$ matrix is expresssed in equation 4.25.

$$W_{im}^{LS} = \begin{bmatrix} \frac{1}{\left(\epsilon(\mathbb{S}_{im1})^2+\eta(\mathbb{S}_{im1})^2\right)}^{\frac{\wp}{2}} & & \\ & \ddots & \\ & & \frac{1}{\left(\epsilon(\mathbb{S}_{imN_S})^2+\eta(\mathbb{S}_{imN_S})^2\right)}^{\frac{\wp}{2}} \end{bmatrix} \tag{4.25}$$

4.1.4 WENO Polynomial Model

After obtaining the different polynomial models for a certain characteristic variable, the complete polynomial model in the respective cell and face results from a convex combination of the coefficients of those same models. As WENO schemes aim to avoid polynomials that exhibit the Gibbs phenomen, the weight given to each polynomial model is a function of how much they oscillate. The CWENO full polynomial model (M^{CW}) can be expressed as by equation 4.26.

$$M_{ijk}^{CW} = W_{ijk1}^{CW} \cdot M_{ijk1} + W_{ijk2}^{CW} \cdot M_{ijk2} + ... \tag{4.26}$$

The non-linear weight function (W^{CW}) is defined in equation equation 4.27.

$$W^{CW} = \frac{\Gamma}{\sum_{m=1}^{4||5} \Gamma_e}; \quad \Gamma = \frac{w}{(\iota+SI)^r} \tag{4.27}$$

Concerning equation 4.27, w is the linear weight of the stencil. Central stencils are normally given a higher value (for this thesis the value used is 10^3) as they are more accurate in smoother areas of the domain (see [4]). For the directional stencils it takes the value of unity ($w = 1$). SI is the smoothness indicator of the polynomial (see [4]) and is computed in accordance with equation 4.28.

$$SI = \sum_{i=1}^{\wp} \int_{S_{st}} \left(\frac{d^i}{dx_t{}^i} \cdot M(\epsilon, \eta)\right)^2 \partial S_{st} \tag{4.28}$$

The value of ι is usually very small ($\iota = 10^{-5}$) and is used to avoid divison by zero.

4.1.5 Face Flux Calculation

After obtaining the complete WENO polynomial models for a said face, the residual of each target cell may be computed. As such, numerical flux integration in every face is required. Knowing that integration in transformed spaces obeys to the property in equation 4.5, the process can be done on the auxiliary

coordinate system of said target cell and then passed to the original coordinate system. Equation 4.29 translates this process.

$$\frac{1}{d(S_{ij}(x,y))}\int_{S_{ij}} F(x,y)\cdot n_x + G(x,y)\cdot n_y \;\; \partial S_{ij} = \frac{1}{d(S_{ij}(\epsilon,\eta))}\int_{S_{ij}} F(\epsilon,\eta)\cdot n_\epsilon + G(\epsilon,\eta)\cdot n_\eta \;\; \partial S_{ij} \quad (4.29)$$

As all target cells get their faces tranformed into the ones in the standard cells, it is possible to, without too much effort, analytically compute the line integral in equation 4.29. However, as the use of GLQ methods provides code that is both simpler and more flexible, this route was chosen. A bijective parametrization (q_j) of the faces in the standard cells in question is then required. The format for this parametrization is exposed in equation 4.30.

$$q_j : [0,1] \to l_j \Rightarrow \begin{matrix} \epsilon(s) = a_j^\epsilon s_j + b_j^\epsilon \\ \eta(s) = a_j^\eta s_j + b_j^\eta \end{matrix} \quad (4.30)$$

Figure 4.9 illustrates how each face (curve/path) can be expressed in terms of s_j.

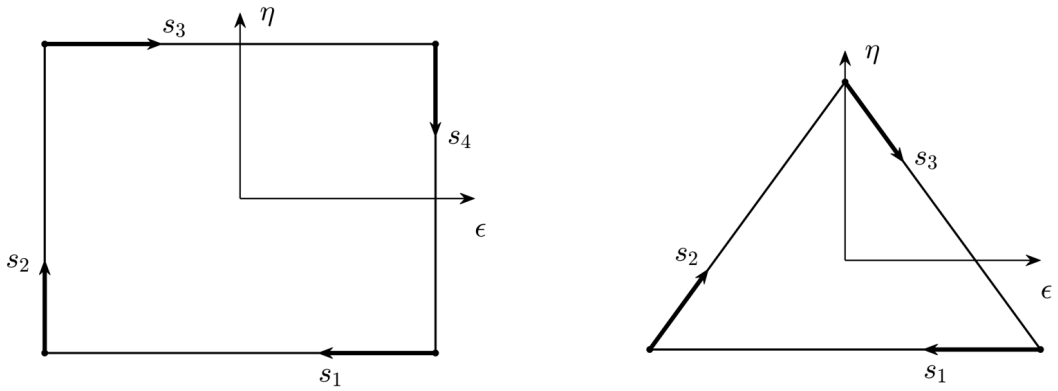

Figure 4.9: Parametrization for the faces of standard square (left) and triangle (right).

One can express the numerical fluxes normal to a face as F^n for simplicity, as established in section 2.3.3 and again stated in equation 4.31.

$$F^n(\epsilon,\eta) = F(\epsilon,\eta)\cdot n_x + G(\epsilon,\eta)\cdot n_y \quad (4.31)$$

Using the bijective parametrization in equation 4.30, the integration in equation 4.29 may be developed into equation 4.32.

$$\int_{S_{ij}} F^n(\epsilon,\eta)\;\partial S_{ij} = \int_0^1 F^n(\epsilon(s),\eta(s))\cdot|q_j^{'}(s)|\;\partial s \quad (4.32)$$

The term $|q'(s)|$ in equation 4.32 conects the lenght of ∂s to the one of ∂S_{ij} and is expressed by equation 4.33.

$$|q'(s)| = \sqrt{\left(\frac{d\epsilon(s)}{ds}\right)^2 + \left(\frac{d\eta(s)}{ds}\right)^2} \tag{4.33}$$

In equation 4.34, the line integral in equation 4.33 is substituted by a Gauss-Legendre quadrature with points (s_g) and weights (W_g^{GL}) for the $[0, 1]$ interval, taken from [9]).

$$\int_0^1 FN(\epsilon(s), \eta(s)) \cdot |q_j^{'}(s)| \, \partial s = \sum_{g=1}^{4} W_g^{GL} \cdot |q_j^{'}(s)| \cdot FN(\epsilon(s_g), \eta(s_g)) \tag{4.34}$$

Figure 4.10 illustrates how the GLQ points (s_g) get transfered into each face (curve/path), by using the bijective parametrization in 4.30.

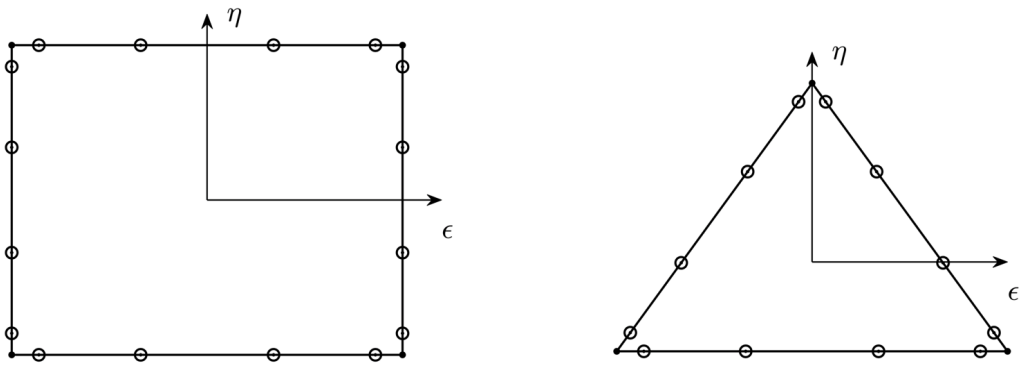

Figure 4.10: Projection of GLQ points (black circles) into the faces for the standard square (left) and triangle (right) cells.

Computation of the flux integral in 4.34 depends on knowing the original euler variables (Q) at the GLQ points. Consequently, the full WENO polynomials for the characteristic quantities (Z) are used to approximate the actual characterictic quantities at those same points, as demonstrated in equation 4.35.

$$Z(\epsilon(s_g), \eta(s_g)) \approx \left[M_{ij1}^{CW}(\epsilon(s_g), \eta(s_g)) \qquad ... \qquad M_{ij4}^{CW}(\epsilon(s_g), \eta(s_g)) \right]^T \tag{4.35}$$

The approximate characteristic quantities can then be used to calculate the original Euler quantities, through equation 4.36.

$$Q(\epsilon(s), \eta(s)) \approx \widetilde{R}_{ij} \cdot Z((\epsilon(s), \eta(s))) \tag{4.36}$$

For the same GLQ point, two different values for the same quantity are calculated: one coming from the WENO polynomial for the target cell (Q_L) and the other from that of the neighbouring cell (Q_R) that shares the face in question. A Riemann problem is set in each integration point with the left state using the values from the target cell (Q_L) and the right state those of the neighbouring cell (Q_R). As expressed in equation 4.37, the HLLC approximate Riemann solver is used to calculate the flux on each integration point.

$$F^n(\epsilon(s_g), \eta(s_g)) \approx HLLC(Q_L, Q_R) \tag{4.37}$$

The resultant flux can then passed to the original coordinate system using the equality in equation 4.29. The residual for each cell on the current time-step can then calculated with equation 4.38.

$$R_i = \sum_{j=1}^{3||4} \int_{S_{ij}} F^n(\epsilon, \eta)\ \partial S_{ij} \tag{4.38}$$

4.1.6 Flux limiting for the 2D-WENO scheme

In some test cases, neither having different directional stencils nor approximating local characteristic variables was enough to ensure that the resulting WENO polynomial did not exhibit the Gibbs phenomenon. Due to the fact that all stencils may encounter discontinuities, either by close proximity of two or more discontinuities, or by deficiencies in the search areas, positivity may not be ensured and simulations fail completely. A single cell is enough to ruin hours of simulation. A simple flux limiting strategy similar to MOOD [10] (Multidimensional Optimal Order Detection) WENO schemes, which alternates between different orders of spatial accuracy, was implemented on this thesis. The quantities extrapolated on the face are checked for positivity (pressure, density and energy) and for monotonicity. If the previous properties are not respected, all the quantities on the face (Q_L and Q_R) are extrapolated using a first-order cell-centred approximation, present in equation 4.39, which is unconditionally monotonicity preserving.

$$Q_L(\epsilon(s_g), \eta(s_g)) = \overline{Q}_L \qquad Q_R(\epsilon(s_g), \eta(s_g)) = \overline{Q}_R \tag{4.39}$$

The full flux scheme assumes a structure similar to that of a traditional TVD scheme, in which the high order approximation is used only when it complies with the necessary requirements. A simple toggle function (t_w) is used to turn on and off the high-order scheme, as evident in equation 4.40.

$$Q = Q^{Godunov} + t_w(Q^{WENO} - Q^{Godunov}) \tag{4.40}$$

4.1.7 CWENO, triangular meshes and shocks

As was found while developing code and running test cases for the scheme presented in this section, the combination of CWENO schemes and triangular meshes is very ill-equiped to deal with shock proximity. This major hindrance comes from the fact that the directional stencils for a triangular element tend to all intersect a shock in the vicinity. As normal WENO schemes mantain monotonicity and positivity by having at least one stencil with sufficiently smooth data, a triangular mesh, altough an adequate choice for complex geometries, maybe not the best choice for use along with the numerical scheme presented in this chapter. Figure 4.11 illustrates the behaviour of stencil search areas near shocks and discontinuities.

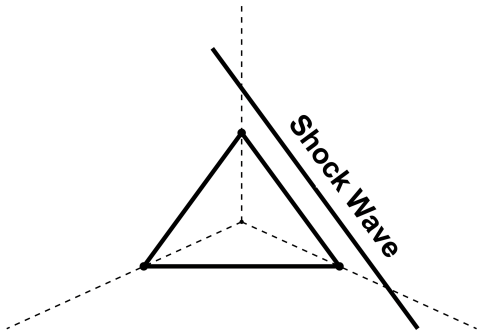

Figure 4.11: Behaviour of the stencil search areas for a triangular cell when in proximity of a shock.

4.1.8 Time integration

As evident in equation 4.41, the time integration method is the same as that of the one-dimensional WENO scheme developed in this thesis.

$$\overline{Q}(S_i, t_0+) = \overline{Q}(S_i, t_0) + \frac{\Delta t}{A(S_i)} \cdot \frac{1}{6}(K_{i1} + 4K_{i2} + K_{i3}) \tag{4.41}$$

The terms K_{i1}, K_{i2} and K_{i3} in equation 4.41 are defined by equation 4.42.

$$K_{i1} = R_i(\overline{Q}_{t_0}) \qquad K_{i2} = R_i(\overline{Q}_{t_0} + 0.5 \cdot k_1 \cdot \Delta t) \qquad K_{i3} = R_i(\overline{Q}_{t_0} + 0.5 \cdot k_2 \cdot \Delta t) \tag{4.42}$$

In equation 4.43, the time step (Δt) is calculated in such a way that preserves stability of the temporal scheme. The configuration for the time step was taken from [4].

$$\Delta t = \mu \frac{\min(d(S_i))}{2 \cdot \max|\lambda|} \tag{4.43}$$

The Courant–Friedrichs–Lewy number provides control over the size of the time-step, emphasizing either stability or computational expense. In the two dimentional case, the term $d(S_i)$ is computed as an hidraulic diameter, as given by equation 4.44.

$$d(S_i) = \frac{4 \cdot A(S_i)}{\sum_{j=1}^{3||4} d(Sij)} \tag{4.44}$$

The term $\max|\lambda|$ is the fastest characteristic speed in all the domain, at that particular time iteration.

4.2 2D Test Cases

4.2.1 Diffusive Flux Reconstruction on Gaussian Function

A two-dimensional Gaussian function was injected on a domain discretized with either square or triangular cells. The point-wise values are set as shown in equation 4.46.

$$U(x,y) = e^{-\frac{(x-0.5)^2+(y-0.5)^2}{0.0175}} \tag{4.45}$$

Taken from [11], the two-dimensional linear diffusion equation in 4.46 served as the governing equation for quantity U.

$$\frac{\partial U}{\partial t} + \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0 \tag{4.46}$$

To evaluate the order of error decay of the full WENO reconstructions, the spatial derivative in x at initial time was evaluated in the face GLQ integration points (see figure 4.12). These same values were compared with the analytic values, and the point-wise errors calculated. The y spatial derivative is equal to the x spatial derivative when rotated ninety degrees so it was ignored.

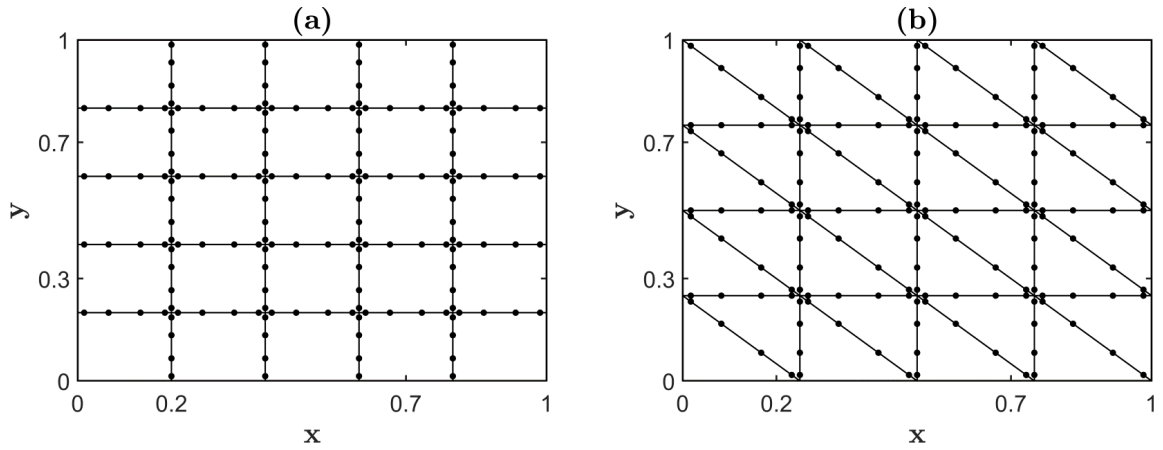

Figure 4.12: Face integration points (black dots) for up to seventh-order WENO for (a) square and (b) triangular meshes.

Figure 4.13 shows the L_2 norm for the error in spatial derivatives in the GLQ integration points as a function of average mesh size (d_{ref}), for both square and triangular meshes. It should be noted that, as the WENO polynomials were derivated with respect to the x-coordinate, the order of accuracy for the extrapolated quantities lowers. Consequently, the order of accuracy for the required derivatives is equal to that of the WENO scheme minus unity.

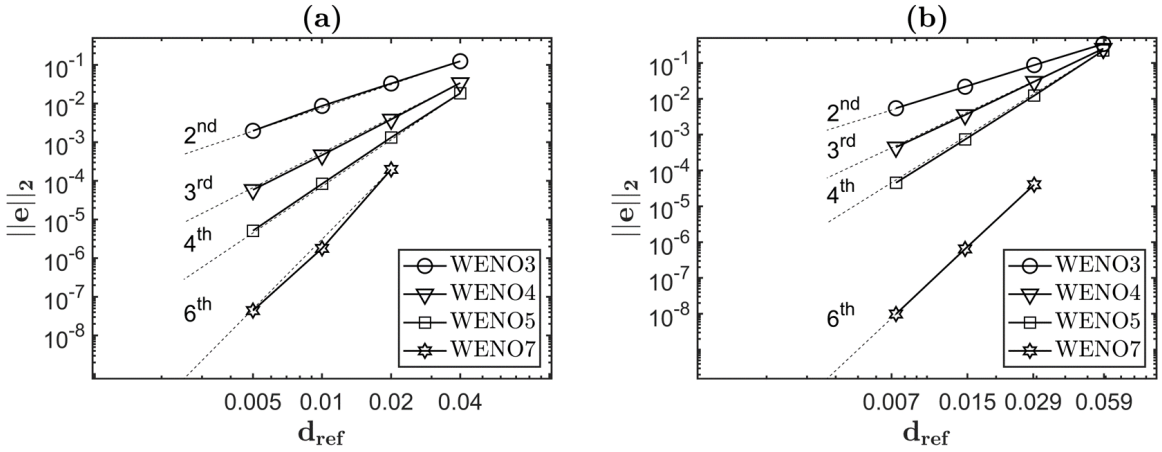

Figure 4.13: x derivatives - L_2 norm of the error in the spatial derivatives in x as a function of average cell size (d_{ref}) for (a) square and (b) triangular meshes.

In both triangular and square meshes, almost all WENO reconstructions followed the expected order of error decay. Only the seventh-order WENO in the square mesh presented a slight deviation from the reference (intermittent line). With the passage from coarser meshes to finer ones, the average number of sucessfull stencils for each cell increases, giving a boost in accuracy beyond that of the reduction in truncation error. The triangular mesh presented slightly worse overall results.

To evaluate the face flux integration method used, the L_2 norm of the error in flux for the non-boundary face fluxes was calculated for different levels of mesh refinement. As the absolute value of the integration on the face depends on the extrapolated point values multiplied by the face lenght, the expected order for the decay of the L_2 error norm is equal to that of the order of accuracy of the scheme. The results are shown in figure 4.14.

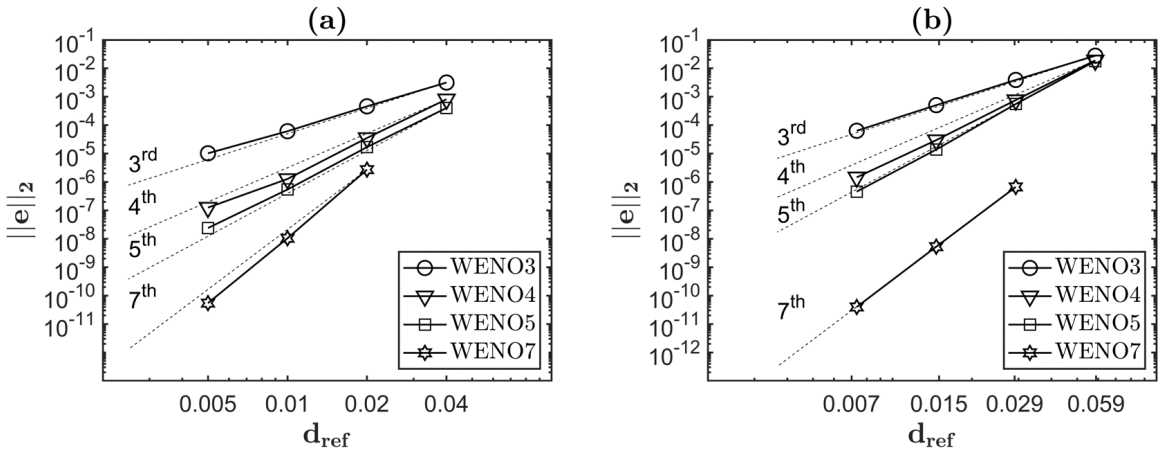

Figure 4.14: Diffusive face fluxes - L_2 norm of the error in the diffusive flux for the (a) square and (b) triangular meshes.

As may be seen in figure 4.14, for the square mesh, the fifth and seventh-order WENO schemes surpassed the predicted order of accuracy. The third and fifth-order WENO schemes provided slightly less error decay than expected. For the triangular mesh, all schemes followed the expected order very closely.

Analysing the results, one can state that the combination of the flux integration method and interpolating method sucessfully provided results similar to those promised by the numerical method design (see [4]).

For the domain boundary fluxes, the scheme also provided the expected order of error decay. The decay of L_2 norm of the error in the fluxes at the boundary for both triangular and regular meshes is presented in appendix A (A).

The average condition number of all P matrices used ($\overline{\kappa}$) dependency on mesh reference size (d_{ref}) was sudied to provide information on the overall conditioning of the flux scheme (figure 4.15). The correct functioning of auxilliary coordinate systems was evaluated.

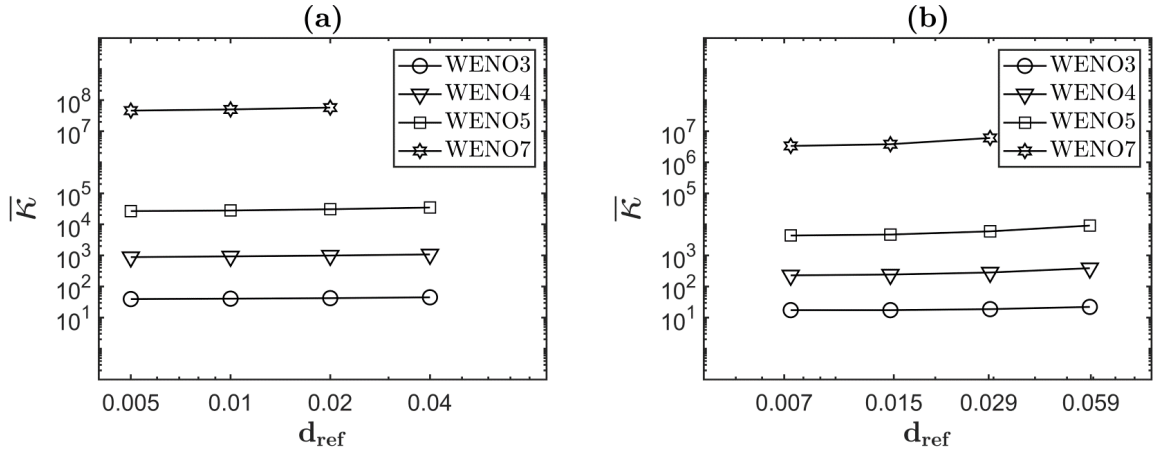

Figure 4.15: Average condition number of P matrices (d_{ref}) for the (a) square meshes and (b) triangular meshes.

The results in figure 4.15 show that the average condition number did almost not change with the different levels of mesh refinement. This is due to both the use of same-size triangular and square cells, and the correct functioning of the code responsible for passage of the mesh onto the sugested auxilliary coordinate system (see 4.1.1), in which the P matrices are computed. For the range of error displayed in the test cases, the matrices used were found to be sufficiently conditioned.

4.2.2 2D Sod's Shock Tube

The SST test case, discussed in 2.4.2 and also present in section 3, was implemented for a two-dimensional domain ($x \in [0, 1]$, $y \in [0, 25]$) discretized with either square or triangular cells. The two-dimensional Euler equations (see section 2.3.3) acted as the governing equations. Null Neumann boundary conditions were applied to all the boundaries. Figure 4.16 illustrates how the meshes were created for this particular case.

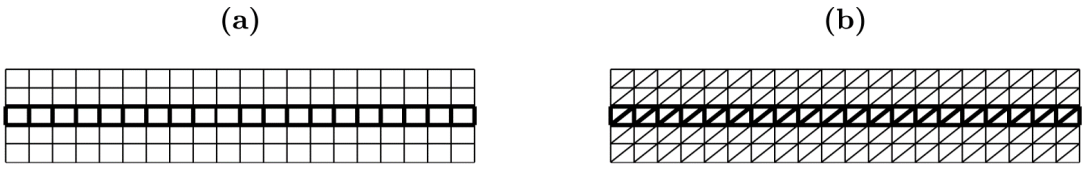

Figure 4.16: Example of a (a) square and (b) triangular mesh used for the test case with the cells coincident with the xx axis highlighted.

The Euler quantities at initial time were set in a similar way to the one-dimentional counterpart. Equation 4.47 shows how the point-wise value were set.

$$Q(x,y) = \begin{bmatrix} \rho, & \rho u, & \rho v, & E \end{bmatrix} = \begin{cases} \begin{bmatrix} 1, & 0, & 0, & 2.5 \end{bmatrix}, & x \leq 0.5 \\ \begin{bmatrix} 0.1, & 0, & 0, & 0.125 \end{bmatrix}, & x > 0.5 \end{cases} \tag{4.47}$$

The test case served as a stepping stone for applying the developed WENO scheme for more complex two-dimensional wave structures. Having only discontinuities and shocks perpendicular to the xx direction, the inter-cell fluxes were calculated for the line of cells that intersect with the xx symmetry axis (see figure 4.16) and assumed constant along yy. However, the functions used were coded for the two-dimensional version of the Euler equations (2.3.3). The two-dimensional WENO scheme was compared with other popular schemes, such as a two-dimensional version of the Godunov and MUSCL schemes. The used MUSCL scheme for triangular and square cells are distinct, with the MUSCL scheme for the regular mesh being taken from [5] and the MUSCL scheme for the triangular mesh taken from [12]), a version based on the use of the Least-Squares method for gradient reconstruction. The code was left to run until it reached $t = 0.1s$. The Courant–Friedrichs–Lewy number was set by comparing the order of magnitude of the temporal discretization errors to those of spatial discretization and ensuring that the latter were of a higher order of magnitude. A value of zero point three (CFL=0.3) was found to be sufficient.

The accuracy of the schemes used was acessed by arranging the L_2 norm of the error for the cell averaged density at $t = 0.1s$ as a function of the average cell size (d_{ref}). These results are shown in figure 4.17. The computational efficiency of the numerical schemes is evaluated by exposing the same L_2 norm as a function of solver run-time (SRT), as illustrated in figure 4.18.

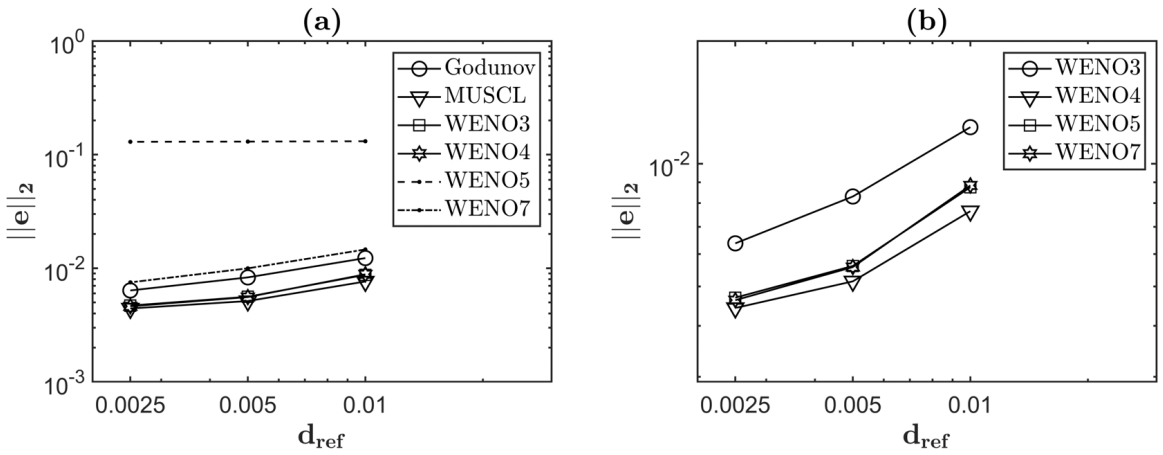

Figure 4.17: Square meshes - L_2 norm of the error in density as a function of average cell size (d_{ref}) at $t = 0.1s$ (without flux limiting) with (b) detail over the WENO schemes.

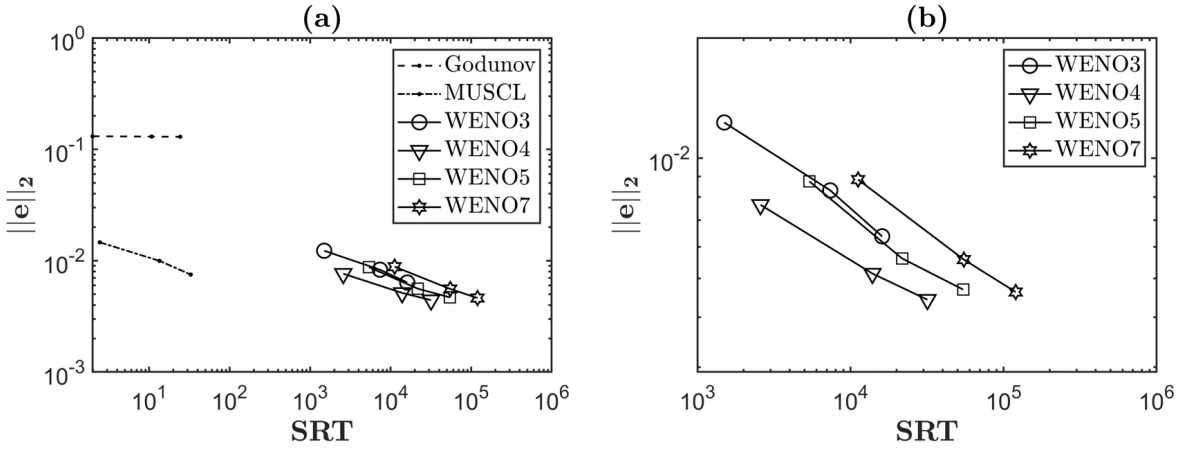

Figure 4.18: Square meshes - L_2 norm of the error in density as a function of solver run-time (SRT, in seconds) at $t = 0.1s$ (without flux limiting) with (b) detail over the WENO schemes.

The results in figure 4.17 show that the WENO schemes successfully surpassed the popular schemes (Godunov and MUSCL), providing better overall accuracy. The Godunov scheme, once again, provided the worst accuracy. It should be reported that, from the fifth order WENO onward, small wiggles intensify and contaminate the solution. These wiggles (presented in figures 4.19 and 4.20) introduce error, limiting the error decay of higher order schemes (fifth and seventh). As such, of all the schemes, the fourth-order scheme provided the best results. As flow structures progress from the initial discontinuities, WENO schemes may not be forced to deal with limited physical separation from discontinuities and shock waves leading to intersection with data discontinuities and all the stencils. From figures (figures 4.19 and 4.20), it becomes apparent that the oscillatory behaviour increases with stencil size and, consequently, with the promised degree of accuracy of the WENO scheme. Comparing these results with the ones for the test case in section 3.2.3, which was run to $t = 0.2s$, it can be noted that added separation between the shock wave and contact discontinuity and a more developed expansion benefited the higher-order schemes. Those conditions allowed them to function with less osccillatory behaviour and better surpass the lower-order schemes.

Figure 4.18 suggests that, for this range of error, the MUSCL scheme provided the greatest computational efficiency. For the WENO schemes, the fourth-order WENO provided the best computational efficiency. All schemes provided better computational efficiency than the Godunov scheme. It should be noted that this test case favoured the MUSCL scheme as it presents a fairly linear expansion wave and a domain dominated by a contact discontinuity and a shock wave with limited spatial separation. The WENO schemes are, as stated before, far more computationally intensive than the Godunov and MUSCL schemes.

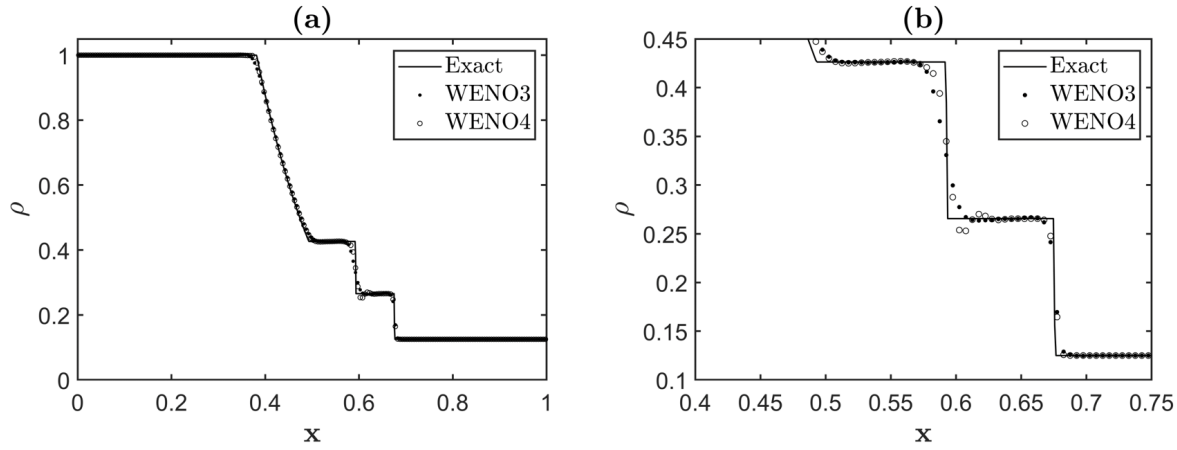

Figure 4.19: Square meshes - Density as a function of the x-coordinate for the third and fourth-order WENO schemes at $t = 0.1s$ (without flux limiting).

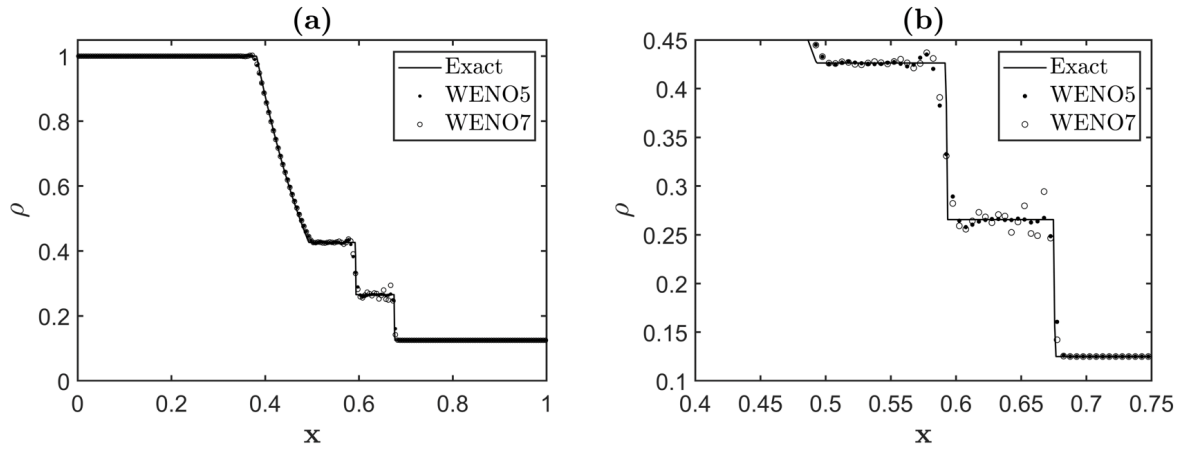

Figure 4.20: Regular meshes - Density as a function of the x-coordinate for the fifth and seventh-order WENO schemes at $t = 0.1s$ (without flux limiting).

As an attempt to reduce oscilattory behaviour in the WENO schemes, the flux limiting strategy present in section 4.1.6 was implemented. The results for the regular mesh combined with the use of flux limiting are present in figures 4.21, 4.22, 4.23 and 4.24. Figure 4.21 concerns the L_2 norm of the error in density at $t = 0.1s$ as a function of cell average size (d_{ref}) and figure 4.22 concerns the same L_2 error norm as a function of solver run time (SRT).

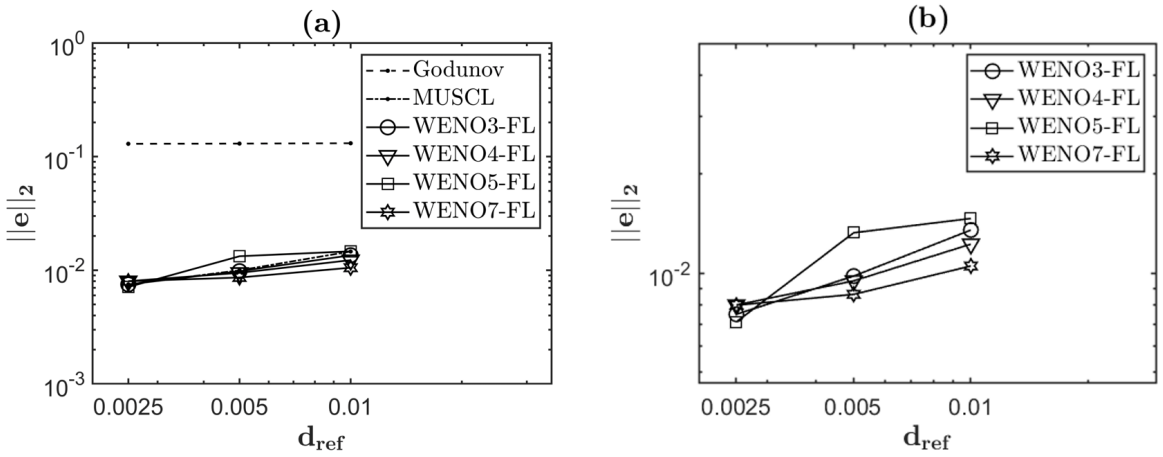

Figure 4.21: Square meshes - L_2 norm of the error in density as a function of average cell size (d_{ref}) at $t = 0.1s$ (with flux limiting) with (b) detail over the WENO schemes.

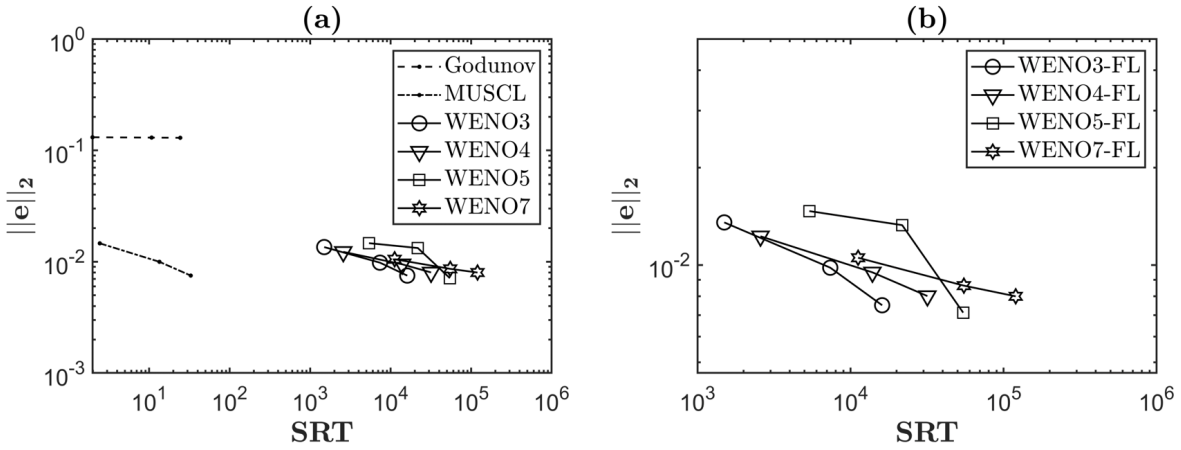

Figure 4.22: Square meshes - L_2 norm of the error in density as a function of solver run-time (SRT) at $t = 0.1s$ (with flux limiting) with (b) detail over the WENO schemes.

Despite killing the wiggles, the flux limiter also highly reduced the accuracy of the WENO schemes. The computational efficiency, consequently, also suffered. The overall accuracy of the WENO schemes closely resembled that of the MUSCL scheme with the latter surpassing that of almost all the WENO schemes for the finer mesh. With higher mesh refinement, the discontinuities in the cell averaged quantities get sharper and, to a certain extent, increase oscillatory behaviour. Another plausible justification for this scenario is that, as the mesh gets more refined, small wiggles have a higher chance of triggering the flux limiting, thus, reverting the scheme to a first-order accurate scheme. The source of these wiggles can be due to intersection of discontinuities and shocks by the stencils or even, in a much lesser scale, due to insufficient conditioning of the matrices. As the test case present in 4.2.1 proved that, for smooth domains, the WENO schemes followed the desired order of decay in lower ranges of error, this last explanation may be discarted. Mesh refinement provided better accuracy in smoother regions of the domain and the fifth-order WENO scheme was able to out-perform all other schemes in the finest mesh used here.

For the flux limited WENO schemes, the third-order WENO provided the best computational effi-

ciency followed by the fourth-order WENO. The MUSCL schemes, unsurprisingly, provided the best computational efficiency. The Godunov scheme provided the worst computational efficiency.

The non-oscillatory behaviour of the flux limited WENO schemes can be viewed in figures 4.23 and 4.24 where density is displayed as a function of the x -coordinate.

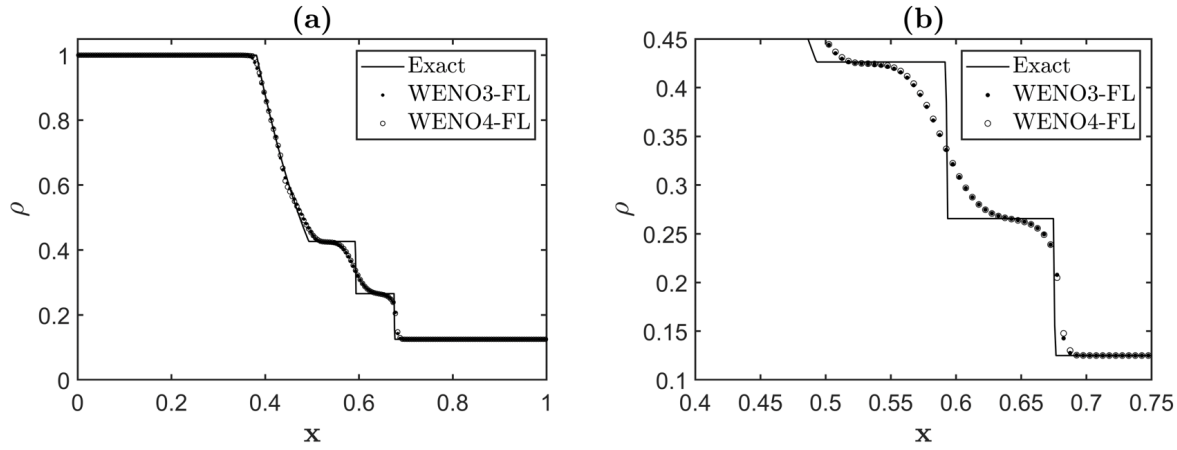

Figure 4.23: Square meshes - Density as a function of the x-coordinate for the third and fourth-order WENO schemes at $t = 0.1s$ (with flux limiting).

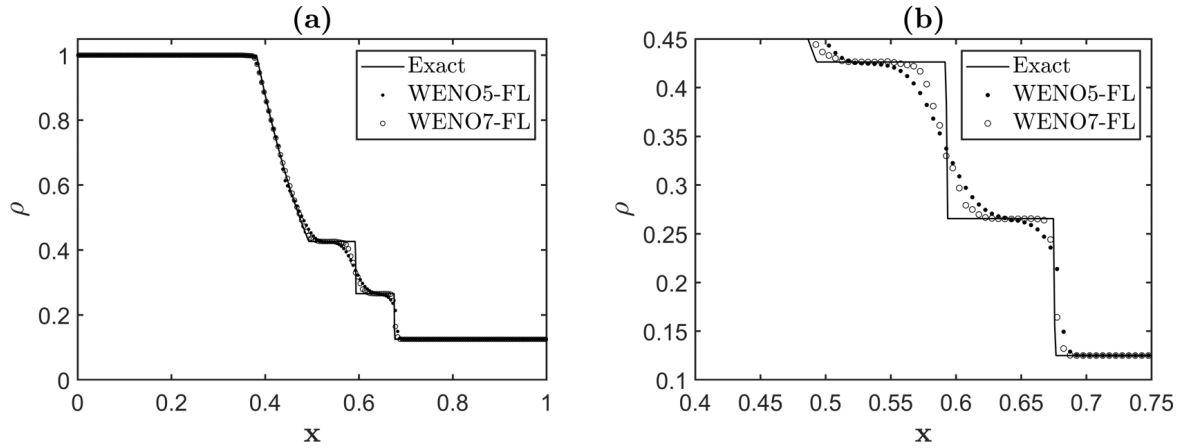

Figure 4.24: Square meshes - Density as a function of the x-coordinate for the fifth and seventh-order WENO schemes at $t = 0.1s$ (with flux limiting).

For the triangular meshes, the WENO schemes only worked in conjunction with flux limiting, as the deficiency in stencil search areas described in 4.1.7 resulted in total failure of the computed solutions.

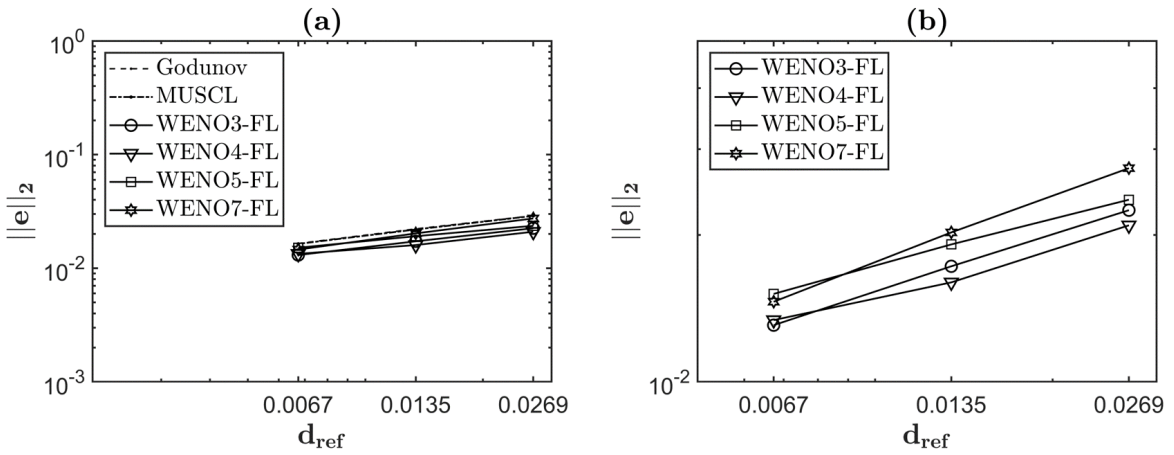

Figure 4.25: Triangular meshes - L_2 norm of the error in density as function of average cell size (d_{ref}) at $t = 0.1s$ (with flux limiting) with (b) detail over the WENO schemes

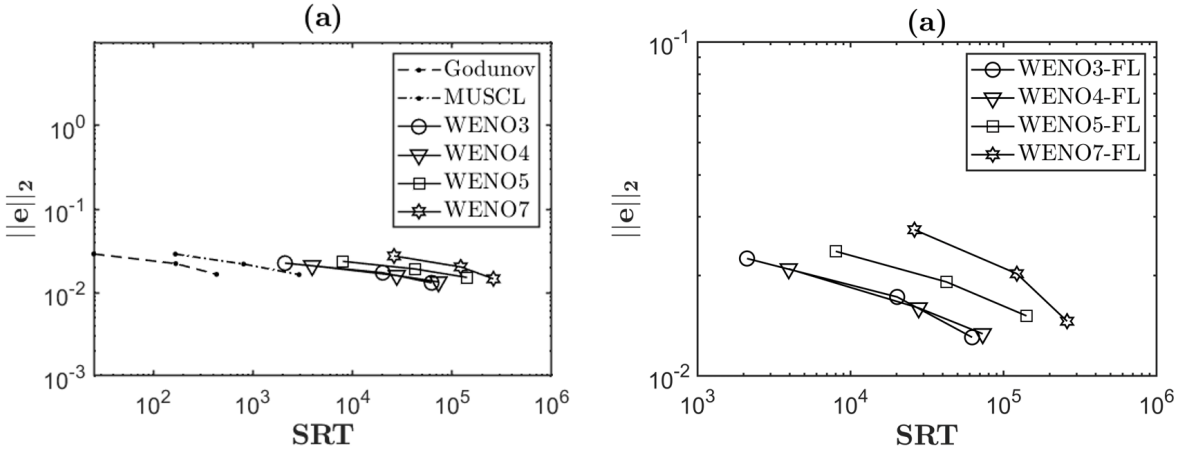

Figure 4.26: Triangular meshes - L_2 norm of the error in density as function of solver run-time(SRT) at $t = 0.1s$ (with flux limiting) with (b) detail over the WENO schemes.

From figure 4.25, the fourth-order WENO exhibited the least amount of error for the different degrees of mesh refinement. All other schemes provided less error than the Godunov scheme but the discrepancy is not as big as the one found when using regular meshes. This can be explained by the scheme reverting to first-order accuracy when the previously described deficiency in the triangle search areas contributed to producing wiggle ridden polynomials. The accuracy of the popular schemes also suffered from the use of trianuglar cells.

Figure 4.26 provides the information that the Godunov scheme exhibited the best computational efficiency. The MUSCL scheme provided better computational efficiency than the WENO schemes. The fourth-order and third-order WENO schemes provided the best computational efficiency of the WENO schemes.

In figures A.3 and A.4 of appendix A (A), density is given as a function of the x-coordinate for the MUSCL, Godunov, third and fourth WENO scheme. The fifth and seventh-order WENO schemes gave results very similar to those of the previous two WENO schmes and, thus, are not shown.

4.2.3 Sedov Blast Wave

A cylindric symmetric blast wave test case was implemented on a square domain ($[0, 0.5] \times [0, 0.5]$). In the typical setup [13], a discrete amount of energy (E_{Blast}) is distribuited by the cell(s) that contain an initial blast region propagating radially from the center of the domain. However, in the prospect of giving the proposed WENO scheme some leverage, the exact solution for $t = 0.05$ was injected into the domain to serve as the initial conditions. This choice provided a larger area of smooth quantities and reduced overall computational time. The exact solution was taken from [13] and then compared to [14] to establish the needed experimental parameters. Because of the symmetry in the xx and yy axis, only a quarter of the blast wave is considered. The left and bottom boundaries received symmetry boundary conditions and the upper and right boundaries freeflow boundary conditions. The code was left to iterate until it reached $t = 0.06$. The time-step was set by achieving temporal convergence with the Godunov scheme within appropriate orders of magnitude. Only the flux-limited fourth-order WENO scheme, which showed the best results in the two-dimensional shock tube test case, is compared to the Godunov and MUSCL schemes. The test is run for triangular and regular meshes (figure 4.27). The triangular mesh was obtained by dividing each square cell into two triangular ones. The orientation of the triangular mesh was taken from [14] and aims to align the gradients of the solution to the face normals.

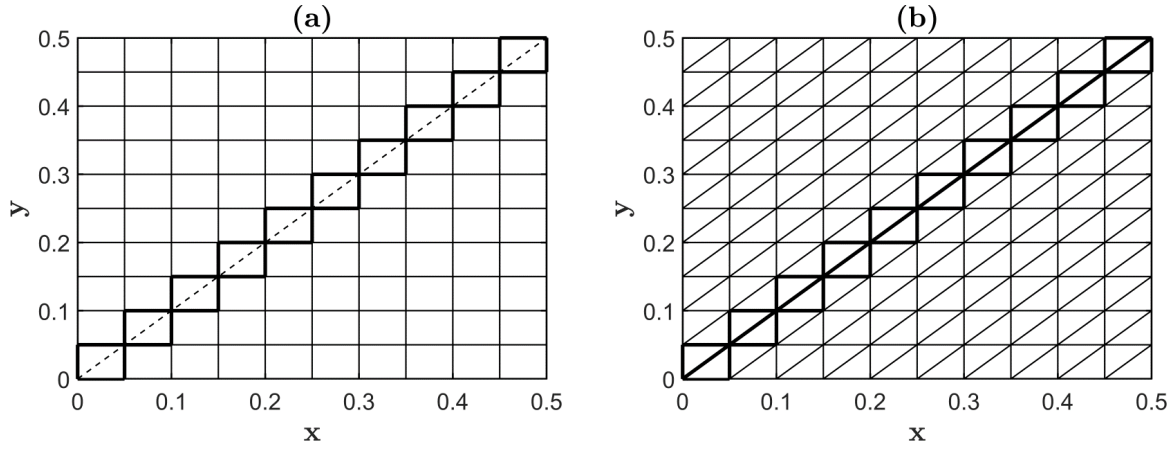

Figure 4.27: Example of meshes for (a) square and (b) triangular cells with the diagonal and adjacent cells highlighted.

A metric similar to the L_2 norm was defined in equation 4.48 ($||e||$) and used to quantify the error in density at $t = 0.06$ along the diagonal of the domain. The diagonal of the domain is highlighted through an intermitent black line in figure 4.27. This set of cells was chosen to study the behaviour of the schemes with diminished interference from the domain boundaries, which were set to have first-order accuracy. This option also provided more aesthetically pleasing results and figures as the level of mesh refinement here used is low.

$$||e|| = \sum_{i \in D} d_D \cdot (\overline{\rho}_i - \overline{\rho}_i^{exact})^2 \tag{4.48}$$

Concerning the previous equation, D is the set of cells adjacent to the diagonal, d_D the lenght of the

section of the diagonal that intersects cell i and $\overline{\rho}_i^{exact}$ the spatial average of the exact solution along that stretch of the diagonal.

As a way to compare accuracy, the metric $||e||$ of the error in density along the diagonal is given as a function of mesh reference size (d_{ref}) in figure 4.28.

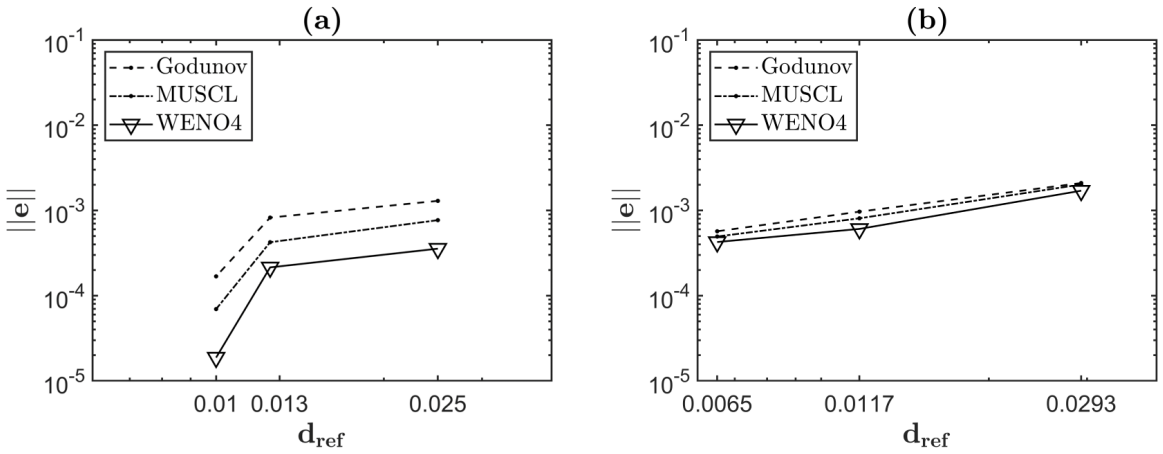

Figure 4.28: $||e||$ metric of the error in density along the diagonal as function of mesh reference size (d_{ref}) for (a) square and (b) triangular meshes.

The fourth-order WENO was able to surpass the two popular schemes, in both triangular and square meshes. However, for the triangular meshes, the gain in accuracy is diminute as defficiencies in triangular cells search areas hinder the performance of the WENO scheme. The same pattern was observed in the triangular MUSCL scheme, which also depends on diretional stencils. For the square meshes, the gain in accuracy is very noticeable, with the MUSCL scheme surppassing the Godunov scheme and the fourth-order WENO surpassing both.

The metric $||e||$ of the error in density along the diagonal is given as a function of solver run-time (SRT) to compare computational efficiency between the three schemes (figure 4.29).

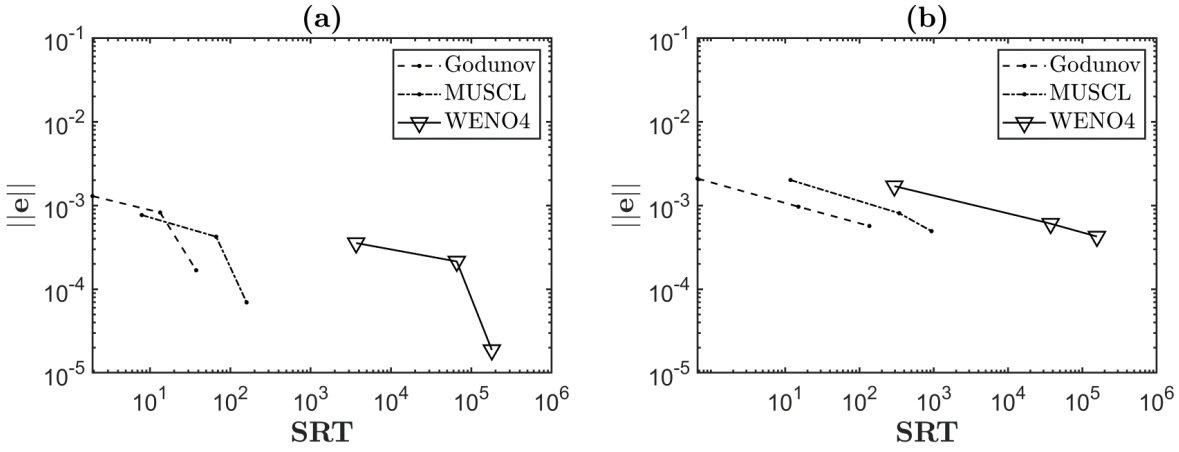

Figure 4.29: $||e||$ metric of the error in density along the diagonal as function of solver run-time (SRT) for (a) square and (b) triangular meshes.

For both triangular and square meshes, the fourth-order WENO provided the worse computational efficiency. For the square meshes, the difference between the required solver run-time for the same

range of error was slightly more accentuated. This fact is probably explained by the existence of an extra face and, consequently, directional stencil for each cell. The use of MATLAB ® programming platform, using just-in-time compilation, also does not help the already computationally strenuous WENO scheme, exarcebating the time to compute the extra processes that high-order schemes require. For the square meshes, the best computational efficiency was achieved by the MUSCL in the lowest level of refinement and by the Godunov in the remaing two. For the triangular meshes, the Godunov scheme provided the best computational efficiency. For both types of mesh, the WENO scheme did not show promiss of surpassing the two others schemes with higher levels of refinement.

The cell averaged density values of the cells adjacent to the diagonal, computed for the highest level of refinement, are shown as function of radial distance (r) to the origin of the domain in figure 4.30.

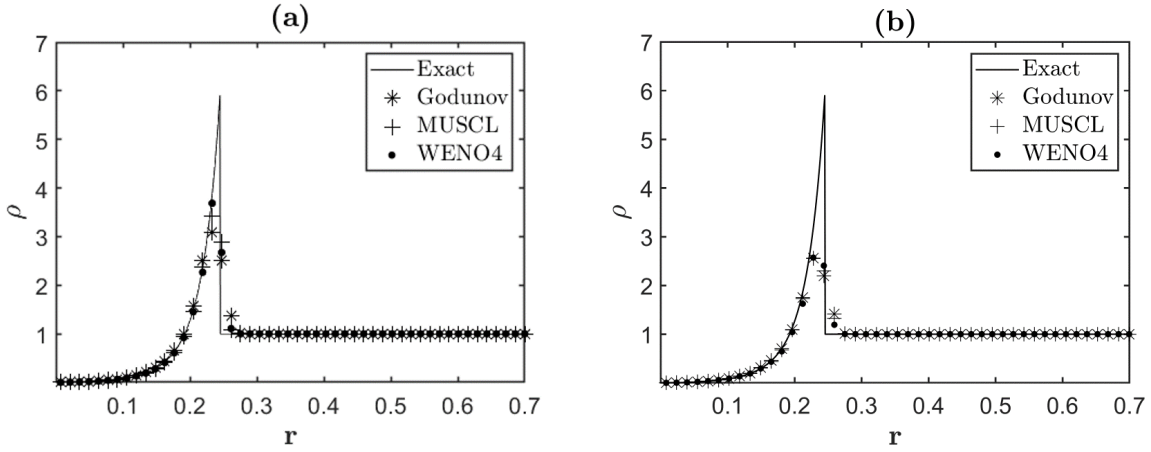

Figure 4.30: Density cell averaged values given as a function of radial distance to the origin (cells in the diagonal) for (a) square and (b) triangular meshes.

4.2.4 Supersonic Forward Facing Step

The Supersonic Forward-Facing Step test case is a very well documented standard test usefull for fail-proofing CFD code. The test consists of an inviscid supersonic flow (Mach=3) over a forward facing step. In this section, only regular cells will be used as triangular cells provided unsatisfatory results for the other test cases. The fourth-order WENO is again the only WENO scheme used. All the necessary information to implement this test case was taken from [15].

The necessary flow properties troughout all the domain are prescribed by equation 4.49, aswell as the inlet flow quantities, at the left boundary of the domain.

$$\gamma(x,y) = 1.4; \quad R(x,y) = 0.714; \quad p(0,y) = 1; \quad T(0,y) = 1; \quad u_x(0,y) = 3; \quad u_y(0,y) = 0 \tag{4.49}$$

The initial quantities inside the domain were the same as the inlet ones. The outlet received a supersonic outflow boundary condition (null Neumann boundary condition in x). The boundary conditions of every other boundary of the domain were all reflective (symmetry planes).

The developed code was left to iterate until it reached $t = 0.5s$ for the fourth-order WENO, Godunov

and MUSCL schemes. The time-step used was set by lowering the Courant-Friedrichs-Lewy number ($CFL = 0.4$) until sufficient visual difference between the results was achieved. The mesh used was a regular square mesh with considerable refinement ($\Delta x = \Delta y = \frac{1}{60}$). In figure 4.31, thirty density countours are shown for the Godunov, MUSCL and fourth-order WENO. The reference solution at $t = 0.5s$ was taken from [15].

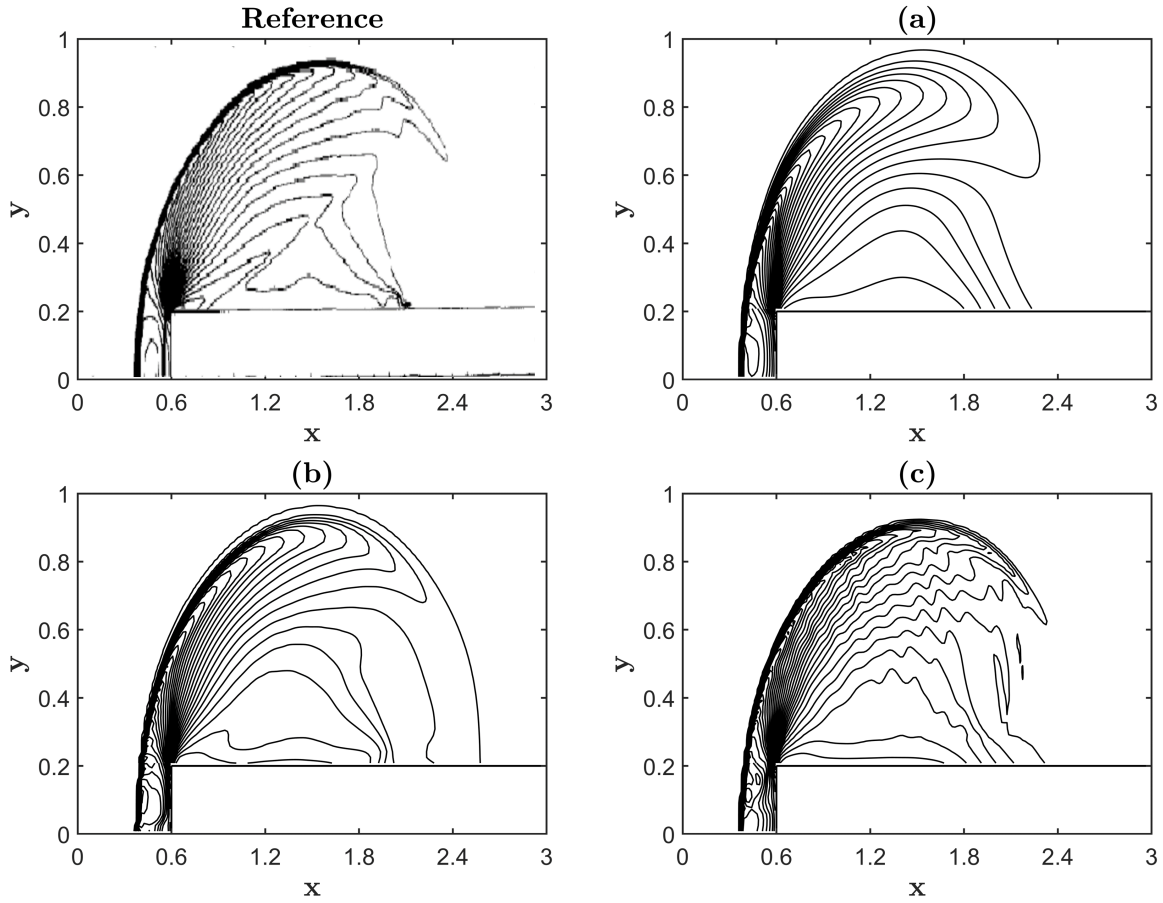

Figure 4.31: Density contours for the SFFS test case at $t = 0.5s$ for the reference solution (top left corncer), (a) Godunov, (b) MUSCL and (c) fourth-order WENO.

From the results on figure 4.31 it becomes apparent that the fourth-order WENO scheme offered better resemblance to the reference than the two other shemes. The thickness of the weak shock was very close to the one in the reference and the scheme succeeded in depicting the initial intersection of the contours related to the expansion fan, which forms on the left step corner, to the top part of the weak shock. These two major characteristic provided a flow structure very similar to that of the reference. The sharp twist in the density contours at the top right of the ”buble” was only captured by the WENO scheme and the wiggles on the density contours for the expansion region are far better depicted. By contrast, the Godunov and MUSCL schemes provided very smooth density contours, the thickness of the shock was depicted mediocrely and the interaction between the expansion fan and the shock was not as sharp. The MUSCL scheme appeared to provide better depiction of the flow strutures near the top face of the step.

Chapter 5

Conclusions

Altough the objective of bringing arbitrary high-order accuracy to discontinuity-ridden solutions was met, it was only accomplished through the use of characteristic variables. The computational expense of setting up a approximate characteristic equation for each face made the applicability of the scheme dwindle for discontinuity dominated solutions. WENO methodology was also found to be sensitive to the grid type as the use of triangular cells provided lackluster results. The behaviour of stencil search areas for triangular cells near discontinuities and shock contributed for these unimpressive results. The developed WENO schemes provided better results, accuracy-wise, than the popular Godunov and MUCL, for all simulated cases. They did, however, provide worse computational efficiencies. The analysis of the numerical results obtained suggests that the fourth-order WENO provided the best overall results, obtaining a better compromise between accuracy and computational time.

5.1 Achievements

In this thesis, the WENO methodology was successfully implemented for triangular and square meshes. The drawbacks and strenghts of the developed schemes were registred, with several numerical results to justify them. Simple flux limiting methodologies for one-dimensional and two-dimensional WENO schemes were successfully tested. Readable and user-friendly code was developed and organized to provide a solid starting platform for future work and developments on WENO schemes with Least-Squares. A considerable part of the theoretical background needed for successful implementation of WENO schemes was catalogued and described. The developed WENO schemes succeed in providing the desired order of accuracy in smooth solutions and better accuracy than the popular schemes, Godunov and MUSCL.

5.2 Future Work

The WENO schemes here implemented can be subjected to a myriad of improvements. At the top of the list should be reducing the computational time required. For this purpose, more efficient methods of

matrix invertion can be used, such as QR decomposition, and implementation in other lower-level coding languages explored. C language, for example, can decrease computational time by several orders of magnitude, especially when using paralell computing. Optimization of the code for a limited array of desired orders of accuracy should also be considered. Some implemented functions are designed to handle arbitrary orders of accuracy and ,thus, trade computational performance for flexibility.

The two-dimensional shemes can be extended to polihedral meshes and to the full Navier-Stokes equations.

In a future thesis, more advanced adaptative WENO methodology can be obtained. Recent developments in machine learning applied to CFD allow for detection of smooth regions and consequent decision-making on what order of accuracy WENO to use. Using this methodology avoids reversion to lower orders when all the required stencils intersect either discontinuities or shocks.

Better flux limiting strategies and alternatives to the use of characteristics variables can also be explored.

The methodology for the Riemann problems set on the faces can be reworked with machine learning. The complicated behaviour of the flow structures can be modeled providing more efficient solutions for these problems.

Bibliography

[1] C.-W. Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes. *Acta Numerica*, 29:701–762, 2020. doi: 10.1017/S0962492920000057.

[2] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, 1994. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1994.1187. URL https://www.sciencedirect.com/science/article/pii/S0021999184711879.

[3] A. G. Vasconcelos, D. M. Albuquerque, and J. C. Pereira. A very high-order finite volume method based on weighted least squares for elliptic operators on polyhedral unstructured grids. *Computers Fluids*, 181:383–402, 2019. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2019.02.004. URL https://www.sciencedirect.com/science/article/pii/S0045793019300295.

[4] P. Tsoutsanis, V. Titarev, and D. Drikakis. Weno schemes on arbitrary mixed-element unstructured meshes in three space dimensions. *Journal of Computational Physics*, 230(4):1585–1601, 2011. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2010.11.023. URL https://www.sciencedirect.com/science/article/pii/S0021999110006388.

[5] E. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*, pages 163–212. 03 2009. ISBN 978-3-540-25202-3. doi: 10.1007/b79761_5.

[6] H. Liu and X. Jiao. Wls-eno: Weighted-least-squares based essentially non-oscillatory schemes for finite volume methods on unstructured meshes. *Journal of Computational Physics*, 314:749–773, 2016. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2016.03.039. URL https://www.sciencedirect.com/science/article/pii/S0021999116001911.

[7] D. Pan. A high-order finite volume method for solving one-dimensional convection and diffusion equations. *Numerical Heat Transfer, Part B: Fundamentals*, 71:1–16, 06 2017. doi: 10.1080/10407790.2017.1326769.

[8] P. Tsoutsanis. Stencil selection algorithms for weno schemes on unstructured meshes. *Journal of Computational Physics: X*, 4:100037, 08 2019. doi: 10.1016/j.jcpx.2019.100037.

[9] J. E. Akin. *Finite element analysis with error estimators : an introduction to the FEM and adaptive error analysis for engineering students*. Elsevier/Butterworth-Heinemann, England, 2005. ISBN 0750667222.

[10] P. S. Farmakis, P. Tsoutsanis, and X. Nogueira. Weno schemes on unstructured meshes using a relaxed a posteriori mood limiting approach. *Computer Methods in Applied Mechanics and Engineering*, 363:112921, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.112921. URL https://www.sciencedirect.com/science/article/pii/S0045782520301043.

[11] A. Cheniguel. Numerical simulation of two-dimensional diffusion equation with non local boundary conditions. *International Mathematical Forum*, 01 2012.

[12] J. White, H. Nishikawa, and R. Baurle. Weighted least-squares cell-average gradient construction methods for the vulcan-cfd second-order accurate unstructured grid cell-centered finite-volume solver. 01 2019. doi: 10.2514/6.2019-0127.

[13] L. Sedov. *Similarity and Dimensional Methods in Mechanics*. 05 2018. ISBN 9780203739730. doi: 10.1201/9780203739730.

[14] Y. Xia, D. Andrs, and R. Martineau. A hermite weno-based slope limiter for cell-centered finite volume methods on unstructured meshes. 04 2017.

[15] P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics*, 54(1):115–173, 1984. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(84)90142-6. URL https://www.sciencedirect.com/science/article/pii/0021999184901426.

Appendix A

Secondary Results

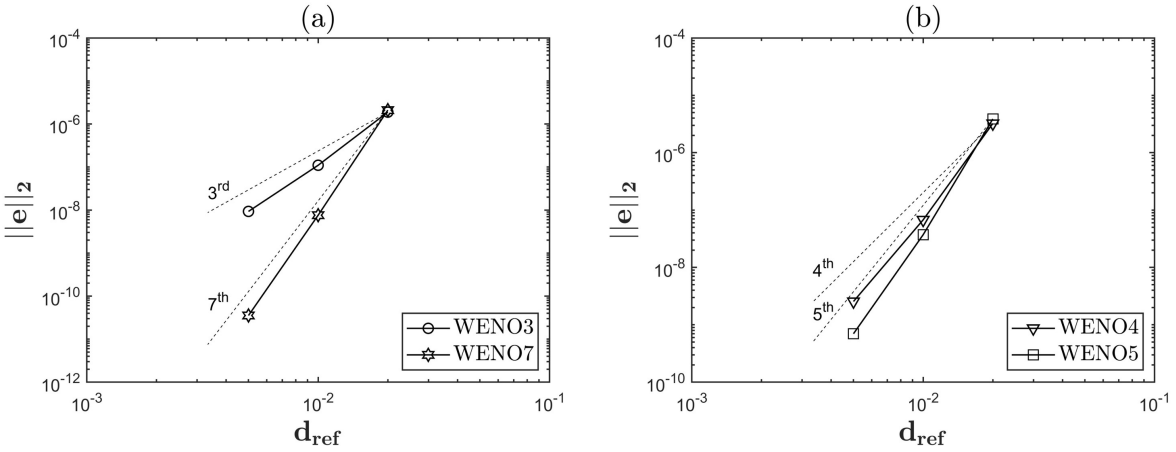

Figure A.1: Fluxes on the boundaries of the square mesh- (a) L_2 norm of the error versus the average reference size (d_{ref}) for the third-order and seventh-order CWENO schemes. (b) L_2 norm of the error versus the average reference size (d_{ref}) for the fourth-order and fifth-order CWENO schemes.

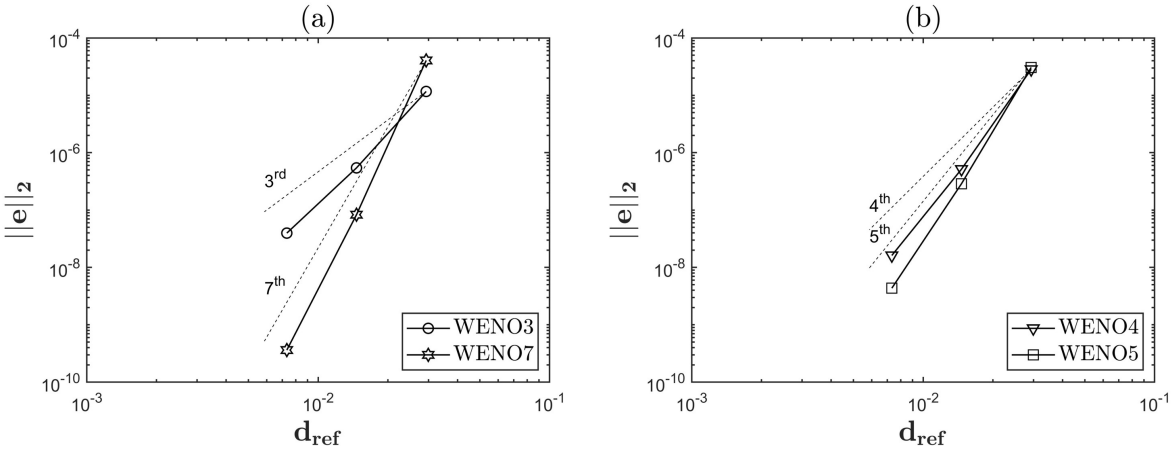

Figure A.2: Fluxes on the boundaries of the triangular mesh- (a) L_2 norm of the error versus the average reference size (d_{ref}) for the third-order and seventh-order CWENO schemes. (b) L_2 norm of the error versus the average reference size (d_{ref}) for the fourth-order and fifth-order CWENO schemes.

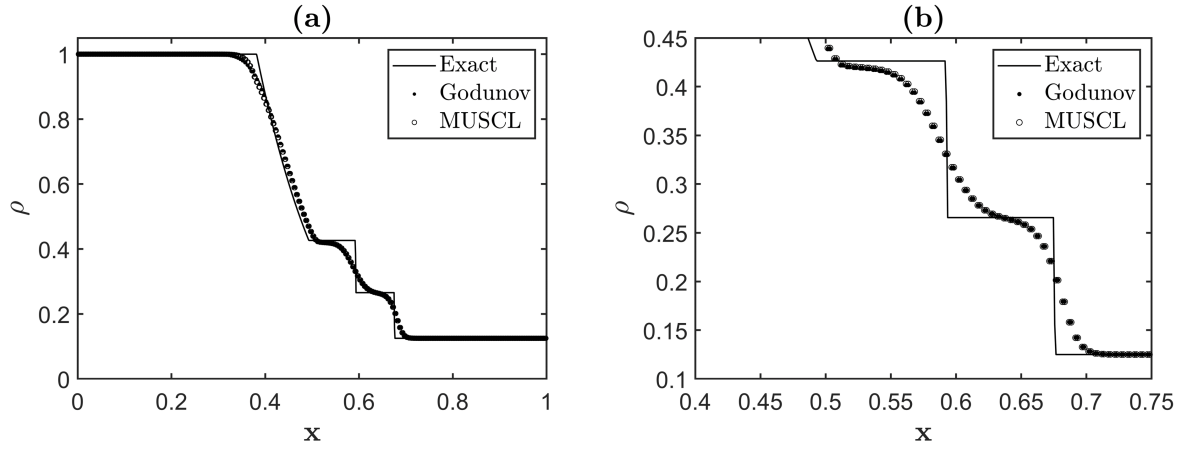

Figure A.3: Triangular meshes - Density as a function of the x-coordinate for the Godunov and MUSCL schemes at $t = 0.1s$.

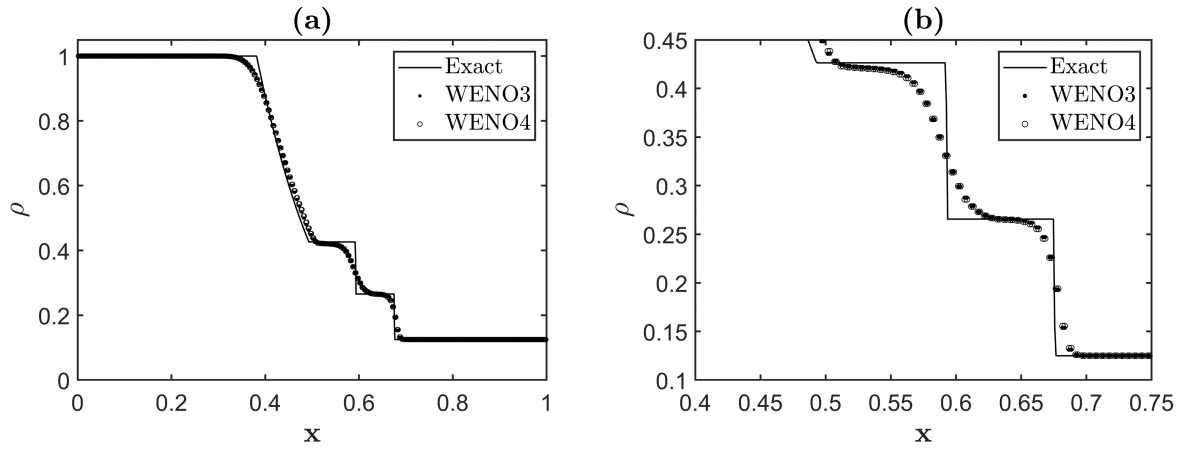

Figure A.4: Triangular meshes - Density as a function of the x-coordinate for the third and fourth-order WENO schemes at $t = 0.1s$.

Appendix B

MATLAB Code

Listing B.1: Main

```
clc, clear all

%% User Provided Inputs
global gamma
Solver=input('Choose solver!!! \n 0 for Godunov, 1 for MUSCL, 2 for WENO: ');
if Solver==2, r=input('Degree of interpolation: ');end
if Solver==1, r=1; end
gamma=input('Specific Heat Ratio: ');

%% Quadrilateral mesh
global p, global t, global faces
N=12; %level of refinement
%%first retangle
Lx=[0,3]; %the first entry is for the left of the rectangle, the last for the right
Ly=[0,1]; %the first entry is for the top of the rectangle, the last for the bottom
cell_side=(1/N)*0.2; Ny=1/cell_side; Nx=3*Ny; n_faces=4;
[p,t,faces]=rect_mesh(Lx,Ly,Nx,Ny,n_faces); %creating the mesh

%% Passing the retangular mesh to the step problem
[t,faces]=rect_to_step(p,t,faces); %mesh_ploter(p,t,faces,0)

%% Mesh information matrix
[A]=mesh_info(p,t,faces,n_faces,0); nx=size(A,2); %number of cells
int_cells=1:1:nx; %interior cells

if Solver==2
   %% Modified Taylor Basis Polynomials
   global Taylor_Basis
   [Taylor_Basis]=Basis_Functions(r);

   %% Stencils
   global Stencils, global fails
   S_size=2*(((r+1)*(r+2))/2)-2;  %size of desired stencil
```

```
    [Stencils,fails]=Complete_Stencils(A,S_size,p,t,int_cells);
    %stencil_checker(A,Stencils,fails,int_cells)

    %% P Matrices and Jacobians
    global P_Matrices, global Jacobians
    [P_Matrices,Jacobians]=All_P_Matrices(A,p,t,r,int_cells);
end

%Update dt and time
CFL=0.4; it=0;
time=0;  tFinal=0.5;

while time < tFinal
    A2=A; A3=A;

    if Solver==0, [L,radius,signal]=Godunov_Flux(A);
    elseif Solver==1, [L,radius,signal]=MUSCL_Flux(A);
    elseif Solver==2, [L,signal,radius]=CWENO_Flux(A,r);
    end

    %iteration local time
    dt=CFL*(cell_side)/(2*signal);
    if time+dt>tFinal, dt=tFinal-time; end, time=time+dt;

    for m=1:1:nx
        A(1,m)= A(1,m)-L(m,1)*dt*(1/A(7,m));
        A(2,m)= A(2,m)-L(m,2)*dt*(1/A(7,m));
        A(3,m)= A(3,m)-L(m,3)*dt*(1/A(7,m));
        A(4,m)= A(4,m)-L(m,4)*dt*(1/A(7,m));
    end

    %second stage
    if Solver==0, [L,radius,signal]=Godunov_Flux(A);
    elseif Solver==1, [L,radius,signal]=MUSCL_Flux(A);
    elseif Solver==2, [L,signal,radius]=CWENO_Flux(A,r);
    end

    for m=1:1:nx
        A2(1,m)= (3/4)*A(1,m)+(1/4)*A1(1,m)-(1/4)*L(m,1)*(dt/cell_side);
        A2(2,m)= (3/4)*A(2,m)+(1/4)*A1(2,m)-(1/4)*L(m,2)*(dt/cell_side);
        A2(4,m)= (3/4)*A(4,m)+(1/4)*A1(4,m)-(1/4)*L(m,4)*(dt/cell_side);
    end

    %third stage
    if Solver==0, [L,radius,signal]=Godunov_Flux(A);
    elseif Solver==1, [L,radius,signal]=MUSCL_Flux(A);
    elseif Solver==2, [L,signal,radius]=CWENO_Flux(A,r);
    end
```

```
    for m=1:1:nx
        A(1,m)= (1/3)*A(1,m)+(2/3)*A2(1,m)-(2/3)*L(m,1)*(dt/cell_side);
        A(2,m)= (1/3)*A(2,m)+(2/3)*A2(2,m)-(2/3)*L(m,2)*(dt/cell_side);
        A(4,m)= (1/3)*A(4,m)+(2/3)*A2(4,m)-(2/3)*L(m,4)*(dt/cell_side);
    end

    %Plotting the residual of density
    residual=sum(abs(L(:,1)))/nx; loglog(time,residual,'kx'), hold on
    drawnow

    %Update iteration counter
    it=it+1;
end

%% Plotting results
[~,~,~]=Q_plotter(A,cell_side,1); %plotting density contours
```

Listing B.2: Basis Functions

```
function [Binomials]=Basis_Functions(r)
%This function calculates the basis functions for the transformed space
num=(((r+1)*(r+2))/2);
Binomials=zeros(num,3); %first row has the power(n) for x (x^n)
                        %second row has the power(m) for y (y^m)
                        %third row has the integration value for the
                        %standard equilateral unit triangle.

%% Creating the standard binomials (x^n)*(y^m)

i=1; %Number of Binomials already created
n=0; %Initialization of power of x
while i<=num
 for m=0:1:r-n, Binomials(i,1)=n; Binomials(i,2)=m; i=i+1; end, n=n+1;
end

%Removing the first term of the binomials (=1)
Binomials(1,:)=[];

%% Integrating the binomials in the standard unit square
for j=1:1:size(Binomials,1)
   Binomials(j,3)=(1/(Binomials(j,2)+1)) * (1/(Binomials(j,1)+1)) *...
   (0.5^(Binomials(j,1)+1) - (-0.5)^(Binomials(j,1)+1)) * (0.5^(Binomials(j,2)+1)...
  - (-0.5)^(Binomials(j,2)+1));  Binomials(j,3)=-Binomials(j,3);
end

end
```

Listing B.3: Stencils

```
function [Stencils,fail] = stencil(cell,A,S_size,p,t)
```

```
% cell : target cell for which the stencil is being computed
% A : mesh information and cell values
% r : order of the interpolation
% p : vertices/nodes coordinates
% t : cell vertices/nodes

n_faces=size(t,2);
%calculation of the search area straights and respective quadrants
[angles,~]=Quadrants_and_slopes(cell,A,p,t);

%initialization of the four directional stencils
Stencils = zeros(4,S_size);

%Stencil size count
count = zeros(1,4);

%Stencil fail (respective entry turns to 1 if the stencil has failed
%to be completed, entry 1 is for a12, entry 2 for a13 and 3 for a23)
fail=zeros(1,4);

%Stencil completeness (respective entry turns to 1 if the stencil has
%been completed, entry 1 is for a12, entry 2 for a13 and 3 for a23)
full=zeros(1,4);

%Initialization
cand=cell;
iter=1;

while (sum(fail==1) + sum(full==1))<n_faces %Until all directional stencils have failed or are completed
      %From the previous cells we extract new candidates
       candidates= zeros(1,4*size(cand(iter,:),2));%a previous candidate
                                                   %has on the very max
                                                   %three new neighbours
        j=0; aux_cand=cand(iter,:); %Removing zero values
        aux_cand(aux_cand==0)=[];
        for i=1:1:size(aux_cand,2)
            aux = A(8,aux_cand(1,i));
            if sum(sum(cand==aux))==0 && aux~=0 && sum(sum(candidates==aux))==0
            j=j+1;
            candidates(1,j)=aux;
            end

            aux = A(24,aux_cand(1,i));
            if sum(sum(cand==aux))==0 && aux~=0 && sum(sum(candidates==aux))==0
            j=j+1;
            candidates(1,j)=aux;
            end

            aux = A(40,aux_cand(1,i));
```

```
        if sum(sum(cand==aux))==0 && aux~=0 && sum(sum(candidates==aux))==0
        j=j+1;
        candidates(1,j)=aux;
        end

        aux= A(56,aux_cand(1,i));
        if sum(sum(cand==aux))==0 && aux~=0 && sum(sum(candidates==aux))==0
        j=j+1;
        candidates(1,j)=aux;
        end
    end

 %Removing the allocated space with zero value
 candidates(candidates==0)=[];

%Auxilliaries to detect stencil failure
 prev_a12=count(1);
 prev_a13=count(2);
 prev_a23=count(3);
 prev_a34=count(4);

%Evaluating in which search area the candidate cell belongs
for m=1:1:size(candidates,2)
    [S_search,~] = search_area(cell,A,angles,candidates(1,m));

    if S_search==1 && count(1)<S_size  && fail(1,1)==0
        count(1)=count(1)+1;
        Stencils(1,count(1))=candidates(1,m);
    elseif S_search==2 && count(2)<S_size && fail(1,2)==0
        count(2)=count(2)+1;
        Stencils(2,count(2))=candidates(1,m);
    elseif S_search==3 && count(3)<S_size && fail(1,3)==0
        count(3)=count(3)+1;
        Stencils(3,count(3))=candidates(1,m);
    elseif S_search==4 && count(4)<S_size && fail(1,4)==0
        count(4)=count(4)+1;
        Stencils(4,count(4))=candidates(1,m);
    end
end

 %Checking for stencils failures
 if count(1)==prev_a12 && count(1)<S_size
     fail(1,1)=1;
 elseif count(1)==S_size
     full(1,1)=1;
 end

 if count(2)==prev_a13 && count(2)<S_size
     fail(1,2)=1;
```

```
    elseif count(2)==S_size
        full(1,2)=1;
    end

    if count(3)==prev_a23 && count(3)<S_size
        fail(1,3)=1;
    elseif count(3)==S_size
        full(1,3)=1;
    end

    if count(4)==prev_a34 && count(4)<S_size
        fail(1,4)=1;
    elseif count(4)==S_size
        full(1,4)=1;
    end



    %Filling in the cand matrix with all the cells analyzed to compare if future
    %cells aren t already there
    if isempty(candidates)==0
       if j>size(cand,2)
          cand = [cand zeros(iter,j-size(cand,2)); candidates];
       elseif j== size(cand,2)
          cand = [cand; candidates];
       elseif j<size(cand,2)
          candidates=[candidates zeros(1,size(cand,2)-j)];
          cand =[cand; candidates];
       end
    else
        cand = [cand; zeros(1,size(cand,2))];
    end

    %iteration counter
    iter=iter+1;
end

 %% Discarting failed stencils
for i=1:1:size(Stencils,1)
    if fail(i)==1, Stencils(i,:)=0;end
end

end
```

Listing B.4: P Matrices

```
function [P]=P_matrix(A,J,cell,S_num,r)
%This function returns a P matrix for a specific stencil of given target cell.
%cell: identifies the target cell
%S_num: identifies the stencil of the specified 'cell'.
```

```
%'A': the mesh information matrix.
%'p': contains the information of the nodes of the mesh
%'t': contains the nodes belonging to each cell.
%'J': Jacobian of the mapping for the target cell.

global Stencils      %Stencils
global Taylor_Basis  %Modified Taylor Polynomial Basis Functions
global faces, global p, global t

%% Gauss Points and Weights (for an unitary retangle triangle)
Gauss_Points=[ 0 0; 0.5 0; 1 0; 0.5 0.5; 0 1; 0 0.5; 1/3 1/3];
Gauss_Weights=[1/40 1/15 1/40 1/15 1/40 1/15 9/40];

%% Obtaining the desired stencil
Stencil=Stencils(cell,S_num,:);

%% Transforming stencil coordinates into auxiliary space
IJ=inv(J); xc=A(5,cell); yc=A(6,cell); Scoord=zeros(2,size(Stencil,3));
for m=1:1:size(Scoord,2)
    aux=[A(5,Stencil(m));A(6,Stencil(m))];
    Scoord(:,m)=(IJ)*(aux-[xc;yc]);
end

%% Transforming p onto auxiliary space
Tp=p;
Tp(:,1)=IJ(1,1)*(p(:,1)-xc)+IJ(1,2)*(p(:,2)-yc);
Tp(:,2)=IJ(2,1)*(p(:,1)-xc)+IJ(2,2)*(p(:,2)-yc);

%% Calculating the P Matrix
% Computing the D^{LS} matrix
S_size=size(Stencils,3);
n_coeffs=size(Taylor_Basis,1);
D=zeros(S_size,n_coeffs); %the number of rows is equal to the size of
                          %the stencil and the number of columns equal to
                          %the number of basis functions in the polynomial
                          %model

n_faces=size(t,2);
for m=1:1:S_size %going throught the rows of the D^{LS} matrix
    Scell=Stencil(m); Sarea=0;
    if n_faces==3, triangles=zeros(1,6); %triangle ---> one triangle
    elseif n_faces==4,  triangles=zeros(2,6); %square ---> two triangles
    elseif n_faces>4, triangles=zeros(n_faces,6); %higher than square
    end

    if size(triangles,1)==1 %Decomposing element into a particular number of triangles
        Point1=t(Scell,1); Point2=t(Scell,2); Point3=t(SCell,3);
        triangles(1,1:3)=[Tp(Point1,1),Tp(Point2,1), Tp(Point3,1)]; %x-coord of the nodes
        triangles(2,4:6)=[Tp(Point1,2),Tp(Point2,2), Tp(Point3,2)]; %y-coord of the nodes
```

```
    elseif size(triangles,1)==2 %beware of the orientation of the nodes for square cells!!!
        Point1=t(Scell,1); Point2=t(Scell,2); Point3=t(SCell,3); Point4=t(SCell,4);
        %first triangle
        triangles(1,1:3)=[Tp(Point1,1),Tp(Point2,1), Tp(Point4,1)]; %x-coord of the nodes
        triangles(1,4:6)=[Tp(Point1,2),Tp(Point2,2), Tp(Point4,2)]; %y-coord of the nodes
        %second triangle
        triangles(2,1:3)=[Tp(Point1,1),Tp(Point3,1), Tp(Point4,1)]; %x-coord of the nodes
        triangles(2,4:6)=[Tp(Point1,2),Tp(Point3,2), Tp(Point4,2)]; %y-coord of the nodes
    elseif size(triangles,1)==n_faces
        for j=1:1:n_faces %Decomposing cell into "n_faces" number of triangles
            Point1=faces(Scell,j,1); Point2=faces(Scell,j,2);
            triangles(j,1:3)=[Scoord(1,m), Tp(Point1,1), Tp(Point2,1)];
            triangles(j,1:3)=[Scoord(2,m), Tp(Point1,2), Tp(Point2,2)];
        end
    end

    for j=1:1:size(triangles,1) %sum of the contribution of the triangles
        x1=triangles(j,1); x2=triangles(j,2); x3=triangles(j,3);
        y1=triangles(j,4); y2=triangles(j,5); y3=triangles(j,6);
        %Passing the integration points to the considered triangle
        SJ=[x2-x1 x3-x1; y2-y1 y3-y1];
        TGauss=Gauss_Points;
        TGauss(:,1)=Gauss_Points(:,1)*SJ(1,1)+Gauss_Points(:,2)*SJ(1,2)+x1;
        TGauss(:,2)=Gauss_Points(:,1)*SJ(2,1)+Gauss_Points(:,2)*SJ(2,2)+y1;
        %Checking cells and integration points
        %intpoints_checker(triangles,TGauss,j)
        %Calculating area of the triangle
        [TArea]=Sarea_calculator(triangles,j); Sarea=Sarea+TArea;

        for n=1:1:size(Taylor_Basis,1) %going trought the columns
            for k=1:1:size(Gauss_Weights,2) %going trought the integration points
                D(m,n)=D(m,n)+(TArea/(0.5))*Gauss_Weights(k)*(TGauss(k,1)^Taylor_Basis(n,1)*...
                    TGauss(k,2)^Taylor_Basis(n,2)+Taylor_Basis(n,3));
            end
        end
    end
    D(m,:)=D(m,:)./Sarea; %Passing from integration value to average cell value
end

%Computing the W diagonal matrix (weights matrix)
W=zeros(1,size(Stencil,3));
for m=1:1:size(Stencil,3), W(1,m)=1/(sqrt(Scoord(1,m)^2+Scoord(2,m)^2))^r; end
W=diag(W);
%Computing the Dwf matrix
Dwf=W*D;
%Computing the P Matrix
P=Cond_Numb_inverse(transpose(Dwf)*D)*transpose(Dwf);
end
```

Listing B.5: WENO Flux

```
function [res,fastest_speed,lowest_radius]=CWENO_Flux(A,r)
global p, global t
global Taylor_Basis, global Jacobians, global gamma
nx=size(A,2); n_faces=size(t,2);
% Initizalization of Flux storage
MF = zeros(nx,n_faces); XMF = zeros(nx,n_faces);
YMF = zeros(nx,n_faces); EF= zeros(nx,n_faces);
Completed = zeros(nx,n_faces); %1 indicates that the flux for that
                               %cell face has already been calculated
res=zeros(nx,4); %for 2D Euler

%Variables for time-step computation
fastest_speed=0;
[~,~,lowest_radius]=norm_vector(1,1,A,p);

%Gauss Legendre points [0,1] for polynomials:
%Up to 7th order of accuracy
GLQ_points= [0.5 + (1/2)*(sqrt(525-70*sqrt(30))/35),...
             0.5 - (1/2)*(sqrt(525-70*sqrt(30))/35), ...
             0.5 + (1/2)*(sqrt(525+70*sqrt(30))/35),...
             0.5 - (1/2)*(sqrt(525+70*sqrt(30))/35)];
GLQ_Weights=0.5*[(18+sqrt(30.0))/(36), (18+sqrt(30.0))/(36),...
                 (18-sqrt(30.0))/(36), (18-sqrt(30.0))/(36)];

%Up to fourth order of accuracy
%GLQ_points=0.5*[-1/sqrt(3), 1/sqrt(3)]+0.5;
%GLQ_Weights=0.5*[1 1];


%Tansformed square face parametrization(Xn=an+bn*s; Y=cn+dn*s;)
if n_faces==3, [XFace,YFace,pj]=tri_faces; end
if n_faces==4, [XFace,YFace,pj]=square_faces; end

for cell=1:1:nx %Cycling trough the cells
    for n=1:1:n_faces %Cycling trough the faces

        %Discovering in what neighbour position the current cell(m) is on
        %the current neighbour(n) (n o detetei erro)
        neighbour=A(8+(n-1)*16,cell);
        if neighbour~=0
        for i=1:1:n_faces,if A(8+(i-1)*16,neighbour)==cell,nn= i; end, end
        end

        %Calculating normal, tangent vectors and cell radius
        [normal,tangent,aux_radius]=norm_vector(cell,n,A,p);
        if aux_radius<lowest_radius, lowest_radius=aux_radius; end
        if neighbour~=0,[nnormal,ntangent,~]=norm_vector(neighbour,nn,A,p);end
```

```
if neighbour~=0 && Completed(cell,n)==0 %Checking if the face has a neighbour
   %Characteristic Variables Approximation
   Coeffs=CWENO(A,p,cell,n,r); %Current cell polynomial model
   NCoeffs=CWENO(A,p,neighbour,nn,r); %Current neighbour polynomial model

   %Face identifier for current cell (1 for face 1, 2 for face
   %2,etc... ) (n o detetei erro)
   tface=face_identifier(A,t,cell,n);

   %cell centers for target and neighbouring cells
   xc=A(5,cell); yc=A(6,cell);
   nxc=A(5,neighbour); nyc=A(6,neighbour);

   %Jacobians for the target and neighbouring cells
   NJ=zeros(2,2); J=zeros(2,2);
   for i=1:1:2,for j=1:1:2,J(i,j)=Jacobians(cell,i,j);...
               NJ(i,j)=Jacobians(neighbour,i,j);end, end

   %Riemman Problem in the Gauss integration points
    Gauss_Fluxes=zeros(size(GLQ_points,2),4);
    for i=1:1:size(GLQ_points,2)
        %Gauss points in the current cell transformed space
        epsilon=XFace(tface,1)+XFace(tface,2)*GLQ_points(1,i);
        eta=YFace(tface,1)+YFace(tface,2)*GLQ_points(1,i);

        %Gauss points in the neighbour transformed space
        tran=NJ\([xc;yc]-[nxc;nyc]+J*[epsilon;eta]);
        nepsilon=tran(1); neta=tran(2);

        Left=[0; 0; 0; 0];
        Right=[0; 0; 0; 0];
        for j=1:1:size(Taylor_Basis,1)
            for h=1:1:4
                Left(h)=Left(h)+Coeffs(h,j)*((epsilon^Taylor_Basis(j,1))*...
                        (eta^Taylor_Basis(j,2)) +Taylor_Basis(j,3));
                Right(h)=Right(h)+NCoeffs(h,j)*((nepsilon^Taylor_Basis(j,1))*...
                        (neta^Taylor_Basis(j,2)) +Taylor_Basis(j,3));
            end
        end

        %Adding the respective cell averages to the polynomials
        Left=Left+[A(1,cell);A(2,cell)*normal(1)+A(3,cell)*normal(2);...
             A(2,cell)*tangent(1)+A(3,cell)*tangent(2);A(4,cell)];
        Right=Right+[A(1,neighbour);A(2,neighbour)*nnormal(1)+A(3,neighbour)*nnormal(2);...
                    A(2,neighbour)*ntangent(1)+A(3,neighbour)*ntangent(2);A(4,neighbour)];
        %Converting from neighbour face axis to cell face axis
        %Normal momentum
        Right(2)=Right(2)*(normal(1)*nnormal(1)+normal(2)*nnormal(2));
```

```
        %Tangent momentum
        Right(3)=Right(3)*(tangent(1)*ntangent(1)+tangent(2)*ntangent(2));

        %Converting them into normal Euler properties
        %(from normal and tangent diretions to x and y)
        alpha=auxtan(normal(1),normal(2)); beta=auxtan(tangent(1),tangent(2));

        aux_Left=Left; Left(2)=cos(alpha)*aux_Left(2)+cos(beta)*aux_Left(3);
                       Left(3)=sin(alpha)*aux_Left(2)+sin(beta)*aux_Left(3);
        aux_Right=Right; Right(2)=cos(alpha)*aux_Right(2)+cos(beta)*aux_Right(3);
                         Right(3)=sin(alpha)*aux_Right(2)+sin(beta)*aux_Right(3);

        %Riemann Solver in the Gauss Points
        sw=Limiter_Switch(A,Left,Right,cell,neighbour);
        if sw==1
            Left=[A(1,cell);A(2,cell);A(3,cell);A(4,cell)];...
            Right=[A(1,neighbour);A(2,neighbour);A(3,neighbour);A(4,neighbour)];
            counter=counter+1;
        end
        [Gauss_Fluxes(i,:),aux_speed]=HLLCflux(Left,Right,normal,cell);

        if aux_speed>fastest_speed,fastest_speed=aux_speed;end
    end

    %Integrating fluxes in the transformed space
    %(pj is the jacobian of the face parametrization)
    Fluxes=zeros(4,1);
    Lface=A(11+(n-1)*16,cell); %lenght of face in original space
    TLface=1; %lenght of face in auxilliary coordinate system
    for i=1:1:size(Gauss_Fluxes)
        Fluxes(1)=Fluxes(1)+(Lface/TLface)*GLQ_Weights(i)*Gauss_Fluxes(i,1)*pj;
        Fluxes(2)=Fluxes(2)+(Lface/TLface)*GLQ_Weights(i)*Gauss_Fluxes(i,2)*pj;
        Fluxes(3)=Fluxes(3)+(Lface/TLface)*GLQ_Weights(i)*Gauss_Fluxes(i,3)*pj;
        Fluxes(4)=Fluxes(4)+(Lface/TLface)*GLQ_Weights(i)*Gauss_Fluxes(i,4)*pj;
    end

    %Filling in the fluxes(of the cell and the neighbour)
    MF(cell,n)=Fluxes(1);  XMF(cell,n)=Fluxes(2);
    YMF(cell,n)=Fluxes(3); EF(cell,n)=Fluxes(4);
    Completed(cell,n)=1;

    MF(neighbour,nn)=-Fluxes(1);  XMF(neighbour,nn)=-Fluxes(2);
    YMF(neighbour,nn)=-Fluxes(3); EF(neighbour,nn)=-Fluxes(4);
    Completed(neighbour,nn)=1;

elseif Completed(cell,n)==0
    %% Boundary Conditions
    Point1=A(9+(n-1)*16,cell);  Point2=A(10+(n-1)*16,cell);
    %left side of domain (INFLOW)
```

```
            if p(Point1,1)==p(Point2,1) && p(Point1,1)==0
                qL=[A(1,cell);A(2,cell);A(3,cell);A(4,cell)];
                qR=[1/(0.714286*1);(1/(0.714286*1))*3;0;(1/(gamma-1)) + 0.5*(1/(0.714286*1))*(3^2)];
            end

            %right side of the domain (OUTFLOW)
            if p(Point1,1)==p(Point2,1) && p(Point1,1)==3
                qL=[A(1,cell);A(2,cell);A(3,cell);A(4,cell)];
                qR=[A(1,cell);A(2,cell);A(3,cell);A(4,cell)];
                %qR=[1/(0.714*1);3;0;(1/(gamma-1)) + 0.5*(1/(0.714*1))*(3^2)];
            end

            %forward facing face of the step
            if p(Point1,1)==p(Point2,1) && p(Point1,1)==0.6
                qL=[A(1,cell);A(2,cell);A(3,cell);A(4,cell)];
                qR=[A(1,cell);-A(2,cell);A(3,cell);A(4,cell)];
            end

            %top of the domain
            if p(Point1,2)==p(Point2,2) && p(Point1,2)==1
                qL=[A(1,cell);A(2,cell);A(3,cell);A(4,cell)];
                qR=[A(1,cell);A(2,cell);-A(3,cell);A(4,cell)];
            end

            %bottom of domain
            if p(Point1,2)==p(Point2,2) && p(Point1,2)==0
                qL=[A(1,cell);A(2,cell);A(3,cell);A(4,cell)];
                qR=[A(1,cell);A(2,cell);-A(3,cell);A(4,cell)];
            end

            %horizontal wall of step
            if p(Point1,2)==p(Point2,2) && p(Point1,2)==0.2
                qL=[A(1,cell);A(2,cell);A(3,cell);A(4,cell)];
                qR=[A(1,cell);A(2,cell);-A(3,cell);A(4,cell)];
            end

            face_lenght=A(11+(n-1)*16,cell);
            [Flux,~]=HLLCflux(qL,qR,normal);
            Flux=Flux*face_lenght;
            MF(cell,n)=Flux(1);  XMF(cell,n)=Flux(2);
            YMF(cell,n)=Flux(3); EF(cell,n)=Flux(4);
            Completed(cell,n)=1;
        end
    end

end

for i=1:1:nx
    res(i,1)=sum(MF(i,:));
```

```
    res(i,2)=sum(XMF(i,:));
    res(i,3)=sum(YMF(i,:));
    res(i,4)=sum(EF(i,:));
end

end
```