

**ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY**

**NUMERICAL ANALYSIS OF FLOWS ABOUT AIRFOILS  
USING WENO SCHEMES**

**M.Sc. Thesis by  
Saygın AYABAKAN, B.Sc.**

**Department : Aeronautics and Astronautics Engineering**

**Programme : Aeronautics and Astronautics Engineering**

**FEBRUARY 2007**

**NUMERICAL ANALYSIS OF FLOWS ABOUT AIRFOILS  
USING WENO SCHEMES**

**M.Sc. Thesis by  
Saygın AYABAKAN, B.Sc.  
511031025**

**Date of submission : 25 December 2006**

**Date of defence examination : 1 February 2007**

**Supervisor (Chairman) : Prof. Dr. Alim Rüstem ASLAN**

**Members of the Examining Committee : Prof. Dr. Kadir KIRKKÖPRÜ**

**Assoc. Prof. Dr. Fırat Oğuz EDİS**

**FEBRUARY 2007**

**KANAT PROFİLLERİ ETRAFINDAKİ AKIŞIN  
WENO ŞEMALARI İLE SAYISAL ÇÖZÜMÜ**

**Yüksek Lisans Tezi  
Müh. Saygın AYABAKAN  
511031025**

**Tezin Enstitüye Verildiği Tarih : 25 Aralık 2006**

**Tezin Savunulduğu Tarih : 1 Şubat 2007**

**Tez Danışmanı : Prof. Dr. Alim Rüstem ASLAN**

**Diğer Jüri Üyeleri : Prof. Dr. Kadir KIRKKÖPRÜ**

**Doç. Dr. Fırat Oğuz EDİS**

**ŞUBAT 2007**

## **PREFACE**

In this thesis project, an unsteady Euler code generated and used with WENO scheme in order to calculate fluid flow about airfoil cross sections for finite volume decompositions. Finite difference decomposition is also used for some other test cases. Runge-Kutta method is used for time stepping.

I would like to express my gratitude to my supervisor Prof. Dr. A. Rüstem ASLAN for his guidance, encouragement and support during this thesis project.

In addition to that, I would like to express special thanks to whom gave me help during this time.

December 2006

Saygın AYABAKAN

## TABLE OF CONTENTS

|                                                                                                            |              |
|------------------------------------------------------------------------------------------------------------|--------------|
| <b>ABBREVIATIONS</b>                                                                                       | <b>vi</b>    |
| <b>TABLE LIST</b>                                                                                          | <b>vii</b>   |
| <b>FIGURE LIST</b>                                                                                         | <b>viii</b>  |
| <b>SYMBOL LIST</b>                                                                                         | <b>xiv</b>   |
| <b>SUMMARY</b>                                                                                             | <b>xvi</b>   |
| <b>ÖZET</b>                                                                                                | <b>xviii</b> |
| <br>                                                                                                       |              |
| <b>1. INTRODUCTION</b>                                                                                     | <b>1</b>     |
| <br>                                                                                                       |              |
| <b>2. GOVERNING EQUATIONS AND POLYNOMIAL APPROXIMATION</b>                                                 | <b>4</b>     |
| 2.1 Governing Equations for 1D (Euler Equations)                                                           | 4            |
| 2.1.1 Characteristic Form of Euler Equations                                                               | 7            |
| 2.1.2 Roe Averaging Procedure for Euler Equations                                                          | 9            |
| 2.2 Extensions to Second Dimension                                                                         | 12           |
| 2.3 One Dimensional Polynomial Approximation and Reconstruction                                            | 14           |
| 2.3.1 Lagrange Form Polynomial                                                                             | 15           |
| 2.3.2 Newton Form Polynomial                                                                               | 16           |
| <br>                                                                                                       |              |
| <b>3. ONE DIMENSIONAL APPROXIMATION AND RECONSTRUCTION</b>                                                 | <b>18</b>    |
| 3.1 Constructing the Grid                                                                                  | 18           |
| 3.2 One Dimensional Reconstruction from Cell Averages (Finite Volume)                                      | 18           |
| 3.3 One Dimensional Conservative Approximation to the Derivatives<br>From Point Values (Finite Difference) | 25           |
| <br>                                                                                                       |              |
| <b>4. ONE DIMENSIONAL ENO and WENO SCHEMES</b>                                                             | <b>28</b>    |
| 4.1 ENO Approximation                                                                                      | 29           |
| 4.2 WENO Approximation and Reconstruction                                                                  | 31           |
| 4.2.1 Procedure 4.1 - 1D WENO Reconstruction                                                               | 37           |
| 4.3 WENO Scheme for One Dimensional Conservation Laws                                                      | 38           |
| 4.3.1 Finite Volume Formulation in the Scalar Case                                                         | 39           |
| 4.3.2 Finite Difference Formulation in the Scalar Case                                                     | 40           |
| 4.3.3 WENO in Systems with Component-wise and Characteristic-wise<br>Decompositions                        | 41           |
| 4.3.4 Procedure 4.2 - FV 1D Scalar WENO Scheme                                                             | 43           |
| 4.3.5 Procedure 4.3 - FD 1D Scalar Flux Splitting WENO Scheme                                              | 44           |
| 4.3.6 Procedure 4.4 - Component-wise FV 1D Vectorial WENO Scheme                                           | 44           |
| 4.3.7 Procedure 4.5 - Component-wise FD 1D Vectorial WENO Scheme                                           | 45           |
| 4.3.8 Procedure 4.6 - Characteristic-wise FV 1D Vectorial WENO Scheme                                      | 45           |

|                                                                       |           |
|-----------------------------------------------------------------------|-----------|
| 4.3.9 Procedure 4.7 - Characteristic-wise FD 1D Vectorial WENO Scheme | 45        |
| 4.4 Time Discretization for ENO and WENO Schemes                      | 46        |
| 4.4.1 TVD Runge-Kutta Methods                                         | 47        |
| <b>5. TWO DIMENSIONAL APPROXIMATION AND RECONSTRUCTION</b>            | <b>49</b> |
| 5.1 WENO Scheme for Two Dimensional Conservation Laws                 | 50        |
| 5.1.1 Finite Volume Formulation in the Scalar Case                    | 50        |
| 5.1.2 Finite Difference Formulation in the Scalar Case                | 52        |
| 5.1.3 WENO in Systems                                                 | 53        |
| 5.1.4 Procedure 5.1 - FV 2D WENO Scheme                               | 53        |
| 5.1.5 Procedure 5.2 - FD 2D WENO Scheme                               | 54        |
| 5.2 Gaussian Integration Point Rule for WENO Scheme                   | 54        |
| 5.3 Piece-wise Parabolic WENO Reconstruction ( $5^{th}$ Order)        | 55        |
| <b>6. WENO SCHEME FOR NON-UNIFORM GRIDS</b>                           | <b>58</b> |
| 6.1 Piece-wise Parabolic WENO for Non-Uniform Grids ( $5^{th}$ Order) | 59        |
| 6.2 Piece-wise Linear WENO for Non-Uniform Grids ( $3^{rd}$ Order)    | 65        |
| <b>7. BOUNDARY CONDITIONS</b>                                         | <b>67</b> |
| 7.1 Ghost (Dummy) Cells Approach                                      | 67        |
| 7.2 $0^{th}$ Order Extrapolation Boundary Condition                   | 69        |
| 7.3 Linear Extrapolation Boundary Condition                           | 69        |
| 7.4 Periodic/Cut Boundary Condition                                   | 70        |
| 7.5 Farfield Boundary Condition                                       | 70        |
| 7.5.1 Subsonic Inflow                                                 | 71        |
| 7.5.2 Subsonic Outflow                                                | 72        |
| 7.5.3 Supersonic Inflow                                               | 72        |
| 7.5.4 Supersonic Outflow                                              | 73        |
| 7.6 Wall (Solid) Boundary Condition                                   | 73        |
| 7.7 Multiblock Interface Boundary Condition                           | 74        |
| <b>8. COORDINATE TRANSFORMATIONS</b>                                  | <b>76</b> |
| 8.1 Obtaining Metrics for Coordinate Transformations                  | 77        |
| 8.2 2D Governing Equations for Coordinate Transformations             | 79        |
| <b>9. ONE DIMENSIONAL RESULTS AND DISCUSSION</b>                      | <b>82</b> |
| 9.1 1D Results for Scalar Conservation Laws                           | 82        |
| 9.1.1 Piecewise Smooth Problem with Advection and Burger's Equations  | 82        |
| 9.1.2 Sin-wave Problems with Advection and Burger's Equations         | 85        |
| 9.1.3 Riemann Problems with Nonconvex Flux Function                   | 93        |
| 9.1.4 Buckley-Leverett Problem                                        | 96        |
| 9.2 1D Results for Systems of Conservation Laws                       | 97        |
| 9.2.1 Sod Shock Tube Problem                                          | 97        |

|                                                                                       |            |
|---------------------------------------------------------------------------------------|------------|
| 9.2.2 Lax Shock Tube Problem                                                          | 104        |
| 9.2.3 Moving Shock Wave Interacting with Fluctuations (Shu's Problem)                 | 108        |
| 9.2.4 Interaction of Two Blast Waves                                                  | 112        |
| 9.3 CPU Time Comparison for 1D Results                                                | 115        |
| <b>10. TWO DIMENSIONAL RESULTS AND DISCUSSION</b>                                     | <b>117</b> |
| 10.1 2D Results for Scalar Conservation Laws                                          | 117        |
| 10.1.1 Piecewise Smooth Problem with Advection and Burger's Equations                 | 118        |
| 10.1.2 Sin-wave Problems with Advection and Burger's Equations                        | 120        |
| 10.2 2D Results for Systems of Conservation Laws                                      | 124        |
| 10.2.1 Vortex Evolution Problem                                                       | 124        |
| 10.2.2 Shock Vortex Interaction Problem                                               | 131        |
| 10.2.3 Double Mach Reflection Problem                                                 | 135        |
| 10.2.4 Forward Facing Step Problem                                                    | 143        |
| 10.2.5 Supersonic Flow Past A Cylinder                                                | 146        |
| 10.2.6 Flow About NACA0012 Airfoil Cross Section                                      | 151        |
| 10.2.6.1 NACA0012 Airfoil Cross Section (Case-1 - $M = 0.8$ , $\alpha = 0^\circ$ )    | 151        |
| 10.2.6.2 NACA0012 Airfoil Cross Section (Case-2 - $M = 0.8$ , $\alpha = 1.25^\circ$ ) | 153        |
| 10.2.6.3 NACA0012 Airfoil Cross Section (Case-3 - $M = 0.85$ , $\alpha = 1.0^\circ$ ) | 155        |
| 10.2.6.4 NACA0012 Airfoil Cross Section (Case-4 - $M = 0.95$ , $\alpha = 0^\circ$ )   | 158        |
| 10.2.6.5 NACA0012 Airfoil Cross Section (Case-5 - $M = 1.2$ , $\alpha = 0^\circ$ )    | 160        |
| 10.2.6.6 NACA0012 Airfoil Cross Section (Case-6 - $M = 1.2$ , $\alpha = 7^\circ$ )    | 163        |
| 10.2.7 Planar Shock Interaction with NACA0018 Airfoil Cross Section                   | 167        |
| 10.3 CPU Time Comparison for 2D Results                                               | 172        |
| <b>11. CONCLUSION</b>                                                                 | <b>174</b> |
| <b>REFERENCE LIST</b>                                                                 | <b>176</b> |
| <b>APPENDIX-A ALTERNATIVE WAY FOR FINITE DIFFERENCE WENO</b>                          | <b>179</b> |
| <b>APPENDIX-B INTRODUCTION TO THE CODE</b>                                            | <b>182</b> |
| <b>CIRRICULUM VITAE</b>                                                               | <b>191</b> |

## **ABBREVIATIONS**

**ENO** : Essentially Non-Oscillatory  
**WENO** : Weighted Essentially Non-Oscillatory  
**TVD** : Total Variation Diminishing  
**GIP** : Gaussian Integration Point

## TABLE LIST

|                                                                                      | <u>Page</u> |
|--------------------------------------------------------------------------------------|-------------|
| <b>Table 3.1 :</b> The constants $c_{rj}$                                            | 25          |
| <b>Table 9.1 :</b> Norm Errors of $u_0(x) = \sin(\pi x)$ with Adv. Eq. (FD-LF-RK3)   | 89          |
| <b>Table 9.2 :</b> Norm Errors of $u_0(x) = \sin(\pi x)$ with Adv. Eq. (FD-LF-RK4)   | 90          |
| <b>Table 9.3 :</b> Norm Errors of $u_0(x) = \sin(\pi x)$ with Adv. Eq. (FV-LF-RK3)   | 91          |
| <b>Table 9.4 :</b> Norm Errors of $u_0(x) = \sin(\pi x)$ with Adv. Eq. (FV-LF-RK4)   | 91          |
| <b>Table 9.5 :</b> Norm Errors of $u_0(x) = \sin^4(\pi x)$ with Adv. Eq. (FV-LF-RK3) | 91          |
| <b>Table 9.6 :</b> Norm Err. of $u_0(x) = \sin^4(\pi x)$ with Adv. Eq. (FV-LLF-RK3)  | 91          |
| <b>Table 9.7 :</b> Norm Err. of $u_0(x) = \sin^4(\pi x)$ with Adv. Eq. (FV-LF-RK4)   | 92          |
| <b>Table 9.8 :</b> Norm Err. of $u_0(x) = \sin^4(\pi x)$ with Adv. Eq. (FV-LLF-RK4)  | 92          |
| <b>Table 9.9 :</b> 1D CPU Time Comparison (Characteristic-wise)                      | 115         |
| <b>Table 9.10 :</b> 1D CPU Time Comparison (Component-wise)                          | 116         |
| <b>Table 10.1 :</b> $L$ -Norm Errors of $\sin(\pi(x+y))$ with Adv. Eq. (FD-LF-RK3)   | 122         |
| <b>Table 10.2 :</b> $L$ -Norm Errors of $\sin(\pi(x+y))$ with Adv. Eq. (FD-LF-RK4)   | 122         |
| <b>Table 10.3 :</b> $L$ -Norm Errors of $\sin(\pi(x+y))$ with Adv. Eq. (FV-LF-RK3)   | 123         |
| <b>Table 10.4 :</b> $L$ -Norm Errors of $\sin(\pi(x+y))$ with Adv. Eq. (FV-LF-RK4)   | 123         |
| <b>Table 10.5 :</b> $L$ -Norm Err. of $\sin(\pi(x+y))$ with Adv. Eq. (FV-LLF-RK3)    | 123         |
| <b>Table 10.6 :</b> $L$ -Norm Err. of $\sin(\pi(x+y))$ with Adv. Eq. (FV-LLF-RK4)    | 124         |
| <b>Table 10.7 :</b> $C_L$ and $C_D$ Values Comparison for Test Cases                 | 166         |
| <b>Table 10.8 :</b> 2D CPU Time Comparison (Case-1)                                  | 172         |
| <b>Table 10.9 :</b> 2D CPU Time Comparison (Case-2)                                  | 172         |
| <b>Table 10.10 :</b> 2D CPU Time Comparison (Case-3)                                 | 172         |
| <b>Table 10.11 :</b> 2D CPU Time Comparison (Case-4)                                 | 172         |
| <b>Table 10.12 :</b> 2D CPU Time Comparison (Case-5)                                 | 172         |

## FIGURE LIST

|                                                                                            | <u>Page</u> |
|--------------------------------------------------------------------------------------------|-------------|
| <b>Figure 3.1 :</b> Relevant Range for Cell Boundary Values ( $I_i$ )                      | 20          |
| <b>Figure 3.2 :</b> Relevant Range for Right Boundary Value at $x_{i+1/2}$                 | 20          |
| <b>Figure 4.1 :</b> Stencil Selection for Piecewise-Quadratic ENO Reconstruction           | 31          |
| <b>Figure 4.2 :</b> Three Candidate Stencils for 5 <sup>th</sup> Order WENO Reconstruction | 33          |
| <b>Figure 5.1 :</b> Usage of Two Point Gaussian Integration Point Rule for WENO            | 55          |
| <b>Figure 6.1 :</b> A Non-Uniform Grid for the Stencil                                     | 58          |
| <b>Figure 7.1 :</b> Concept of Ghost Cells Approach                                        | 67          |
| <b>Figure 7.2 :</b> Concept of Characteristic Variables at Boundaries (2D)                 | 71          |
| <b>Figure 7.3 :</b> Wall Boundary Condition by Creating the Image of the Flow              | 74          |
| <b>Figure 7.4 :</b> Sketch of Multiblock Interfaces                                        | 75          |
| <b>Figure 9.1 :</b> Piece-wise Smooth Pr. with Advection Equation (FD)                     | 83          |
| <b>Figure 9.2 :</b> Piece-wise Smooth Pr. with Advection Equation (FV)                     | 83          |
| <b>Figure 9.3 :</b> Piece-wise Smooth Pr. with Burger's Eq. (FD & FV)                      | 84          |
| <b>Figure 9.4 :</b> Piece-wise Smooth Pr. with Burger's Eq. (LF & LLF)                     | 84          |
| <b>Figure 9.5 :</b> $\sin(\pi x)$ Problem with Advection Equation                          | 85          |
| <b>Figure 9.6 :</b> $\sin(\pi x)$ Pr. with Advection Equation (20 to 100 cells)            | 85          |
| <b>Figure 9.7 :</b> $\sin(\pi x)$ Pr. with Burger's Equation ( $t : 0.0s$ to $1.0s$ )      | 86          |
| <b>Figure 9.8 :</b> $\sin^4(\pi x)$ Pr. with Advection Equation ( $t : 0.0s$ to $100.0s$ ) | 86          |
| <b>Figure 9.9 :</b> $\sin^4(\pi x)$ Pr. with Advection Eq. (Figure 9.8 zoomed-in)          | 87          |
| <b>Figure 9.10 :</b> $\sin^4(\pi x)$ Pr. with Advection Equation (FV RK3 & RK4)            | 87          |
| <b>Figure 9.11 :</b> $\sin^4(\pi x)$ Pr. with Advection Equation (FD RK3 & RK4)            | 88          |
| <b>Figure 9.12 :</b> $\sin^4(\pi x)$ Pr. with Burger's Equation ( $t : 0.0s$ to $2.0s$ )   | 88          |
| <b>Figure 9.13 :</b> $L_1$ Norm Errors of $\sin(\pi x)$ Pr. with Advection Eq. (RK3)       | 89          |
| <b>Figure 9.14 :</b> $L_\infty$ Norm Errors of $\sin(\pi x)$ Pr. with Advection Eq. (RK3)  | 89          |

|                      |                                                                      |     |
|----------------------|----------------------------------------------------------------------|-----|
| <b>Figure 9.15 :</b> | $L_1$ Norm Errors of $\sin(\pi x)$ Pr. with Advection Eq. (RK4)      | 90  |
| <b>Figure 9.16 :</b> | $L_\infty$ Norm Errors of $\sin(\pi x)$ Pr. with Advection Eq. (RK4) | 90  |
| <b>Figure 9.17 :</b> | $L$ -Norm Errors of $\sin^4(\pi x)$ Pr. with Adv. Eq. (LF-RK4)       | 92  |
| <b>Figure 9.18 :</b> | $L$ -Norm Errors of $\sin^4(\pi x)$ Pr. with Adv. Eq. (LLF-RK4)      | 93  |
| <b>Figure 9.19 :</b> | Riemann Pr. with Nonconvex Flux Fnc. (Case 1 - FD)                   | 94  |
| <b>Figure 9.20 :</b> | Riemann Pr. with Nonconvex Flux Fnc. (Case 1 - FV)                   | 94  |
| <b>Figure 9.21 :</b> | Riemann Pr. with Nonconvex Flux Fnc. (Case 2 - FD & FV)              | 95  |
| <b>Figure 9.22 :</b> | Riemann Pr. with Nonconvex Flux Fnc. (Case 2 - FD)                   | 95  |
| <b>Figure 9.23 :</b> | Buckley-Leverett Problem (RK3 & RK4)                                 | 96  |
| <b>Figure 9.24 :</b> | Sod Shock Tube Pr. (Comp.-wise - FD - CFL:0.6 & 1.0)                 | 97  |
| <b>Figure 9.25 :</b> | Sod Shock Tube Pr. (Comp.-wise - FV - CFL:0.6 & 1.0)                 | 98  |
| <b>Figure 9.26 :</b> | Sod Shock Tube Pr. (Char.-wise - FD - CFL:0.6 & 1.0)                 | 98  |
| <b>Figure 9.27 :</b> | Sod Shock Tube Pr. (Char.-wise - FV - CFL:0.6 & 1.0)                 | 98  |
| <b>Figure 9.28 :</b> | Sod Shock Tube Problem (FD - LF & LLF)                               | 99  |
| <b>Figure 9.29 :</b> | Sod Shock Tube Problem (FV - LF & LLF)                               | 99  |
| <b>Figure 9.30 :</b> | Sod Shock Tube Problem (FD - RK3 & RK4)                              | 100 |
| <b>Figure 9.31 :</b> | Sod Shock Tube Problem (FV - RK3 & RK4)                              | 100 |
| <b>Figure 9.32 :</b> | Sod Shock Tube Problem (FD - 100 & 200 cells)                        | 101 |
| <b>Figure 9.33 :</b> | Sod Shock Tube Problem (FV - 100 & 200 cells)                        | 101 |
| <b>Figure 9.34 :</b> | Sod Shock Tube Problem, Other Variables (FD)                         | 102 |
| <b>Figure 9.35 :</b> | Sod Shock Tube Problem, Other Variables (FV)                         | 103 |
| <b>Figure 9.36 :</b> | Lax Shock Tube Problem (FD - Char. & Comp.)                          | 104 |
| <b>Figure 9.37 :</b> | Lax Shock Tube Problem (FV - Char. & Comp.)                          | 104 |
| <b>Figure 9.38 :</b> | Lax Shock Tube Problem (FD - 200 & 400 cells)                        | 105 |
| <b>Figure 9.39 :</b> | Lax Shock Tube Problem (FV - 200 & 400 cells)                        | 105 |
| <b>Figure 9.40 :</b> | Lax Shock Tube Problem, Other Variables (FD)                         | 106 |
| <b>Figure 9.41 :</b> | Lax Shock Tube Problem, Other Variables (FV)                         | 106 |
| <b>Figure 9.42 :</b> | Shu's Problem (FD - 200 cells)                                       | 108 |
| <b>Figure 9.43 :</b> | Shu's Problem (FD - 400 cells)                                       | 109 |
| <b>Figure 9.44 :</b> | Shu's Problem (FV - 200 cells)                                       | 109 |

|                       |                                                                        |     |
|-----------------------|------------------------------------------------------------------------|-----|
| <b>Figure 9.45 :</b>  | Shu's Problem (FV - 400 cells)                                         | 109 |
| <b>Figure 9.46 :</b>  | Shu's Problem, taken from [13] (FD - 200 & 400 nodes)                  | 110 |
| <b>Figure 9.47 :</b>  | Shu's Problem (FD - RK3 & RK4) and (FV - LF & LLF)                     | 110 |
| <b>Figure 9.48 :</b>  | Shu's Problem, Other Variables (FV)                                    | 111 |
| <b>Figure 9.49 :</b>  | Two Interacting Blast Waves Pr. (FD - 100 & 200 & 400 cells)           | 112 |
| <b>Figure 9.50 :</b>  | Two Interacting Blast Waves Pr. (FD - RK3 & RK4)                       | 113 |
| <b>Figure 9.51 :</b>  | Two Interacting Blast Waves Pr. (FD - RK3 - LF & LLF)                  | 113 |
| <b>Figure 9.52 :</b>  | Two Interacting Blast Waves Pr. (FD - RK4 - LF & LLF)                  | 113 |
| <b>Figure 9.53 :</b>  | Two Interacting Blast Waves Pr., Other Var. (FD - LLF)                 | 114 |
| <b>Figure 10.1 :</b>  | Piece-wise Smooth Pr. with Advection Eq. (FD & FV)                     | 118 |
| <b>Figure 10.2 :</b>  | Piece-wise Smooth Pr. with Advection Eq. ( $y$ -cut fig. 10.1)         | 118 |
| <b>Figure 10.3 :</b>  | Piece-wise Smooth Pr. with Burger's Eq. (FD & FV)                      | 119 |
| <b>Figure 10.4 :</b>  | Piece-wise Smooth Pr. with Burger's Eq. ( $y$ -cut from fig. 10.3)     | 119 |
| <b>Figure 10.5 :</b>  | $\sin(\pi(x + y))$ Pr. with Advection Equation (FD & FV)               | 120 |
| <b>Figure 10.6 :</b>  | Sin-wave Problem with Burger's Eq. (FD & FV)                           | 120 |
| <b>Figure 10.7 :</b>  | Sin-wave Problem with Burger's Eq. ( $y$ -cut from fig. 10.6)          | 121 |
| <b>Figure 10.8 :</b>  | Sin-wave Problem with Burger's Eq. ( $x$ -cut from fig. 10.6)          | 121 |
| <b>Figure 10.9 :</b>  | $L$ -Norm Err. of $\sin(\pi(x + y))$ Pr. with Adv. Eq. (FD&FV-RK3&RK4) | 122 |
| <b>Figure 10.10 :</b> | $L$ -Norm Errors $\sin(\pi(x + y))$ with Adv. Eq. (FV-LLF-RK3&RK4)     | 123 |
| <b>Figure 10.11 :</b> | Vortex Evolution Pr. in Density Var.(FV-LF-RK3&RK4)                    | 125 |
| <b>Figure 10.12 :</b> | Vortex Evolution Pr. in Density Var.(FV-LF-RK3&RK4)                    | 126 |
| <b>Figure 10.13 :</b> | Vortex Evolution Problem, taken from [2]                               | 126 |
| <b>Figure 10.14 :</b> | Vortex Evolution Pr. in Density Var. $dt = 1.0$ s (FD-LF-RK3)          | 127 |
| <b>Figure 10.15 :</b> | Vortex Evolution Problem Uniform Grid Comparison                       | 128 |
| <b>Figure 10.16 :</b> | Grid for Vortex Evolution Problem (Clustered Case-1)                   | 128 |
| <b>Figure 10.17 :</b> | Vortex Evolution Pr. Non-Uniform Grid Comparison (Case-1)              | 129 |
| <b>Figure 10.18 :</b> | Non-Uniform Grid Comparison (Fig. 10.17 zoomed in)                     | 129 |
| <b>Figure 10.19 :</b> | Grid for Vortex Evolution Problem (Clustered Case-2)                   | 130 |
| <b>Figure 10.20 :</b> | Non-Uniform Grid Comparison ( $x$ -cut Case-2)                         | 130 |

|                       |                                                                 |     |
|-----------------------|-----------------------------------------------------------------|-----|
| <b>Figure 10.21 :</b> | Non-Uniform Grid Comparison ( $y$ -cut Case-2)                  | 130 |
| <b>Figure 10.22 :</b> | Shock Vortex Int. with Uni. and Non-Uniform Grids (LF-RK3)      | 133 |
| <b>Figure 10.23 :</b> | Shock Vortex Int. with Uniform Grid (LLF-RK3&RK4)               | 133 |
| <b>Figure 10.24 :</b> | Shock Vort. Int. with Non-Uniform Grid ( $dt = 0.05$ s. LF-RK3) | 134 |
| <b>Figure 10.25 :</b> | Non-Uniform Grid for 2D Shock Vortex Interaction                | 135 |
| <b>Figure 10.26 :</b> | Double Mach Reflection 30 Contours from 1.731 to 20.92          | 137 |
| <b>Figure 10.27 :</b> | Double Mach Reflection, taken from [2]                          | 137 |
| <b>Figure 10.28 :</b> | Double Mach Reflection (Figure 10.26 zoomed-in)                 | 138 |
| <b>Figure 10.29 :</b> | Double Mach Reflection, Other Variables                         | 139 |
| <b>Figure 10.30 :</b> | Double Mach Reflection ( $dt = 0.04$ s)                         | 140 |
| <b>Figure 10.31 :</b> | Double Mach Ref. 30 Cont. from 1.731 to 20.92 (720x180 cells)   | 140 |
| <b>Figure 10.32 :</b> | Double Mach Ref. (720x180 cells, figure 10.31 zoomed-in)        | 141 |
| <b>Figure 10.33 :</b> | Double Mach Ref. 30 Cont. from 1.731 to 20.92 (960x240 cells)   | 141 |
| <b>Figure 10.34 :</b> | Double Mach Ref. (960x240 cells, figure 10.33 zoomed-in)        | 142 |
| <b>Figure 10.35 :</b> | Double Mach Ref. 30 Cont. from 1.731 to 20.92 (1440x360 cells)  | 142 |
| <b>Figure 10.36 :</b> | Double Mach Ref. (1440x360 cells, figure 10.35 zoomed-in)       | 143 |
| <b>Figure 10.37 :</b> | Forward Facing Step Line Contours at $t = 4.0$ s                | 144 |
| <b>Figure 10.38 :</b> | Forward Facing Step ( $dt = 0.4$ s)                             | 145 |
| <b>Figure 10.39 :</b> | Forward Facing Step Velocity Vectors                            | 146 |
| <b>Figure 10.40 :</b> | Forward Facing Step, taken from [2]                             | 146 |
| <b>Figure 10.41 :</b> | Physical Grid for Flow Past a Cylinder (Red:Ghost Cells)        | 147 |
| <b>Figure 10.42 :</b> | Line Contours of Flow Past a Cylinder in Pre. Var. (LF-RK3)     | 148 |
| <b>Figure 10.43 :</b> | Flood Contours of Flow Past a Cylinder in Pre. Var. (LF-RK3)    | 148 |
| <b>Figure 10.44 :</b> | Flow Past a Cylinder in Pre. Var. $t = 0.0$ s to 12.0s (LF-RK3) | 149 |
| <b>Figure 10.45 :</b> | Flow Past a Cylinder, Other Variables (LF-RK3)                  | 150 |
| <b>Figure 10.46 :</b> | Flow Past a Cylinder, taken from [33]                           | 150 |
| <b>Figure 10.47 :</b> | Grid for NACA0012 Airfoil Cross Section (180x49)                | 151 |
| <b>Figure 10.48 :</b> | Line Contours of NACA0012 (Case-1 Static Pressure)              | 152 |
| <b>Figure 10.49 :</b> | Flood Contours of NACA0012 (Case-1 Static Pressure)             | 152 |
| <b>Figure 10.50 :</b> | Flood Contours of NACA0012 (Case-1 Local Mach Number)           | 152 |

|                       |                                                            |     |
|-----------------------|------------------------------------------------------------|-----|
| <b>Figure 10.51 :</b> | $C_p$ Distribution of NACA0012 (Case-1)                    | 153 |
| <b>Figure 10.52 :</b> | $C_p$ Distribution of NACA0012 (Case-2)                    | 153 |
| <b>Figure 10.53 :</b> | Line Contours of NACA0012 (Case-2 Static Pressure)         | 154 |
| <b>Figure 10.54 :</b> | Flood Contours of NACA0012 (Case-2 Static Pressure)        | 154 |
| <b>Figure 10.55 :</b> | Flood Contours of NACA0012 (Case-2 Local Mach Number)      | 154 |
| <b>Figure 10.56 :</b> | Flood Contours of NACA0012 (Case-2 Entropy)                | 155 |
| <b>Figure 10.57 :</b> | $C_p$ Distribution of NACA0012 (Case-3)                    | 156 |
| <b>Figure 10.58 :</b> | Flood Contours of NACA0012 (Case-3 Static Pressure)        | 156 |
| <b>Figure 10.59 :</b> | Flood Contours of NACA0012 (Case-3 Local Mach Number)      | 156 |
| <b>Figure 10.60 :</b> | Line Contours of NACA0012 (Case-3 Local Mach Number)       | 157 |
| <b>Figure 10.61 :</b> | Line Contours (Case-3 Local Mach Number), taken from [35]  | 157 |
| <b>Figure 10.62 :</b> | $C_p$ Distribution of NACA0012 (Case-4)                    | 158 |
| <b>Figure 10.63 :</b> | Flood Contours of NACA0012 (Case-4 Static Pressure)        | 158 |
| <b>Figure 10.64 :</b> | Flood Contours of NACA0012 (Case-4 Local Mach Number)      | 159 |
| <b>Figure 10.65 :</b> | Line Contours of NACA0012 (Case-4 Local Mach Number)       | 159 |
| <b>Figure 10.66 :</b> | $C_p$ Distribution of NACA0012 (Case-5)                    | 160 |
| <b>Figure 10.67 :</b> | Continuous Flood Cont. of NACA0012 (Case-5 Total Pressure) | 160 |
| <b>Figure 10.68 :</b> | Flood Contours of NACA0012 (Case-5 Static Pressure)        | 161 |
| <b>Figure 10.69 :</b> | Flood Contours of NACA0012 (Case-5 Local Mach Number)      | 161 |
| <b>Figure 10.70 :</b> | Line Contours of NACA0012 (Case-5 Local Mach Number)       | 162 |
| <b>Figure 10.71 :</b> | Line Contours (Case-5 Local Mach Num.), taken from [35]    | 162 |
| <b>Figure 10.72 :</b> | $C_p$ Distribution of NACA0012 (Case-6)                    | 163 |
| <b>Figure 10.73 :</b> | Flood Contours of NACA0012 (Case-6 Static Pressure)        | 163 |
| <b>Figure 10.74 :</b> | Flood Contours of NACA0012 (Case-6 Local Mach Number)      | 164 |
| <b>Figure 10.75 :</b> | Continuous Flood Cont.of NACA0012 (Case-6 Total Pressure)  | 164 |
| <b>Figure 10.76 :</b> | Continuous Flood Contours of NACA0012 (Case-6 Entropy)     | 164 |
| <b>Figure 10.77 :</b> | Line Contours of NACA0012 (Case-6 Local Mach Number)       | 165 |
| <b>Figure 10.78 :</b> | Line Contours (Case-6 Local Mach Number), taken from [35]  | 165 |
| <b>Figure 10.79 :</b> | Grid for NACA0018 Airfoil Cross Section (298x79)           | 168 |
| <b>Figure 10.80 :</b> | Density Cont. at Some Time Instances, taken from [36]      | 168 |

|                       |                                                                   |     |
|-----------------------|-------------------------------------------------------------------|-----|
| <b>Figure 10.81 :</b> | Line Contours (Density) at Some Time Instances                    | 169 |
| <b>Figure 10.82 :</b> | Flood Contours (Static Pressure) at Some Time Instances           | 170 |
| <b>Figure 10.83 :</b> | Line Contours (Left:Density, Right:Static Pressure)               | 171 |
| <b>Figure 10.84 :</b> | Line Cont. (Left:Density, Right:Static Pressure), taken from [36] | 171 |
| <b>Figure B.1 :</b>   | Subroutine Invocation Chart                                       | 190 |

## SYMBOL LIST

|             |                                                |
|-------------|------------------------------------------------|
| $\rho$      | Density                                        |
| $u$         | x-velocity                                     |
| $v$         | y-velocity                                     |
| $p$         | Pressure                                       |
| $a$         | Sound speed                                    |
| $M$         | Mach number                                    |
| $s$         | Specific entropy                               |
| $e$         | Internal energy per unit mass                  |
| $E$         | Total energy per unit mass                     |
| $\gamma$    | Ratio of specific heats                        |
| $R$         | Gas constant                                   |
| $c_p$       | Specific heat coefficient at constant pressure |
| $c_v$       | Specific heat coefficient at constant volume   |
| $h$         | Enthalpy                                       |
| $H$         | Total (stagnation) enthalpy                    |
| $U$         | Conservative variables vector                  |
| $F$         | Flux vector in x-direction                     |
| $G$         | Flux vector in y-direction                     |
| $A$         | Jacobian matrix                                |
| $R$         | Right eigenvectors                             |
| $R^{-1}$    | Left eigenvectors                              |
| $\Lambda$   | Eigenvalues matrix                             |
| $\lambda_i$ | Eigenvalues                                    |
| $u_L$       | Left state variable                            |
| $u_R$       | Right state variable                           |
| $u_{RL}$    | Roe averaged variable                          |
| $\Omega$    | Control volume                                 |
| $F_u$       | Flux function vector in general coordinates    |
| $V$         | Contravariant velocity                         |
| $n_x$       | Unit normal vector in x-direction              |
| $n_y$       | Unit normal vector in y-direction              |
| $f(x_i)$    | Function                                       |
| $f[x_i]$    | Newton divided difference function             |
| $\bar{u}$   | Cell average of a function $u(x)$              |
| $\bar{v}$   | Cell average in characteristic field           |
| $c_{rj}$    | Stencil constants                              |
| $I_i$       | $i^{th}$ cell                                  |
| $S_i$       | $i^{th}$ stencil                               |

|                 |                                                      |
|-----------------|------------------------------------------------------|
| $d_r$           | Linear weight for $r^{th}$ stencil                   |
| $w_r$           | Non-linear weight for $r^{th}$ stencil               |
| $\beta_r$       | Smoothness indicator for $r^{th}$ stencil            |
| $u_t$           | Time derivative of the variable                      |
| $u_x$           | Spatial derivative of the variable                   |
| $\hat{f}$       | Numerical flux function                              |
| $\alpha$        | Constant of Lax-Friedrichs flux                      |
| $\beta_\alpha$  | Gaussian quadrature nodes                            |
| $\omega_\alpha$ | Gaussian quadrature weights                          |
| $h(a,b)$        | One-dimensional flux function                        |
| $x_g$           | Ghost cell node                                      |
| $x_b$           | Boundary node                                        |
| $x_i$           | Interior node                                        |
| $X$             | Contravariant vector component of distance           |
| $\xi, \eta$     | Constant lines in curvilinear coordinates wrt $x, y$ |
| $J$             | Jacobian matrix of transformation                    |
| $\tilde{F}$     | Function in general coordinates                      |
| $L_1$           | First norm                                           |
| $L_2$           | Second norm                                          |
| $L_\infty$      | Infinity norm                                        |
| $M_s$           | Mach number of shock wave                            |
| $C_p$           | Pressure coefficient                                 |
| $C_L$           | Lift coefficient                                     |
| $C_D$           | Drag coefficient                                     |

## SUMMARY

Computers represent functions by sequences of real numbers. These numbers are samples at this point. The samples  $f_i = f(x)$  represent the function. Samples are especially popular for solving differential equations, since differential equations are “pointwise” equations (i.e., they describe solutions in terms of rates of change at individual points).

There are two different ways to interpolate samples throughout the domain. One is finite difference method, and the other one is finite volume method. In addition to that finite elements method is another way, but this method is not in the scope of this work.

Finite difference methods use samples as their primary representation. They often switch from samples to functional representation in order to differentiate or to perform other functional operations. Any function created from samples is called a “reconstruction”. Any reconstruction passing through the sample points is called an “interpolation” on the domain. Finite volume method uses cell averages for reconstruction, and they integrate fluxes over the boundary.

Traditional finite difference and finite volume methods use fixed stencil approximation to interpolate functions. For a fixed stencil, to obtain an interpolation for cell  $i$  to third order accuracy, the information from the three cells  $i-1$ ,  $i$  and  $i+1$  can be used to build a second order interpolation polynomial. This works well for globally smooth problems. However, fixed stencil interpolation of second or higher order accuracy is necessarily “oscillatory” near a discontinuity. Such oscillations are called the Gibbs phenomena in spectral methods do not decay in magnitude even if the mesh is refined.

Earlier attempts to eliminate or reduce such spurious oscillations were mainly based on; adding explicit artificial viscosity and using limiters. However, both of these two methods have restrictions and difficulties, consequently it is not easy to implement them to problems.

The ENO idea seems to be the first successful attempt to obtain a self similar (i.e. no mesh size dependent parameter), uniformly high order accurate, yet essentially non-oscillatory interpolation (i.e. the magnitude of oscillations decays as  $O(\Delta x^k)$  where  $k$  is the order of accuracy) for piecewise smooth functions. The generic solution to hyperbolic conservation laws is in the class of piecewise smooth functions.

The basic idea of ENO schemes is the use of a Lagrange type interpolation with an adapted stencil. ENO chooses the “smoothest” stencil among several candidates to approximate the fluxes at cell boundaries to a high order accuracy and at the same time to avoid spurious oscillations near shocks.

ENO schemes are uniformly high order accurate right up to the shock and are very robust to use. However, they also have certain drawbacks. One problem is with

the freely adaptive stencil, which could change even by a round-off error perturbation near zeroes of the solution and its derivatives. Also, this free adaptation of stencils is not necessary in regions where the solution is smooth. Another problem is that ENO schemes are not cost effective on vector supercomputers, because stencil choosing process involves heavy uses of logical statements.

WENO scheme of Liu, Osher and Chan is another way to overcome these drawbacks while keeping the robustness and high order accuracy of ENO schemes. The idea is the following; instead of approximating the numerical flux using only one of the candidate stencils, use a “convex combination” of all the candidate stencils. Each of the candidate stencils is assigned a weight which determines the contribution of this stencil to the final approximation of the numerical flux.

WENO schemes completely remove the logical statements that appear in the ENO stencil choosing step. As a result, the WENO schemes run at least twice as fast as ENO schemes on vector machines and are not sensitive to round-off errors that arise in actual computation. Another advantage of WENO schemes is that its flux is smoother than that of the ENO schemes.

WENO is in the class of high order schemes. It is specially designed for long time simulations, non-oscillatory shock calculations, and it would show its real power when the solution contains both shocks and complex smooth region structures, such as shock turbulence interactions.

In this thesis project numerous one- and two-dimensional scalar and vectorial test cases are applied with the help of a Fortran 90 code generated for this work. The main subject, which is the flow about airfoil cross sections, is implemented for NACA0012 airfoil cross section for transonic and supersonic flow regimes. Results are presented in the corresponding sections. It is observed that the scheme works properly for more demanding problems as well, like Mach 1.5 and Mach 20 shock past an airfoil cross section.

## KANAT PROFİLLERİ ETRAFINDAKİ AKIŞIN WENO ŞEMALARI İLE SAYISAL ÇÖZÜMÜ

### ÖZET

Bilgisayarlar fonksiyonları ardışık reel sayılar ile gösterirler. Bu sayıları örnek sayılar olarak adlandırabiliriz. Örnek sayılar  $f_i = f(x)$  fonksiyonunu ifade ederler. Diferansiyel denklemler, çözümleri ayrık noktalardaki değişimler olarak gösterdikleri için çözümlerinde örnek nokta yaklaşımı popüler olarak kullanılmaktadır.

Örneklerin belirli bir alan içerisinde interpolasyonu için iki farklı yöntem kullanılır. Bunlardan birincisi sonlu fark yöntemi, diğeri ise sonlu hacim yöntemidir. Sonlu elemanlar yöntemi de ayrı bir yöntemdir, fakat sonlu elemalar yöntemi bu tez projesinin kapsamı içerisinde değildir.

Sonlu fark yöntemleri örnek noktaların kullanılmasına dayanır. Yöntem ile örnek noktalardan fonksiyon gösterimine geçilir, bu sayede türev alma gibi fonksiyon işlemleri yapılır. Örnek noktalardan oluşturulmuş olan fonksiyonlara “yeniden oluşturulmuş” denir. Yeniden oluşturulmuş fonksiyonların örnek noktalarla çakıştırılmasına ise o alandaki “interpolasyon” denir. Sonlu hacim yöntemi ise hücre değerlerinin ortalamasını kullanır ve akıları sınır üzerinde integre etme yöntemi ile çalışır.

Geleneksel sonlu fark ve sonlu hacim metodları sabit hesaplama molekülü yaklaşımı ile fonksiyonları interpolate eder. Mesela sabit hesaplama molekülü yöntemi ile  $i$  hücresinde üçüncü dereceden hassasiyete sahip bir interpolasyon bulmak için,  $i-1$ ,  $i$  ve  $i+1$  hücrelerinden ikinci dereceden bir interpolasyon polinomu bulunarak kullanılabilir. Bu yaklaşım genel düzgün problemler için iyi çalışır. Fakat ikinci ve daha büyük derecelerden sabit hesaplama molekülü yaklaşımı bir süreksizlik civarında esasen “salınımlıdır”. Bu salınımlara Gibbs olgusu denir ve sayısal ağdaki hücre büyüklükleri küçültülse bile değer olarak azalmazlar.

Salınımları yok etmek yada azaltmak için ilk zamanlarda yapay vizkozite ekleme ve limiter kullanma yöntemleri uygulanmaktaydı. Fakat, her iki yönteminde çeşitli kısıtlamaları ve zorlukları vardır ve problemlere uygulamak kolay değildir.

ENO düşüncesi, sayısal ağın hücre büyüklüğüne bağımlı parametresi olmayan ve düzgün yüksek dereceden hassasiyete sahip parçalı düzgün fonksiyonlar için uygulanmış ilk başarılı çalışma olarak görülmektedir. Hiperbolik korunum yasalarının genel çözümleri parçalı düzgün fonksiyonlar cinsinden olmaktadır.

ENO şemalarının temel düşüncesi adapte edilmiş hesaplama molekülü ile Lagrange tipinde interpolasyon kullanmaktır. ENO birçok aday hesaplama molekülü içerisinde en düzgün hesaplama molekülünü seçerek kullanır. Bu sayede hücre sınırlarındaki akıları yüksek hassasiyetle hesaplar ve şoklar etrafındaki salınımlardan kurtulmaya çalışır.

ENO şemaları, şoklar etrafında düzgün yüksek dereceden hassasiyete sahiptir ve kullanım açısından çok güçlüdür. Buna rağmen bazı dezavantajları vardır. İlk problem serbestçe adapte edilmiş hesaplama molekülü ile ilgilidir, çünkü çözümün yuvarlatma hatalarından bile etkilenebilmektedir. Buna ek olarak, serbestçe adapte edilmiş hesaplama molekülü yaklaşımı çözümün düzgün olduğu bölgelerde gereksiz olmaktadır. Bir diğer problem ise ENO şemalarının çokça mantık ifadeleri kullanmasından dolayı vektör süperbilgisayarlarında kullanımının etkin olamamasıdır.

Liu, Osher ve Chan bu tür dezavantajların üstesinden gelebilmek ve aynı zamanda ENO şemalarının güçlülüğünü ve yüksek hassasiyetini koruyabilmek için WENO şemalarını geliştirmişlerdir. WENO şemalarının uyguladığı düşünce; aday hesaplama moleküllerinden tek bir tanesini kullanmak yerine, hepsinin “konveks kombinasyonunu” kullanaktır. Bütün aday hesaplama moleküllerine, bu molekülün sayısal akının son ifadesine olan katkısını hesaplayan bir ağırlık katsayısı atanır.

WENO şemaları ENO şemalarının hesaplama molekülünü seçmek için kullanmak zorunda olduğu mantık ifadelerinden kurtulmuştur. Sonuç olarak WENO şemaları vektör süperbilgisayarlarında ENO şemalarından en az iki kat daha hızlı çalışır ve çözümlerde ortaya çıkan yuvarlatma hatalarına ENO kadar hassas değildir. WENO şemalarının bir diğer avantajı ise akısının ENO şemalarından daha düzgün olmasıdır.

WENO yüksek dereceli şemalar arasındadır. Özellikle uzun zaman simülasyonları, salınımsız şok çözümleri için dizayn edilmiştir, ve gerçek gücünü ise şok ile karmaşık akım alanının etkileşimini gerektiren problemlerde gösterir, buna örnek olarak şok türbülans etkileşimi verilebilir.

Bu tez projesinde çok sayıda bir ve iki boyutlu skaler ve vektörel test problemleri, bu işi için geliştirilmiş olan Fortran 90 kodu ile çözülmüştür. Tezin ana konusu olan kanat profilleri etrafındaki akışın çözümü için NACA0012 kanat profiline transonik ve süpersonik akış rejimleri uygulanmıştır. Sonuçlar ilgili bölümlerde sunulmuştur. Ayrıca şemanın daha talepkar problemler, mesela kanat profili üzerinden geçen Mach 1.5 ve Mach 20 şokları için de iyi sonuçlar verdiği gözlemlenmiştir.

## 1. INTRODUCTION

During the past few years, a growing interest appeared for constructing high order accurate and robust schemes for simulating compressible fluid flow. One of the difficulties is the appearance of strong discontinuities that may come out even for smooth initial data. In order to overcome this difficulty, a possibility is to use TVD (total variation diminishing) scheme. Such a scheme has the property, at least for 1D scalar equations, not to create new extrema, thus to provide a nice treatment to discontinuities. Nevertheless, one of their main weaknesses is that the order of accuracy must degenerate to first order in regions of discontinuity and at extrema, leading to excessive numerical dissipation [5]. A TVB (total variation bounded) modification of such schemes recovers global high order of accuracy even at critical points. However, the mentioned TVD or TVB schemes use a “fixed”, wide stencil, thus restricting the advantage of going to high order through smearing of discontinuities and resulting degradation of accuracy [3].

For a fixed stencil, to obtain an interpolation for cell  $i$  to third order accuracy, the information from the three cells  $i-1$ ,  $i$ , and  $i+1$  can be used to build a second order interpolation polynomial. This works well for globally smooth problems. However, fixed stencil interpolation of second or higher order accuracy is necessarily “oscillatory” near a discontinuity. Such oscillations are called the Gibbs phenomena in spectral methods; do not decay in magnitude even if the mesh is refined.

Earlier attempts to eliminate or reduce such spurious oscillations were mainly based on adding explicit artificial viscosity and using limiters.

Artificial viscosity can be tuned so that it is large enough near a discontinuity to suppress, or at least reduce oscillations, but is small elsewhere to maintain the order of accuracy. One disadvantage of this method is, fine tuning of the parameters of artificial viscosity is problem dependent.

Limiters can be applied to eliminate oscillations. It reduces the order of accuracy of the interpolation near discontinuity, for example, by reducing the slope of the linear interpolant, or by using a linear rather than a quadratic interpolant near a discontinuity. One disadvantage of limiters is accuracy necessarily degenerates to first order near smooth extrema, because of its TVD property.

ENO (Essentially Non-Oscillatory) schemes were first introduced by Harten, Engquist, Osher and Chakravarty in 1987 [9]. Their paper now has become a classic and has been quoted numerous times. The Journal of Computational Physics decided to republish it as part of the journal's celebration of its 30<sup>th</sup> birthday.

The ENO idea proposed in [9] seems to be the first successful attempt to obtain a self similar (i.e. no mesh size dependent parameter), uniformly high order accurate, yet essentially non-oscillatory interpolation (i.e. the magnitude of oscillations decays as  $O(\Delta x^k)$  where  $k$  is the order of accuracy) for piecewise smooth functions. The generic solution to hyperbolic conservation laws is in the class of piecewise smooth functions [1].

The basic idea of ENO schemes is the use of a Lagrange type interpolation with an adapted stencil; when a discontinuity is detected (when Newton divided/undivided differences are high compared to others) the procedure looks for a region around this discontinuity where the function is smooth or least oscillatory. This reconstruction technique may be applied either to the nodal values [3] or to particular function constructed from cell averages in control volumes [9]. In this latter case, the approximation is conservative. This enables one to approximate any piecewise smooth function with any order of accuracy [5].

ENO chooses the “smoothest” stencil among several candidates to approximate the fluxes at cell boundaries to a high order accuracy and at the same time to avoid spurious oscillations near shocks.

ENO schemes are uniformly high order accurate right up to the shock and are very robust to use. However, they also have certain drawbacks. One problem is with the freely adaptive stencil, which could change even by a round-off error perturbation near zeroes of the solution and its derivatives. Also, this free adaptation of stencils is not necessary in regions where the solution is smooth. Another problem

is that ENO schemes are not cost effective on vector supercomputers, because stencil choosing process involves heavy uses of logical statements

WENO scheme of Liu, Osher and Chan [10] is another way to overcome these drawbacks while keeping the robustness and high order accuracy of ENO schemes. The idea is the following; instead of approximating the numerical flux using only one of the candidate stencils, use a “convex combination” of all the candidate stencils. Each of the candidate stencils is assigned a weight which determines the contribution of this stencil to the final approximation of the numerical flux. The weights can be defined in such a way that in smooth regions it approaches certain optimal weights to achieve a higher order accuracy (a  $r^{th}$  order ENO scheme leads to a  $(2r - 1)^{th}$  order WENO scheme in the optimal case), while in regions near discontinuities, the stencils which contain discontinuities are assigned a nearly zero weight. Thus, the essentially non-oscillatory property is achieved by emulating ENO schemes around discontinuities and a higher order of accuracy is obtained by emulating upstream central schemes with optimal weights away from discontinuities. WENO schemes completely remove the logical statements that appear in the ENO stencil choosing step. As a result, the WENO schemes run at least twice as fast as ENO schemes on vector machines and are not sensitive to round-off errors that arise in actual computation. Another advantage of WENO schemes is that its flux is smoother than that of the ENO schemes.

Although the WENO schemes are faster than ENO schemes on vector supercomputers, they are only as fast as ENO schemes on serial computers. In addition to that, WENO schemes have the same smearing at contact discontinuities as ENO schemes [6].

The time discretization of ENO and WENO schemes has been implemented by a class of high order TVD and non-TVD Runge-Kutta type methods developed by Shu and Osher [3].

Today the study and application of ENO and WENO schemes are still very active. The authors of these schemes expect the schemes and the basic methodology to be developed further and to become even more successful in the future.

## 2. GOVERNING EQUATIONS AND POLYNOMIAL APPROXIMATION

In this section, governing equations that are used for the scheme, polynomial approximation and reconstruction procedures are explained. Polynomial approximation and reconstruction from discrete values (data) are the infrastructures of this work. Interpolation from those discrete data by using polynomials is given at the following subsections, and before explaining approximation and reconstruction a discussion about governing equations is given.

### 2.1 Governing Equations for 1D (Euler Equations)

In this thesis project Euler equations are used for solving systems of conservation laws. Although the WENO scheme can be implemented to other governing equations such as Navier Stokes, they are especially designed and used for Euler equations.

Navier Stokes equations give fluid flow motion at a most general and comprehensive case, and the Euler equations are some specialized case of these equations. Viscosity effects are neglected in Euler case in order to handle the equations in a simplified manner.

The conservation form of Euler equations for one dimensional case are expressed as the following, besides that, equation (2.1d) is optional.

$$\text{(Continuity Equation)} \quad \frac{\partial}{\partial t}(\rho) + \frac{\partial}{\partial x}(\rho u) = 0 \quad (2.1a)$$

$$\text{(Momentum Equation)} \quad \frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}(\rho u^2 + p) = 0 \quad (2.1b)$$

$$\text{(Energy Equation)} \quad \frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x}u(\rho E + p) = 0 \quad (2.1c)$$

$$\text{(Entropy Inequality)} \quad \frac{\partial}{\partial t}(\rho s) + \frac{\partial}{\partial x}(\rho us) \geq 0 \quad (2.1d)$$

Where  $\rho$  is the density,  $u$  is the x-velocity,  $p$  is the pressure (we also call these three as primitive variables),  $s$  is the specific entropy which is a function of specific internal energy and density [7, page11], and  $E$  is the total energy per unit mass  $\left(e + \frac{u^2}{2}\right)$  obtained by adding its internal energy per unit mass to its kinetic energy per unit mass [11, page10].

The vector form of the Euler equations consists of conservative variables vector  $U$ , and the corresponding flux function vector  $F$ .

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho E \end{bmatrix} \quad (2.2a)$$

$$F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (\rho E + p)u \end{bmatrix} \quad (2.2b)$$

Using this vector notation (2.2a) and (2.2b), the conservation form can be written as the following equation.

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} = 0 \quad (2.3)$$

Then by chain rule:

$$\frac{\partial F}{\partial x} = \frac{\partial F}{\partial U} \frac{\partial U}{\partial x} \quad (2.4)$$

Where  $\frac{\partial F}{\partial U}$  is called the Jacobian matrix of  $F$ . If we call the Jacobian matrix as  $A$ , equation (2.4) becomes:

$$\frac{\partial F}{\partial x} = A \frac{\partial U}{\partial x} \quad (2.5)$$

Then equation (2.3) can be written in the form below, and where  $A$  is:

$$\frac{\partial U}{\partial t} + A \frac{\partial U}{\partial x} = 0 \quad (2.6)$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{\gamma-3}{2}u^2 & (3-\gamma)u & \gamma-1 \\ -\rho u E + (\gamma-1)\rho u^3 & \rho E - \frac{3}{2}(\gamma-1)\rho u^2 & \rho u \end{bmatrix} \quad (2.7)$$

As a brief conclusion for this subsection, in our case the Euler equations of gas dynamics assumes perfect gas, and uses the following equalities in addition to equations (2.2) to solve the systems of equations.

$$R = c_p - c_v \quad (2.8a)$$

$$\gamma = \frac{c_p}{c_v} = 1.4 \quad (2.8b)$$

$$p = \rho R T = (\gamma - 1) \rho \left[ E - \frac{u^2}{2} \right] \quad (2.8c)$$

$$H = h + \frac{u^2}{2} \quad (2.8d)$$

$$H = E + \frac{p}{\rho} = \frac{a^2}{\gamma - 1} + \frac{u^2}{2} \quad (2.8e)$$

$$a = \sqrt{\frac{\gamma p}{\rho}} \quad (2.8f)$$

Where  $R$  is the gas constant,  $c_p$  and  $c_v$  are the specific heat coefficients at constant pressure and constant heat respectively,  $\gamma$  is the ratio of specific heats,  $h$  is the enthalpy,  $H$  is the total (stagnation) enthalpy, and  $a$  is the speed of sound.

Succeeding subsection explains how these Euler equations can be handled in a characteristic manner.

### 2.1.1 Characteristic Form of Euler Equations

The schemes ENO and WENO use two different kinds of solution algorithms to solve the systems.

First one is a component-by-component fashion, which has an alias as component-wise. This means that, the reconstruction process is being done for each of the component of the conservative variable vector separately. After that, exact or approximate Riemann solver is used to compute the corresponding fluxes at the cell faces. This component-wise version of ENO and WENO is simple and cost effective. They work reasonably well for many problems especially when the order of accuracy is not high like second and sometimes third order. However, when the order of accuracy is high or when the scheme is used for more complicated and demanding problems we should use the more costly but more effective characteristic decomposition, which is the latter one [2].

After some introduction for the reason to use characteristic decomposition, we can continue considering characteristic form of Euler equations.

Consider the equation (2.6), this equation is linear if  $A$  is constant, and quasi-linear if  $A = A(U, x, t)$ . An equation or systems of equations with complete wave description is sometimes called hyperbolic. This means equation (2.6) is hyperbolic if and only if  $A$  is diagonalizable. In other words that equation is hyperbolic if and only if

$$R^{-1}AR = \Lambda \quad (2.9)$$

for some matrix  $R$ , where  $\Lambda$  is a diagonal matrix. More specifically,  $\Lambda$  is a diagonal matrix whose diagonal elements  $\lambda_i$  are characteristic values or eigenvalues of  $A$ ,  $R$  is a matrix whose columns are right characteristic vectors or in other words right eigenvectors,  $R^{-1}$  is a matrix whose rows are left characteristic vectors or left eigenvectors of  $A$ .

Multiply both sides of equation (2.6) by  $R^{-1}$  to obtain:

$$R^{-1} \frac{\partial U}{\partial t} + R^{-1} A \frac{\partial U}{\partial x} = 0 \quad (2.10)$$

This is called a characteristic form of equation (2.6). Afterwards characteristic variables  $V$  are defined as the following.

$$dV = R^{-1}dU \quad (2.11)$$

Then the characteristic form becomes the following and the latter.

$$\frac{\partial V}{\partial t} + R^{-1}AR \frac{\partial V}{\partial x} = 0 \quad (2.12a)$$

$$\frac{\partial V}{\partial t} + \Lambda \frac{\partial V}{\partial x} = 0 \quad (2.12b)$$

This is also called the characteristic form, thus this time the equation is written in terms of characteristic variables  $V$  rather than conservative variables  $U$ . The characteristic form is a wave form, to see this consider the  $i^{th}$  equation of (2.12b).

$$\frac{\partial v_i}{\partial t} + \lambda_i \frac{\partial v_i}{\partial x} = 0 \quad (2.12c)$$

This equation looks like a scalar one, except that, for quasi-linear systems of equations  $\lambda_i$  depends on all characteristic variables, this means  $\lambda_i$  does not depend just on the single characteristic variable  $v_i$ . In addition to that, in ENO and WENO this non-linearity problem solved by using a trick, which is approximating the value at cell faces and freezing them, this process is explained in the following subsection.

Equation (2.12c) leads us to say that:

$$\text{If } \frac{dx}{dt} = \lambda_i, \quad \text{then } v_i = \text{const.} \quad (2.13)$$

The curves  $dx = \lambda_i dt$  are called wavefronts or characteristics, the variables  $v_i$  are called signals or information carried by waves or characteristic variables, and the characteristic values  $\lambda_i$  are called wavespeeds or characteristic speeds or signal speeds. Using this terminology, equation (2.13) implies that the  $i^{th}$  characteristic variable is constant along the  $i^{th}$  characteristic curve [7, page26].

We can conclude this subsection by giving right and left eigenvalues and eigenvectors for Euler equations of gas dynamics.

The wavespeeds  $\lambda_i$  are given to be;  $u - a$ ,  $u$ , and  $u + a$ .

Corresponding right characteristic matrix and left characteristic matrix are given as the following.

$$R = \begin{bmatrix} 1 & 1 & 1 \\ u - a & u & u + a \\ H - ua & \frac{1}{2}u^2 & H + ua \end{bmatrix} \quad (2.14)$$

$$R^{-1} = \begin{bmatrix} \frac{1}{2}\left(b_2 + \frac{u}{a}\right) & -\frac{1}{2}\left(b_1 + \frac{1}{a}\right) & \frac{b_1}{2} \\ 1 - b_2 & b_1 u & -b_1 \\ \frac{1}{2}\left(b_2 - \frac{u}{a}\right) & -\frac{1}{2}\left(b_1 - \frac{1}{a}\right) & \frac{b_1}{2} \end{bmatrix} \quad (2.15)$$

Where  $b_1 = \frac{\gamma - 1}{a^2}$  and  $b_2 = \frac{u^2 b_1}{2}$  [12].

### 2.1.2 Roe Averaging Procedure for Euler Equations

As shortly discussed in the previous subsection, in ENO and WENO schemes non-linearity problem is solved by some averaging process. Let's think about the situation where the Jacobian matrix  $A$  is not constant. The trouble is that now all the matrices  $R$ ,  $R^{-1}$ , and  $\Lambda$  are dependent upon  $U$ . So, in order to deal with that situation we must "freeze" them locally in order to carry out a similar procedure as in the constant coefficient case. Thus, to compute the flux at the cell boundary  $x_{i+1/2}$ , we would need an approximation to the Jacobian at the middle value  $U_{i+1/2}$ . This can be simply taken as the arithmetic mean

$$U_{i+1/2} = \frac{1}{2}(U_i + U_{i+1}) \quad (2.16)$$

or a more detailed average satisfying some nice properties like the mean value theorem

$$F_{i+1} - F_i = F'_{i+1/2}(U_{i+1} - U_i) \quad (2.17)$$

Roe averaging is such an example for the compressible Euler equations of gas dynamics and some other physical systems [2]. However, before going deep into Roe averaging it would be beneficial to have a glance at the attempts to solve this non-linearity problem.

According to his paper, [4] Roe says; if  $U$  is a vector and  $F$  is non-linear, then the problem involves non-linear algebraic equations together with, usually, logical conditions which express the fact that a given member of the wave system may be present either as a shockwave or as an expansion fan. In general, the most efficient way to solve these equations will depend on the system of conservation laws from which they derive. For the special case of unsteady Euler equations in one space dimension, an algorithm was devised by Godunov. Godunov supposed that the initial data could be replaced by a piecewise constant set of states with the discontinuities at  $x_{i+1/2}$ . He found the exact solution to this simplified problem. After some time step  $\Delta t$  (less than  $\Delta x$  divided by the greatest wavespeed found in the Riemann solutions) he replaced the exact solution by a new piecewise constant approximation, whilst preserving integral properties of the conserved variable  $U$ . The first major extension to this line approach was made by van Leer, who approximated the data, and the solution at each subsequent time level, by piecewise linear segments, allowing discontinuities between the segments. A parallel line of development was initiated by Glimm, who followed Godunov as far as the exact solution to the simplified problem, but then obtained the new approximation by a random sampling procedure.

These are some attempts before Roe averaging algorithm was found, and Roe continues; it seems to the present author that the expense of producing an accurate solution to the Riemann problem would only be justified if the abundance of information which is thereby made available could be put to some rather sophisticated use.

After that, Roe developed a solution algorithm to approximate the values at  $x_{i+1/2}$ , equations taken from this algorithm are given below.

$$\theta = \frac{\sqrt{\rho_L}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad (2.18)$$

$$u_{RL} = \theta u_L + (1 - \theta) u_R \quad (2.19)$$

Notice that,  $0 \leq \theta \leq 1$ . Then  $u_{RL}$  is somewhere between  $u_L$  and  $u_R$ . If  $0 \leq \theta \leq 1$  then averages such as (2.19) are called linear averages, linear interpolations, or convex linear combinations [7, page 88].

The Roe-averaged quantities are as follows.

$$\rho_{RL} = \sqrt{\rho_L \rho_R} \quad (2.20a)$$

$$u_{RL} = \frac{u_L \sqrt{\rho_L} + u_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad (2.20b)$$

$$H_{RL} = \frac{H_L \sqrt{\rho_L} + H_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad (2.20c)$$

$$a_{RL} = \sqrt{(\gamma - 1) \left( H_{RL} - \frac{1}{2} u_{RL}^2 \right)} \quad (2.20d)$$

Roe-averaged wavespeeds are given below.

$$\lambda_1 = u_{RL} \quad (2.20e)$$

$$\lambda_2 = u_{RL} + a_{RL} \quad (2.20f)$$

$$\lambda_3 = u_{RL} - a_{RL} \quad (2.20g)$$

Then the Roe-averaged Jacobian matrix is the same as the Jacobian matrix given before, except that we are now using the averaged quantities.

$$A_{RL} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{\gamma - 3}{2} u_{RL}^2 & (3 - \gamma) u_{RL} & \gamma - 1 \\ -u_{RL} H_{RL} + \frac{1}{2} (\gamma - 1) u_{RL}^3 & H_{RL} - (\gamma - 1) u_{RL}^2 & \gamma u_{RL} \end{bmatrix} \quad (2.20h)$$

## 2.2 Extensions to Second Dimension

This thesis project involves writing an inviscid code about airfoil cross sections, therefore one-dimensional approximation is not sufficient and an extension to second dimension should be made.

In this subsection governing equations for two-dimensional inviscid flow is considered, additionally left and right eigenvectors and eigenvalues for two-dimensional flow is given. In addition to that, Roe averaging process is extended to the second dimension.

Before explaining two-dimensional equation in a dimension-by-dimension fashion, a general perspective is needed. In general coordinates, conservation law and governing equations for Euler equations is as the following

$$\frac{\partial}{\partial t} \int_{\Omega} U d\Omega + \oint_{d\Omega} F_u dS = 0 \quad (2.21)$$

here  $\Omega$  corresponds to control volume and  $d\Omega$  corresponds to the area that surrounds it. Conservative variables vector  $U$ , and the corresponding flux function vector  $F_u$ , in general two dimensions is as the following

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \quad (2.22a)$$

$$F_u = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ (\rho E + p)V \end{bmatrix} \quad (2.22b)$$

where  $V$  is the contravariant velocity (the velocity normal to the surface element  $dS$ ) being defined as the scalar product of the velocity vector and the unit normal vector [11, page 16].

$$V \equiv \vec{v} \cdot \vec{n} = n_x u + n_y v \quad (2.23)$$

In WENO schemes, we used a structured dimension-by-dimension reconstruction procedure for this thesis project, therefore the flux functions are modified in the following fashion ( $n_x = 1$ ,  $n_y = 0$  for i-direction, and  $n_x = 0$ ,  $n_y = 1$  for j-direction), where conservative variables vector  $U$  is given in (2.22a), and the corresponding flux function vectors are  $F$  and  $G$ .

$$F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho E + p)u \end{bmatrix} \quad (2.24a)$$

$$G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho E + p)v \end{bmatrix} \quad (2.24b)$$

Using this vector notation (2.24a) and (2.24b), the conservation form can be written as the following equation:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \quad (2.25)$$

Gas dynamics equations should be modified in the following manner:

$$H = h + \frac{u^2 + v^2}{2} \quad (2.26a)$$

$$E = e + \frac{u^2 + v^2}{2} \quad (2.26b)$$

$$p = \rho RT = (\gamma - 1)\rho \left[ E - \frac{u^2 + v^2}{2} \right] \quad (2.26c)$$

Roe-averaged quantities in two dimensions should be modified in the following manner:

$$v_{RL} = \frac{v_L \sqrt{\rho_L} + v_R \sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad (2.27a)$$

$$q_{RL} = \frac{u_{RL}^2 + v_{RL}^2}{2} \quad (2.27b)$$

$$a_{RL} = \sqrt{(\gamma - 1)(H_{RL} - q_{RL})} \quad (2.27c)$$

$$V_{RL} = n_x u_{RL} + n_y v_{RL} \quad (2.27d)$$

Right and left eigenvalues and eigenvectors for Euler equations of gas dynamics in general two dimensions is as the following.

The wavespeeds  $\lambda_{ij}$  are given to be;  $V_{RL} - a_{RL}$ ,  $V_{RL}$ ,  $V_{RL}$ , and  $V_{RL} + a_{RL}$ , and the corresponding Roe-averaged right characteristic matrix and left characteristic matrix are given as:

$$R = \begin{bmatrix} 1 & 0 & 1 & 1 \\ u_{RL} - n_x a_{RL} & n_y & u_{RL} & u_{RL} + n_x a_{RL} \\ v_{RL} - n_y a_{RL} & -n_x & v_{RL} & v_{RL} + n_y a_{RL} \\ H_{RL} - V_{RL} a_{RL} & u_{RL} n_y - v_{RL} n_x & q_{RL} & H_{RL} + V_{RL} a_{RL} \end{bmatrix} \quad (2.28a)$$

$$R^{-1} = \begin{bmatrix} \frac{1}{2} \left( b_2 + \frac{V_{RL}}{a_{RL}} \right) & -\frac{1}{2} \left( b_1 u_{RL} + \frac{n_x}{a_{RL}} \right) & -\frac{1}{2} \left( b_1 v_{RL} + \frac{n_y}{a_{RL}} \right) & \frac{1}{2} b_1 \\ -u_{RL} n_y + v_{RL} n_x & n_y & -n_x & 0 \\ 1 - b_2 & b_1 u_{RL} & b_1 v_{RL} & -b_1 \\ \frac{1}{2} \left( b_2 - \frac{V_{RL}}{a_{RL}} \right) & -\frac{1}{2} \left( b_1 u_{RL} - \frac{n_x}{a_{RL}} \right) & -\frac{1}{2} \left( b_1 v_{RL} - \frac{n_y}{a_{RL}} \right) & \frac{1}{2} b_1 \end{bmatrix} \quad (2.28b)$$

Where  $b_1 = \frac{\gamma - 1}{a_{RL}^2}$  and  $b_2 = q_{RL} b_1$ .

### 2.3 One Dimensional Polynomial Approximation and Reconstruction

Computers represent functions by sequences of real numbers. These numbers are samples at this point. Consider any set of sample points  $(x_0, x_1, \dots, x_N)$  in the domain of a function  $f(x)$ . Then the samples  $f_i = f(x_i)$  represent the function. The spacing between samples is  $\Delta x = x_i - x_{i-1}$ . Samples are especially popular for solving differential equations, since differential equations are “pointwise” equations

(i.e., they describe solutions in terms of rates of change at individual points). Finite difference methods use samples as their primary representation.

Finite difference methods often switch from samples to functional representation in order to differentiate or to perform other functional operations. Any function created from samples is called a “reconstruction”. Any reconstruction passing through the sample points is called an “interpolation” on the domain  $[x_0, x_N]$  and an “extrapolation” on the domains  $(-\infty, x_0)$  and  $(x_N, +\infty)$ .

One can find a unique  $N^{th}$  order polynomial passing through any set of  $N + 1$  samples. For example, there is a unique line passing through any two points, there is a unique quadratic passing through any three points, there is a unique cubic passing through any four points, and so on. Depending on where it is evaluated, this polynomial is either an interpolation or an extrapolation polynomial [7, page133].

In the following subsections, Lagrange type and Newton type polynomials are discussed. ENO and WENO use Lagrange type polynomials to find some constants in order to interpolate the values at the cell faces. They also use Newton type polynomials in order to interpolate the function within the stencil they use. In addition to that, they use Newton divided/undivided differences to indicate the smoothness of the function inside the stencil.

### 2.3.1 Lagrange Form Polynomial

The Lagrange form of a polynomial is defined as follows.

$$p_N(x) = \sum_{i=0}^N a_i (x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_{N-1})(x - x_N) \quad (2.29)$$

Notice that the sum skips the factor  $(x - x_i)$ . The coefficients  $a_i$  of the interpolation polynomial are found by solving the linear system of equations  $f(x_i)$ .

$$a_i = \frac{f(x_i)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_{N-1})(x_i - x_N)} \quad (2.30)$$

Then the Lagrange form of the interpolation polynomial can be written as:

$$p_N(x) = \sum_{i=0}^N f(x_i) \prod_{j=0, j \neq i}^N \frac{x - x_j}{x_i - x_j} \quad (2.31)$$

The Lagrange form is easy to derive and easy to remember, but may be difficult to work with, like integration and differentiation operations [7, page134].

### 2.3.2 Newton Form Polynomial

The Newton form of a polynomial is defined as the following two equal forms.

$$p_N(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots + a_N(x - x_0) \dots (x - x_{N-1}) \quad (2.32a)$$

$$p_N(x) = a_0 + \sum_{i=1}^N a_i \prod_{j=0}^{i-1} (x - x_j) \quad (2.32b)$$

The coefficients  $a_i$  of the interpolation polynomial are found by solving the triangular linear system of equations constituted by  $f(x_i)$ 's, for example for i:

$$f(x_i) = a_0 + a_1(x_i - x_0) + \dots + a_i(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1}) \quad (2.33)$$

Finding  $a_0$  and  $a_1$  is easy, however the subsequent coefficients are progressively more difficult to solve. Luckily there is a recursive solution, to state this solution we must first define Newton divided difference.

The zeroth, first, second and  $n^{th}$  Newton divided differences are defined respectively as the following equations.

$$f[x_i] = f(x_i) \quad (2.34a)$$

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i} = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \quad (2.34b)$$

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i} \quad (2.34c)$$

$$f[x_i, \dots, x_{i+n}] = \frac{f[x_{i+1}, \dots, x_{i+n}] - f[x_i, \dots, x_{i+n-1}]}{x_{i+n} - x_i} \quad (2.34d)$$

It is interesting to note the connection between the Newton divided differences and functional derivatives. If  $f(x)$  has  $n$  continuous derivatives in  $[x_i, x_{i+n}]$  then

$$f[x_i, \dots, x_{i+n}] = \frac{1}{n!} \frac{d^n f}{dx^n}(\xi) \quad (2.35)$$

for some  $x_i < \xi < x_{i+n}$ . If the  $p^{th}$  derivative of  $f(x)$  has a jump discontinuity at  $z$  in  $[x_i, x_{i+n}]$ , but otherwise  $f(x)$  is continuously differentiable, then

$$f[x_i, \dots, x_{i+n}] = O\left(\frac{1}{\Delta x^{n-p}} \left( \frac{d^p f(z_R)}{dx^p} - \frac{d^p f(z_L)}{dx^p} \right)\right) \quad (2.36)$$

for all  $n \geq p$ , where  $z_L$  and  $z_R$  refer to the left- and right-hand limits of the  $p^{th}$  derivative, respectively. Thus the  $n^{th}$  Newton divided difference is proportional to the jump in the  $p^{th}$  derivative divided by  $\Delta x^{n-p}$ . It is necessary to say here that; ENO exploits this property for indicating the measurement of the smoothness of the function inside the stencil it uses.

The coefficients of the Newton form of the interpolation polynomial can now be easily defined in terms of Newton divided differences.

$$a_i = f[x_0, x_1, \dots, x_i] \quad (2.37)$$

In other words, the Newton form of interpolation polynomial is as follows.

$$p_N(x) = f[x_0] + \sum_{i=1}^N f[x_0, \dots, x_i] \prod_{j=0}^{i-1} (x - x_j) \quad (2.38)$$

### 3. ONE DIMENSIONAL APPROXIMATION AND RECONSTRUCTION

In this section polynomial approximation and reconstruction specific for ENO and WENO schemes are explained. Necessary background for this section is already given in the previous section. There are two kind of approximations used for solving hyperbolic conservation laws. First one is the “finite volume method”, and second one is the “finite difference method”. Finite volume method uses cell averages for reconstruction, however finite difference method uses point values to approximate derivatives. Both methods are explained in the following subsections.

#### 3.1 Constructing the Grid

The grid used for one dimension is given with the following features.

$$a = x_{1/2} < x_{3/2} < \dots < x_{N-1/2} < x_{N+1/2} = b \quad (3.1)$$

Cells, cell centers and cells sizes are given respectively as the following

$$I_i = \left[ x_{i-1/2}, x_{i+1/2} \right] \quad (3.2)$$

$$x_i = \frac{1}{2} \left( x_{i-1/2} + x_{i+1/2} \right) \quad (3.3)$$

$$\Delta x = x_{i+1/2} - x_{i-1/2}, \quad i = 1, 2, \dots, N \quad (3.4)$$

#### 3.2 One Dimensional Reconstruction from Cell Averages (Finite Volume)

Given the cell averages of a function  $v(x)$ , where  $i = 1, 2, \dots, N$  [1]

$$\bar{v}_i \equiv \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} v(\xi) d\xi \quad (3.5)$$

find a polynomial  $p_i(x)$ , for each cell  $I_i$ , of degree at most  $k-1$ , such that it is a  $k^{th}$  order accurate approximation to the function  $v(x)$  inside the cell. In addition to that, we assume the boundary (ghost cell) values of cell averages are available.

$$p_i(x) = v(x) + O(\Delta x^k), \quad x \in I_i \quad (3.6)$$

In particular, this gives approximations to the function  $v(x)$  at the cell boundaries

$$v_{i+1/2}^- = p_i\left(x_{i+1/2}\right) \quad (3.7a)$$

$$v_{i-1/2}^+ = p_i\left(x_{i-1/2}\right) \quad (3.7b)$$

which are  $k^{th}$  order accurate

$$v_{i+1/2}^- = v\left(x_{i+1/2}\right) + O(\Delta x^k) \quad (3.8a)$$

$$v_{i-1/2}^+ = v\left(x_{i-1/2}\right) + O(\Delta x^k) \quad (3.8b)$$

The figures 3.1 and 3.2 are showing the relevant ranges for the corresponding boundary values; that means from where the cell boundary values at  $x_{i+1/2}$  are constructing the polynomials for WENO  $5^{th}$  order scheme from their 3 candidate stencils (where the first candidate is from  $i-2$  to  $i$ , the second one is from  $i-1$  to  $i+1$ , and the last one is from  $i$  to  $i+2$ , a special figure for candidate stencil case is given in section 4). It must be stated here that, these two figures 3.1 and 3.2 are ranges special for ENO  $3^{rd}$  order and WENO  $5^{th}$  order schemes. If the order of accuracy used for the scheme want to be increased, then the width of the stencil must be increased according to the desired order. For example, a  $4^{rd}$  order ENO scheme chooses from 4 candidate stencils, where each of them contains 4 cells.

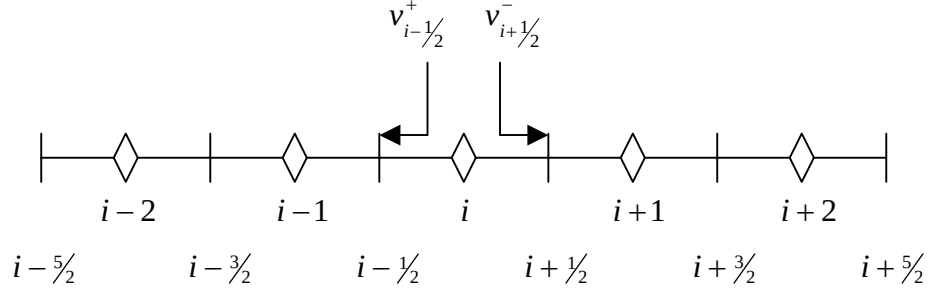


**Figure 3.1 :** Relevant Range for Cell Boundary Values ( $I_i$ )

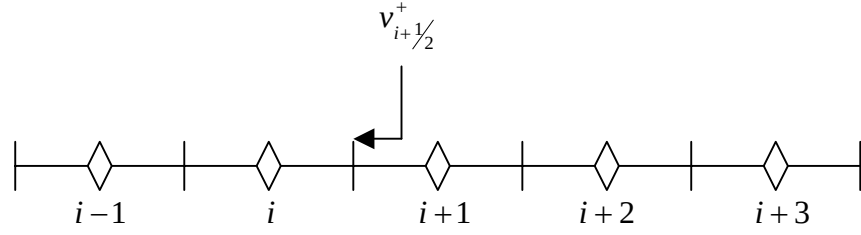


**Figure 3.2 :** Relevant Range for Right Boundary Value at  $x_{i+1/2}$

Given the location  $I_i$  and the order of accuracy  $k$  ( $(2k-1)=5^{th}$  order at this case), we first choose a “stencil”, based on  $r$  cells to the left, and  $s$  cells to the right, and the cell  $I_i$  itself. Where the following notation gives the stencil:

$$S_i \equiv \{I_{i-r}, \dots, I_{i+s}\}, \quad r, s \geq 0, \quad r + s + 1 = k \quad (3.9)$$

There is a unique polynomial of degree at most  $k-1=r+s$ , denoted by  $p(x)$  (we will drop the subscript  $i$  when it does not make confusion), whose cell average in each of the cells in the stencil agrees with that of  $v(x)$ .

$$\frac{1}{\Delta x_j} \int_{x_{j-1/2}}^{x_{j+1/2}} p(\xi) d\xi = \bar{v}_j, \quad j = i-r, \dots, i+s \quad (3.10)$$

As long as the function  $v(x)$  is smooth in the region covered by the stencil  $S(i)$ , this polynomial  $p(x)$  is the  $k^{th}$  order approximation we are looking for.

We also need the approximation to the cell boundary values of  $v(x)$ . Since the mappings from the given cell averages  $\bar{v}_j$  in the stencil  $S(i)$  to the values  $v_{i+1/2}^-$

and  $v_{i-1/2}^+$  by (3.7) are linear, there exist constants  $c_{rj}$  and  $\tilde{c}_{rj}$ , which depend on the left shift  $r$  of the stencil  $S(i)$ , on the order of accuracy  $k$ , and on the cell sizes  $\Delta x_j$  in the stencil, but not on the function  $v$  itself.

$$v_{i+1/2}^- = \sum_{j=0}^{k-1} c_{rj} \bar{v}_{i-r+j} \quad (3.11a)$$

$$v_{i+1/2}^+ = \sum_{j=0}^{k-1} \tilde{c}_{rj} \bar{v}_{i-r+j} \quad (3.11b)$$

$$\tilde{c}_{rj} = c_{r-1,j} \quad (3.12)$$

As a note; the difference between the values with superscripts  $\mp$  at the same location  $x_{i+1/2}$  is due to the possibility of different stencils for the corresponding cells, as mentioned earlier in the figures 3.1 and 3.2 which are showing these relevant ranges for 5<sup>th</sup> order WENO.

Obtaining procedure of these constants  $c_{rj}$  is not trivial. In order to obtain these constants, which will be called as linear weights later on, first of all the Lagrange form interpolation polynomial in equation (3.6) has to be found.

Summarizing this procedure is as the following. Initially, we can get rid of the superscripts  $\mp$  by restricting ourselves to only one reconstruction point like  $x_{i+1/2}$ . It is clear from equation (3.12) that we can shift the reconstruction point (so the stencil) and then we can use the corresponding constants.

Given the  $k$  cell averages  $\bar{v}_{i-r}, \dots, \bar{v}_{i-r+k-1}$  there are constants  $c_{rj}$  such that the reconstructed value at the cell boundary  $x_{i+1/2}$ :

$$v_{i+1/2} = \sum_{j=0}^{k-1} c_{rj} \bar{v}_{i-r+j} \quad (3.13)$$

is  $k^{th}$  order accurate:

$$v_{i+1/2} = v\left(x_{i+1/2}\right) + O(\Delta x^k) \quad (3.14)$$

“Reconstruction via primitive function” technique in [9] can be applied to obtain the primitive function of  $v(x)$ :

$$V(x) = \int_{-\infty}^x v(\xi) d\xi \quad (3.15)$$

where lower limit  $-\infty$  is not important and can be replaced by any fixed number.

$$V\left(x_{i+\frac{1}{2}}\right) = \sum_{j=-\infty}^i \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} v(\xi) d\xi = \sum_{j=-\infty}^i \bar{v}_j \Delta x_j \quad (3.16)$$

Thus, with the knowledge of the cell averages  $\bar{v}_j$  we also know the primitive function  $V(x)$  at the cell boundaries exactly. The unique polynomial of degree at most  $k$ , which interpolates  $V\left(x_{j+\frac{1}{2}}\right)$  at the following  $k+1$  points  $x_{i-r+\frac{1}{2}}, \dots, x_{i+s+\frac{1}{2}}$  is marked by  $P(x)$ , and the derivative of this polynomial is marked by  $p(x)$ , thus:

$$p(x) \equiv P'(x) \quad (3.17a)$$

$$\frac{1}{\Delta x_j} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} p(\xi) d\xi = \frac{1}{\Delta x_j} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} P'(\xi) d\xi \quad (3.17b)$$

$$= \frac{1}{\Delta x_j} \left( P\left(x_{j+\frac{1}{2}}\right) - P\left(x_{j-\frac{1}{2}}\right) \right) \quad (3.17c)$$

$$= \frac{1}{\Delta x_j} \left( V\left(x_{j+\frac{1}{2}}\right) - V\left(x_{j-\frac{1}{2}}\right) \right) \quad (3.17d)$$

$$= \frac{1}{\Delta x_j} \left( \int_{-\infty}^{x_{j+\frac{1}{2}}} v(\xi) d\xi - \int_{-\infty}^{x_{j-\frac{1}{2}}} v(\xi) d\xi \right) \quad (3.17e)$$

$$= \frac{1}{\Delta x_j} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} v(\xi) d\xi \quad (3.17f)$$

$$= \bar{v}_j \text{ where } j = i-r, \dots, i+s \quad (3.17g)$$

In this proof the third equality holds because  $P(x)$  interpolates  $V(x)$  at the points  $x_{j-\frac{1}{2}}$  and  $x_{j+\frac{1}{2}}$  whenever  $j = i - r, \dots, i + s$ .

Thus, we can claim that  $p(x)$  is the interpolation polynomial that we are looking for. From the standard approximation theory:

$$P'(x) = V'(x) + O(\Delta x^k), \quad x \in I_i \quad (3.18)$$

Lagrange form of interpolation polynomials were discussed in section 2.2.1 earlier, thus the interpolation polynomial that we use is like the equation (2.31), which is:

$$P(x) = \sum_{m=0}^k V\left(x_{i-r+m-\frac{1}{2}}\right) \prod_{l=0, l \neq m}^k \frac{x - x_{i-r+l-\frac{1}{2}}}{x_{i-r+m-\frac{1}{2}} - x_{i-r+l-\frac{1}{2}}} \quad (3.19)$$

For easier manipulation subtracting a constant  $V\left(x_{i-r-\frac{1}{2}}\right)$  from equation (3.19) and use the fact (3.20), will lead us to obtain the equation (3.21).

$$\sum_{m=0}^k \prod_{l=0, l \neq m}^k \frac{x - x_{i-r+l-\frac{1}{2}}}{x_{i-r+m-\frac{1}{2}} - x_{i-r+l-\frac{1}{2}}} = 1 \quad (3.20)$$

$$P(x) - V\left(x_{i-r-\frac{1}{2}}\right) = \sum_{m=0}^k \left( V\left(x_{i-r+m-\frac{1}{2}}\right) - V\left(x_{i-r-\frac{1}{2}}\right) \right) \prod_{l=0, l \neq m}^k \frac{x - x_{i-r+l-\frac{1}{2}}}{x_{i-r+m-\frac{1}{2}} - x_{i-r+l-\frac{1}{2}}} \quad (3.21)$$

Taking derivative on both sides of the equation (3.21), and because of the equation (3.16):

$$V\left(x_{i-r+m-\frac{1}{2}}\right) - V\left(x_{i-r-\frac{1}{2}}\right) = \sum_{j=0}^{m-1} \bar{v}_{i-r+j} \Delta x_{i-r+j} \quad (3.22)$$

$$p(x) = \sum_{m=0}^k \sum_{j=0}^{m-1} \bar{v}_{i-r+j} \Delta x_{i-r+j} \left( \frac{\sum_{l=0, l \neq m}^k \prod_{q=0, q \neq m, l}^k \left( x - x_{i-r+q-\frac{1}{2}} \right)}{\prod_{l=0, l \neq m}^k \left( x_{i-r+m-\frac{1}{2}} - x_{i-r+l-\frac{1}{2}} \right)} \right) \quad (3.23)$$

Thus, we find the interpolation polynomial that we are looking for.

If we evaluate the expression (3.23) at the cell boundary  $x = x_{i+\frac{1}{2}}$ , we obtain:

$$v_{i+\frac{1}{2}} = p\left(x_{i+\frac{1}{2}}\right) = \sum_{j=0}^{k-1} \bar{v}_{i-r+j} \Delta x_{i-r+j} \sum_{m=j+1}^k \left( \frac{\sum_{l=0, l \neq m}^k \prod_{q=0, q \neq m, l}^k \left( x_{i+\frac{1}{2}} - x_{i-r+q-\frac{1}{2}} \right)}{\prod_{l=0, l \neq m}^k \left( x_{i-r+m-\frac{1}{2}} - x_{i-r+l-\frac{1}{2}} \right)} \right) \quad (3.24)$$

Then the  $c_{rj}$  in equation (3.13) is given by:

$$c_{rj} = \Delta x_{i-r+j} \sum_{m=j+1}^k \left( \frac{\sum_{l=0, l \neq m}^k \prod_{q=0, q \neq m, l}^k \left( x_{i+\frac{1}{2}} - x_{i-r+q-\frac{1}{2}} \right)}{\prod_{l=0, l \neq m}^k \left( x_{i-r+m-\frac{1}{2}} - x_{i-r+l-\frac{1}{2}} \right)} \right) \quad (3.25)$$

In the case of uniform grid, where  $\Delta x_i = \Delta x$ , the expression for  $c_{rj}$  does not depend on  $i$  or  $\Delta x$  any more, thus the expression becomes [1]:

$$c_{rj} = \sum_{m=j+1}^k \left( \frac{\sum_{l=0, l \neq m}^k \prod_{q=0, q \neq m, l}^k (r - q + 1)}{\prod_{l=0, l \neq m}^k (m - l)} \right) \quad (3.26)$$

Something has to be clarified here that; the expression for  $c_{rj}$  is always constant with respect to time, but they are only constant with respect to space if and only if the grid is uniform, otherwise, for a non-uniform grid case, it will vary along the space with respect to the corresponding cell size.

A more general discussion about non-uniform grid case will be held in the section 6. All of the expressions needed to construct a non-uniform 3<sup>rd</sup> and 5<sup>th</sup> order WENO are derived in the scope of this thesis project. Those expressions which are shown in the section 6 comprising a general WENO reconstruction which is not only restricted to the cell boundaries  $x = x_{i+\frac{1}{2}}$ . Thus, they can be used in order to make a WENO reconstruction at an arbitrary point  $x = x_{i+\Delta}$ .

The list for the constants  $c_{rj}$  in uniform grid case is given in the table 3.1 below. The table is given for 3<sup>rd</sup> and 5<sup>th</sup> order WENO cases. For instance, the 5<sup>th</sup> order WENO case constants are the constants for 3<sup>rd</sup> order ENO as well. Complete list for  $k$ , 1 to 6 can be found in the references [1,2].

**Table 3.1** : The constants  $c_{rj}$

| $k$ | $r$ | $j = 0$ | $j = 1$ | $j = 2$ |
|-----|-----|---------|---------|---------|
| 2   | -1  | 3/2     | -1/2    | -       |
|     | 0   | 1/2     | 1/2     | -       |
|     | 1   | -1/2    | 3/2     | -       |
| 3   | -1  | 11/6    | -7/6    | 1/3     |
|     | 0   | 1/3     | 5/6     | -1/6    |
|     | 1   | -1/6    | 5/6     | 1/3     |
|     | 2   | 1/3     | -7/6    | 11/6    |

### 3.3 One Dimensional Conservative Approximation to the Derivatives From Point Values (Finite Difference)

Solving the hyperbolic conservation laws from point values is the second approximation method we will use [1].

Given the point values of a function  $v(x)$ , where  $i = 1, 2, \dots, N$

$$v_i = v(x_i) \quad (3.27)$$

find a numerical flux function for  $i = 0, 1, \dots, N$

$$\hat{v}_{i+\frac{1}{2}} \equiv \hat{v}(v_{i-r}, \dots, v_{i+s}) \quad (3.28)$$

such that the flux difference approximates the derivative  $v'(x)$  to the  $k^{\text{th}}$  order accuracy, which can be shown as:

$$\frac{1}{\Delta x} \left( \hat{v}_{i+\frac{1}{2}} - \hat{v}_{i-\frac{1}{2}} \right) = v'(x_i) + O(\Delta x^k) \quad (3.29)$$

We again assume the boundary (ghost cell) values of the point values are available. The solution of this problem is essential for the high order schemes based on point values.

This problem look quite different from the finite volume case, however, we will see that there is a close relationship between these two. We assume that the grid is uniform,  $\Delta x_i = \Delta x$ . This assumption is, unfortunately, essential in the following development.

If we can find a function  $h(x)$ , which may depend on the grid size  $\Delta x$ , like

$$v(x) \equiv \frac{1}{\Delta x} \int_{x-\Delta x/2}^{x+\Delta x/2} h(\xi) d\xi \quad (3.30)$$

then clearly

$$v'(x) = \frac{1}{\Delta x} \left[ h\left(x + \frac{\Delta x}{2}\right) - h\left(x - \frac{\Delta x}{2}\right) \right] \quad (3.31)$$

hence all we need to do is to use

$$\hat{v}_{i+1/2} = h\left(x_{i+1/2}\right) + O(\Delta x^k) \quad (3.32)$$

to achieve equation (3.29). We note here that it would like a  $O(\Delta x^{k+1})$  term in (3.32) is needed in order to get (3.29), due to the  $\Delta x$  term in the denominator. However, in practice, the  $O(\Delta x^k)$  term in (3.32) is usually smooth, hence the difference in (3.29) would give an extra  $O(\Delta x)$ , just to cancel the one in the denominator.

It is not easy to approximate  $h(x)$  via (3.30), as it is only implicitly defined there. However, we notice that the known function  $v(x)$  is the cell average of the function  $h(x)$ , so to find  $h(x)$  we just need to use the reconstruction procedure described in the previous subsection. If we take the primitive of  $h(x)$

$$H(x) = \int_{-\infty}^x h(\xi) d\xi \quad (3.33)$$

then (3.30) clearly implies

$$H\left(x_{i+\frac{1}{2}}\right) = \sum_{j=-\infty}^i \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} h(\xi) d\xi = \Delta x \sum_{j=-\infty}^i v_j \quad (3.34)$$

Thus, given point values  $v_j$ , we identify them as cell averages of another function  $h(x)$  in (3.30), then the primitive function  $H(x)$  is exactly known at the cell boundaries  $x_{i+\frac{1}{2}}$ . We thus use the same reconstruction procedure described in the previous subsection, to get a  $k^{th}$  order approximation to  $h\left(x_{i+\frac{1}{2}}\right)$ , which is then taken as the numerical flux  $\hat{v}_{i+\frac{1}{2}}$  in (3.28).

In other words, if the stencil for the flux  $\hat{v}_{i+\frac{1}{2}}$  in (3.28) is the following  $k$  points  $(x_{i-r}, \dots, x_{i+s})$  where  $k-1 = r+s$ , then the flux is expressed as the following

$$\hat{v}_{i+\frac{1}{2}} = \sum_{j=0}^{k-1} c_{rj} v_{i-r+j} \quad (3.35)$$

where the constants  $c_{rj}$  are given in table 3.1.

It must be emphasized here again that, unlike in the reconstruction procedure held in the previous subsection, here the grid must be uniform, means all the cell sizes must be the same. Otherwise, it can be proven that no choice of constants  $c_{rj}$  in (3.35) could make the conservative approximation to the derivative (3.29) higher than second order accurate ( $k > 2$ ). The proof is a simple exercise of Taylor expansions. Thus, the high order finite difference (third or higher order) discussed in these notes can apply only to uniform or smoothly varying grids.

As stated earlier in section 3.2, non-uniform grid variables (which are used to calculate non-uniform conservative variables, linear weights, and smoothness indicators) for WENO is calculated and presented in this thesis project, however, these grid dependent variables can only be applicable to finite volume case.

#### 4. ONE DIMENSIONAL ENO and WENO SCHEMES

This section is devoted for explaining the physical and mathematical background of ENO and WENO schemes in detail. Section is organized in the following manner; giving the background of ENO scheme, explaining the finite difference and finite volume versions of WENO scheme (also an alternative way for finite difference version is given in Appendix-A, which simplifies the coding process and gives CPU time advantage), then procedures for implementing these methods to WENO for scalar and vectorial problems are given, and finally time advancing procedure, which is Runge-Kutta, is explained.

In the previous section we mainly discussed the approximation result when the stencil is chosen as (3.9),  $S_i \equiv \{I_{i-r}, \dots, I_{i+s}\}$ . In this section we will concentrate on the issue of how to choose the stencils.

For solving hyperbolic conservation laws, we are interested in the class of piecewise smooth functions. Piecewise smooth function means, they have as many derivatives as the scheme calls for, everywhere except for a finitely many isolated points. At these isolated points, the function  $v(x)$  and its derivatives are assumed to have finite left and right limits.

For such functions which involves discontinuities, the order of accuracy we refer to is “formal”, that is, it is defined by a local truncation error which the scheme calls for in the smooth regions of the function.

The stencils discussed in section 3 can be dealt in a fixed manner as well, means there won't be any choosing algorithm between these candidate stencils. However, while using this kind of fixed approximation, discontinuity in the function will affect the solution whenever the stencil goes beyond a discontinuity with some part of the cells involved in it. The solution, which will be found near a discontinuity will oscillate, means a “Gibbs Phenomena” will occur. As a result, the approximation property (3.6) will no longer be valid in such stencils [1].

#### 4.1 ENO Approximation

The basic idea of ENO is to avoid cells which involve a discontinuity whenever possible. In order to avoid a discontinuity we need to take the Newton form of interpolation polynomial into account. Newton form polynomials and divided differences have already been discussed in section 2.3.2.

Writing equation (2.38) in the new notation used since section 3, will lead us to the following equation (4.1).

$$P(x) = \sum_{j=0}^k V \left[ x_{i-r-1/2}, \dots, x_{i-r+j-1/2} \right] \prod_{m=0}^{j-1} \left( x - x_{i-r+m-1/2} \right) \quad (4.1)$$

Equation (4.1) is a Newton form of the  $k^{th}$  degree interpolation polynomial  $P(x)$ , which interpolates  $V(x)$  at the  $k+1$  points, using the divided difference expressions. We can take the derivative of this equation to get  $p(x)$  in equation (3.17a).

$$p(x) = \sum_{j=1}^k V \left[ x_{i-r-1/2}, \dots, x_{i-r+j-1/2} \right] \sum_{m=0}^{j-1} \prod_{l=0, l \neq m}^{j-1} \left( x - x_{i-r+l-1/2} \right) \quad (4.2)$$

We already knew from equations (2.35) and (2.36) that, divided difference is a measurement of the smoothness of the function inside that stencil.

From now on, we can describe how the ENO idea works. A stencil consisting of  $k+1$  consecutive points, which includes the points  $x_{i-1/2}$  and  $x_{i+1/2}$  can be found, such that  $V(x)$  in this stencil is the smoothest comparing to other possible stencils.

$$V \left[ x_{i-1/2}, x_{i+1/2} \right] = \frac{V(x_{i+1/2}) - V(x_{i-1/2})}{x_{i+1/2} - x_{i-1/2}} = \bar{v}[x_i] = \bar{v}_i \quad (4.3)$$

We perform this job in two steps; in each step we only add one point to the stencil. We thus start with the following two point stencil:

$$\tilde{S}_2(i) = \left\{ x_{i-1/2}, x_{i+1/2} \right\} \quad (4.4)$$

where  $\tilde{S}$  denotes a stencil for the primitive function  $V$ . Also, through equation (4.3) note that the stencil  $\tilde{S}$  has a corresponding stencil  $S$  for  $\bar{v}$ . For instance, expression (4.4) corresponds to a single cell stencil:

$$S(i) = \{I_i\} \quad (4.5)$$

for  $\bar{v}$ . Finally, the linear interpolation on the stencil  $\tilde{S}_2(i)$  in (4.4) can be written in the Newton form as:

$$P^1(x) = V\left[x_{i-\frac{1}{2}}\right] + V\left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right](x - x_{i-\frac{1}{2}}) \quad (4.6)$$

At the second step, we have two choices to expand the current two point stencil, either to add the left neighbor  $x_{i-\frac{3}{2}}$  or the right neighbor  $x_{i+\frac{3}{2}}$ , resulting in the following quadratic interpolations.

$$R(x) = P^1(x) + V\left[x_{i-\frac{3}{2}}, x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right](x - x_{i-\frac{1}{2}})(x - x_{i+\frac{1}{2}}) \quad (4.7a)$$

$$S(x) = P^1(x) + V\left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}, x_{i+\frac{3}{2}}\right](x - x_{i-\frac{1}{2}})(x - x_{i+\frac{1}{2}}) \quad (4.7b)$$

Notice that the equations (4.7a) and (4.7b) use the same function multiplied by two different constants:

$$V\left[x_{i-\frac{3}{2}}, x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right] \text{ and } V\left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}, x_{i+\frac{3}{2}}\right] \quad (4.8)$$

These constants are the two second degree divided differences of  $V(x)$  in two different stencils. As already mentioned, smaller the divided difference smoother the function in that stencil. Thus, this process indicates us the correct place to add the point to current two point stencil. Briefly, we decide which point to add to the stencil, by comparing two relevant divided differences (4.8), and picking the one with the smaller absolute value. Thus, if

$$\left|V\left[x_{i-\frac{3}{2}}, x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]\right| < \left|V\left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}, x_{i+\frac{3}{2}}\right]\right| \quad (4.9)$$

we will take the 3 point stencil as the former, otherwise the latter one:

$$\tilde{S}_3(i) = \{x_{i-3/2}, x_{i-1/2}, x_{i+1/2}\} \quad (4.10a)$$

$$\tilde{S}_3(i) = \{x_{i-1/2}, x_{i+1/2}, x_{i+3/2}\} \quad (4.10b)$$

This procedure, in the stencil to be constructed, can be continued until the desired number of points reached, eventually forming the ENO reconstruction procedure for the desired stencil. In addition to that, for the uniform grid case  $\Delta x_i = \Delta x$ , there is no need to compute divided differences, computing and comparing undivided differences will be sufficient [1].

Finally, in the case of reconstruction from the values in points  $\bar{v}_i$ , note the equation (4.3), the selection procedure for a piecewise-quadratic ENO reconstruction will be as the following figure 4.1 [7, page155].

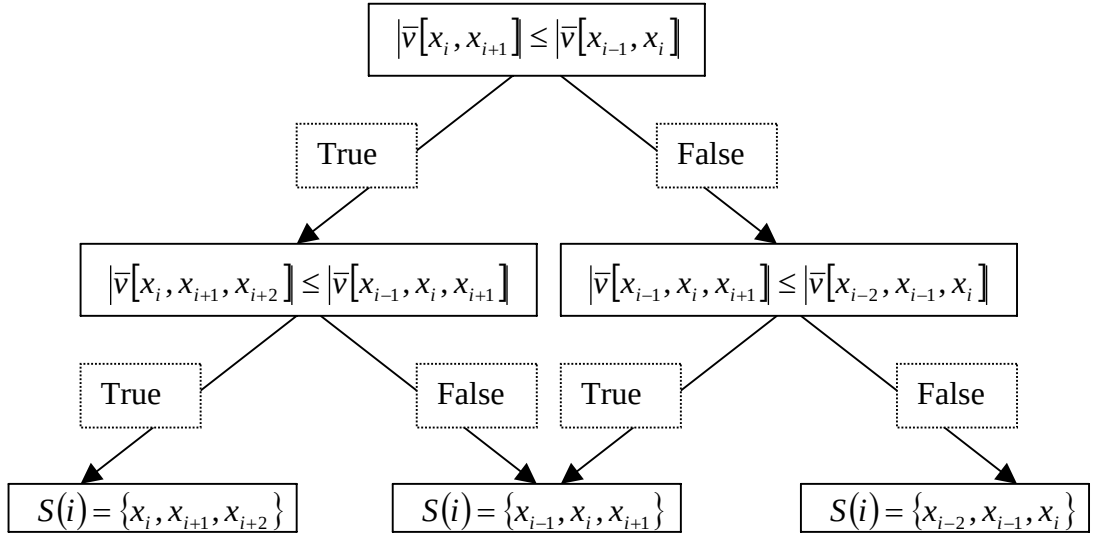


**Figure 4.1 :** Stencil Selection for Piecewise-Quadratic ENO Reconstruction

## 4.2 WENO Approximation and Reconstruction

WENO is based on ENO, it uses the idea of ENO and extends it to a more capable and comprehensive state.

ENO reconstruction is uniformly high order accurate right up to the discontinuity. It achieves this effect by adaptively choosing the stencil based on the

absolute value of divided differences. However, one could make the following remarks about ENO reconstruction, indicating rooms for improvements:

- When both sides of equation (4.9) are near zero, a small change at the round off level would change the direction of the inequality and hence the stencil.
- The resulting flux (3.14) is not smooth, as the stencil pattern may change at neighboring points.
- In the stencil choosing process  $k$  candidate stencils are considered, covering  $2k - 1$  cells, but only one of the candidates is used in the reconstruction (3.11) or the flux (3.32), resulting in  $k^{th}$  order accuracy. However, if all  $2k - 1$  cells could be involved in the reconstruction, one could find  $(2k - 1)^{th}$  order accuracy in smooth regions.
- ENO stencil choosing process involves many logical “if” structures, which are not very efficient on vector supercomputers. Thus, getting rid of those logical choosing processes would be an advantage.

The idea of WENO is the following; instead of using only one of the candidate stencils to form the reconstruction, one uses a convex combination of all of them. Suppose the following  $k$  candidate stencils:

$$S_r(i) = \{x_{i-r}, \dots, x_{i-r+k-1}\}, \quad \text{where } r = 0, \dots, k-1 \quad (4.11)$$

produce  $k$  different reconstructions to the value  $v_{i+\frac{1}{2}}$ , according to (3.13),

$$v_{i+\frac{1}{2}}^{(r)} = \sum_{j=0}^{k-1} c_{rj} \bar{v}_{i-r+j}, \quad \text{where } r = 0, \dots, k-1 \quad (4.12)$$

In figure 4.2, the three candidate stencils presented, which are used to form a  $5^{th}$  order (piece-wise parabolic,  $k = 3$ ) WENO reconstruction at  $x_{i+\frac{1}{2}}$ .

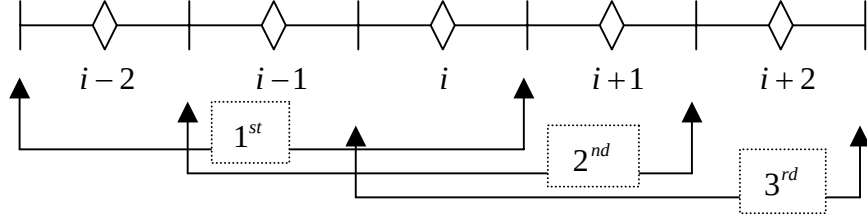


**Figure 4.2 :** Three Candidate Stencils for 5<sup>th</sup> Order WENO Reconstruction

WENO reconstruction would take a convex combination of all  $v_{i+1/2}^{(r)}$  defined in (4.12) as a new approximation to the cell boundary value, thus:

$$v_{i+1/2} = \sum_{r=0}^{k-1} w_r v_{i+1/2}^{(r)} \quad (4.13)$$

Apparently, the key to success of WENO would be the choice of the weights  $w_r$ . We require

$$w_r \geq 0, \quad \sum_{r=0}^{k-1} w_r = 1 \quad (4.14)$$

If the function  $v(x)$  is smooth in all of the candidate stencils (4.11), there are constants  $d_r$  such that

$$v_{i+1/2} = \sum_{r=0}^{k-1} d_r v_{i+1/2}^{(r)} = v\left(x_{i+1/2}\right) + O(\Delta x^{2k-1}) \quad (4.15)$$

For simplicity, in this subsection uniform grid case is assumed,  $\Delta x_i = \Delta x$ . For example,  $d_r$  for  $1 \leq k \leq 3$  are given by

$$d_0 = 1, \quad k = 1;$$

$$d_0 = \frac{2}{3}, \quad d_1 = \frac{1}{3}, \quad k = 2;$$

$$d_0 = \frac{3}{10}, \quad d_1 = \frac{3}{5}, \quad d_2 = \frac{1}{10}, \quad k = 3.$$

It can be seen that  $d_r$  is always positive and, due to consistency:

$$\sum_{r=0}^{k-1} d_r = 1 \quad (4.16)$$

In this smooth case, we would like to have

$$w_r = d_r + O(\Delta x^{k-1}), \quad \text{where } r = 0, \dots, k-1 \quad (4.17)$$

which would imply  $(2k-1)^{th}$  order accuracy:

$$v_{i+\frac{1}{2}} = \sum_{r=0}^{k-1} w_r v_{i+\frac{1}{2}}^{(r)} = v\left(x_{i+\frac{1}{2}}\right) + O(\Delta x^{2k-1}) \quad (4.18)$$

because;

$$\begin{aligned} \sum_{r=0}^{k-1} w_r v_{i+\frac{1}{2}}^{(r)} - \sum_{r=0}^{k-1} d_r v_{i+\frac{1}{2}}^{(r)} &= \sum_{r=0}^{k-1} (w_r - d_r) \left( v_{i+\frac{1}{2}}^{(r)} - v\left(x_{i+\frac{1}{2}}\right) \right) \\ &= \sum_{r=0}^{k-1} O(\Delta x^{k-1}) O(\Delta x^k) \\ &= O(\Delta x^{2k-1}) \end{aligned}$$

where in the first equality we used (4.14) and (4.16).

When the function  $v(x)$  has a discontinuity in one or more stencils (4.11), we would hope that the corresponding weight(s)  $w_r$  to be essentially zero, to emulate the successful ENO idea.

Another consideration is that the weights should be smooth functions of the cell averages involved. In fact, the weights designed in [6] and described below are  $C^\infty$ .

Finally, we would like to have weights, which are computationally efficient. Thus, polynomials or rational functions are preferred over exponential type functions.

All these considerations and ample numerical experiments lead us to the following weight forms:

$$w_r = \frac{\alpha_r}{\sum_{s=0}^{k-1} \alpha_s}, \quad \text{where } r = 0, \dots, k-1 \quad (4.19)$$

$$\alpha_r = \frac{d_r}{(\varepsilon + \beta_r)^2} \quad (4.20)$$

Here  $\varepsilon > 0$  is introduced to avoid the denominator to become zero. We take  $\varepsilon = 10^{-6}$  in all our numerical tests [6]. Smoothness indicators are the  $\beta_r$  in the previous equation of the stencil  $S_r(i)$ , if the function  $v(x)$  is smooth in the stencil  $S_r(i)$ , then

$$\beta_r = O(\Delta x^2)$$

but if  $v(x)$  has a discontinuity inside the stencil  $S_r(i)$ , then

$$\beta_r = O(1)$$

Translating into the weights  $w_r$  in (4.19), when the function  $v(x)$  is smooth in the stencil  $S_r(i)$ :

$$w_r = O(1)$$

and when the function  $v(x)$  has a discontinuity inside stencil  $S_r(i)$ :

$$w_r = O(\Delta x^4)$$

Eventually, emulation of ENO near a discontinuity is thus achieved, also the accuracy requirement which is given in equation (4.17), must be checked when the specific form of smooth indicator  $\beta_r$  is given [1].

Since the weight on a candidate stencil has to vary according to the relative smoothness of this stencil to the other candidate stencils, the way of evaluating the smoothness of a stencil is crucial in the definition of weights.

As we know, on each stencil  $S_r(i)$ , we can construct a  $(k-1)^{th}$  order interpolation polynomial, which is evaluated at  $x = x_{i+\frac{1}{2}}$ , renders the approximation

of  $v\left(x_{i+\frac{1}{2}}\right)$  in equation (4.12). Since the total variation is a good measurement for smoothness, it would be desirable to minimize the total variation for the approximation. Consideration of a smooth flux and of the role of higher order variations leads us to the following measurement for smoothness.

Let the reconstruction polynomial on the stencil  $S_r(i)$  be denoted by  $p_r(x)$ , we define:

$$\beta_r = \sum_{l=1}^{k-1} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \Delta x^{2l-1} \left( \frac{\delta^l p_r(x)}{\delta^l x} \right)^2 dx \quad (4.21)$$

The right hand side of this expression is the sum of the  $L^2$  norms of all the derivatives of the interpolation polynomial  $p_r(x)$  over the interval  $\left(x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right)$ . The term  $\Delta x^{2l-1}$  is used to remove  $\Delta x$  dependency in the derivatives of the polynomials. This is similar but smoother than the total variation measurement based on  $L^1$  norm [6].

When  $k = 2$ , the expression (4.21) gives the following smoothness measurement:

$$\beta_0 = (\bar{v}_{i+1} - \bar{v}_i)^2 \quad (4.22a)$$

$$\beta_1 = (\bar{v}_i - \bar{v}_{i-1})^2 \quad (4.22b)$$

For  $k = 3$ , (4.21) gives the following:

$$\beta_0 = \frac{13}{12}(\bar{v}_i - 2\bar{v}_{i+1} + \bar{v}_{i+2})^2 + \frac{1}{4}(3\bar{v}_i - 4\bar{v}_{i+1} + \bar{v}_{i+2})^2 \quad (4.23a)$$

$$\beta_1 = \frac{13}{12}(\bar{v}_{i-1} - 2\bar{v}_i + \bar{v}_{i+1})^2 + \frac{1}{4}(\bar{v}_{i-1} - \bar{v}_{i+1})^2 \quad (4.23b)$$

$$\beta_2 = \frac{13}{12}(\bar{v}_{i-2} - 2\bar{v}_{i-1} + \bar{v}_i)^2 + \frac{1}{4}(\bar{v}_{i-2} - 4\bar{v}_{i-1} + 3\bar{v}_i)^2 \quad (4.23c)$$

Notice that the discussion here has a one point upwind bias in the optimal linear stencil, which is suitable for a problem with wind blowing from left to right. If the wind blows the other way, the procedure should be modified symmetrically with respect to  $x_{i+\frac{1}{2}}$  [1].

Expressions (4.22) and (4.23) give smoothness indicators for  $3^{rd}$  and  $5^{th}$  order WENO reconstructions respectively. However, they are constructed for the uniform grid case and can only be used when the grid is uniform. Smoothness indicators for  $3^{rd}$  and  $5^{th}$  order WENO reconstructions, which use a structured non-uniform rectangular grid, can be found in section 6.

In summary, the WENO reconstruction procedure is as the following:

#### 4.2.1 Procedure 4.1 - 1D WENO Reconstruction

Given the cell averages  $\{\bar{v}_i\}$  of a function  $v(x)$ , for each cell  $I_i$ , we obtain upwind biased  $(2k-1)^{th}$  order approximations to the function  $v(x)$  at the cell boundaries, which is shown in the figure 3.1 and denoted by  $v_{i-\frac{1}{2}}^+$  and  $v_{i+\frac{1}{2}}^-$ , in the following way:

- Obtain the  $k$  reconstructed values  $v_{i+\frac{1}{2}}^{(r)}$ , of  $k^{th}$  order accuracy, in (4.12), based on the stencils (4.11). Also obtain the  $k$  reconstructed values  $v_{i-\frac{1}{2}}^{(r)}$ , of  $k^{th}$  order accuracy, in (3.11b), based on the stencils (4.11).
- Find the constants, which are called linear weights,  $d_r$  and  $\tilde{d}_r$ , such that (4.15) and

$$v_{i-\frac{1}{2}} = \sum_{r=0}^{k-1} \tilde{d}_r v_{i-\frac{1}{2}}^{(r)} = v\left(x_{i-\frac{1}{2}}\right) + O(\Delta x^{2k-1}) \quad (4.24)$$

are valid. By symmetry,

$$\tilde{d}_r = d_{k-1-r}$$

- Find the smooth indicators  $\beta_r$  in (4.21), for all  $r = 0, \dots, k-1$ .

- Form the weights  $w_r$  and  $\tilde{w}_r$  using (4.19) and (4.20), and

$$\tilde{w}_r = \frac{\tilde{\alpha}_r}{\sum_{s=0}^{k-1} \tilde{\alpha}_s}, \quad \tilde{\alpha}_r = \frac{\tilde{d}_r}{(\varepsilon + \beta_r)^2}, \quad \text{where } r = 0, \dots, k-1$$

- Find the  $(2k-1)^{th}$  order reconstruction by:

$$v_{i+1/2}^- = \sum_{r=0}^{k-1} w_r v_{i+1/2}^{(r)} \quad (4.25a)$$

$$v_{i-1/2}^+ = \sum_{r=0}^{k-1} \tilde{w}_r v_{i-1/2}^{(r)} \quad (4.25b)$$

We can obtain weights for higher orders of  $k$ , using the same recipe. For instance, in [13] finite difference WENO for non-staggered meshes and a variation of WENO based on staggered meshes, which is called Central WENO, is constructed and used up to  $9^{th}$  order of accuracy. Results show that, in both of the variations of the schemes used, the oscillations in component-wise WENO increased with increased order of accuracy. However, by using local characteristic decomposition in the reconstruction, essentially non-oscillatory solutions could be obtained for both of the cases. These high order schemes are useful to simulate problems where shocks and complex smooth region structures coexist, such as the problem of shock interaction with entropy waves.

### 4.3 WENO Scheme for One Dimensional Conservation Laws

One dimensional conservation laws has already discussed in section 2.1. Equation (2.3) can also be written as the following:

$$u_t(x, t) + f_x(u(x, t)) = 0 \quad (4.26)$$

equipped with suitable initial and boundary conditions. In this subsection, spatial discretization will be discussed, and time variable  $t$  will be leaved continuous (method-of-lines) approach. Time discretization will be discussed in section 4.4.

#### 4.3.1 Finite Volume Formulation in the Scalar Case

In finite volume formulations, we deal with the cell averages of the property. We do not solve (4.26) directly, but its integrated version. The equation (4.26) is integrated over the interval  $I_i$  to obtain the following equations:

$$\frac{d\bar{u}(x_i, t)}{dt} = -\frac{1}{\Delta x_i} \left( f(u(x_{i+1/2}, t)) - f(u(x_{i-1/2}, t)) \right) \quad (4.27)$$

where the cell average is

$$\bar{u}(x_i, t) = \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} u(\xi, t) d\xi \quad (4.28)$$

Approximation to the equation (4.27) is accomplished by the following conservative scheme

$$\frac{d\bar{u}_i(t)}{dt} = -\frac{1}{\Delta x_i} \left( \hat{f}_{i+1/2} - \hat{f}_{i-1/2} \right) \quad (4.29)$$

where  $\bar{u}_i(t)$  is the numerical approximation to the cell average  $\bar{u}(x_i, t)$ , and the numerical flux  $\hat{f}_{i+1/2}$  is defined by

$$\hat{f}_{i+1/2} = h(u_{i+1/2}^-, u_{i+1/2}^+) \quad (4.30)$$

with the values  $u_{i+1/2}^\pm$  obtained by the WENO reconstruction, Procedure 4.1.

The function  $h$  in equation (4.30) is a monotone flux. Some examples of monotone fluxes include Godunov flux, Enquist-Osher flux and Lax-Friedrichs flux. Lax-Friedrichs flux is the following [1]:

$$h(a, b) = \frac{1}{2} [f(a) + f(b) - \alpha(b - a)] \quad (4.31)$$

where  $\alpha = \max_u |f'(u)|$  is a constant. The maximum is taken over the relevant range of  $u$  [1] (see page 42 for the discussion on global and local Lax-Friedrichs flux).

In this thesis project, the Lax-Friedrichs flux has been implemented to the generated Fortran code, which is called a monotone flux in scalar case and an approximate Riemann solver in the vectorial case. Lax-Friedrichs flux is the most dissipative (more smearing of discontinuities) of all the three flux types given above. However, smearing differences is becoming much smaller for higher order reconstructions, also it is easy to implement to a code and computationally inexpensive compared to the other two. In addition to that, some other flux types have already been implemented into WENO schemes by various researchers and scientists. Some of them include; HLL, HLLC, MUSTA, FORCE, FLIC, and WAF fluxes. For MUSTA flux corresponding papers of Toro should be investigated. Also, references [8,14] include comparison of some flux types, which are implemented to WENO.

#### 4.3.2 Finite Difference Formulation in the Scalar Case

The first assumption for finite difference case is a uniform grid. Afterwards, equation (4.26) can be solved directly using a conservative approximation to the spatial derivative:

$$\frac{du_i(t)}{dt} = -\frac{1}{\Delta x_i} \left( \hat{f}_{i+1/2} - \hat{f}_{i-1/2} \right) \quad (4.32)$$

where  $u_i(t)$  is the numerical approximation to the point value  $u(x_i, t)$  (not using cell average in this case), and the numerical flux  $\hat{f}_{i+1/2}$  is obtained by the WENO reconstruction, Procedure 4.1, with  $\bar{v}(x) = f(u(x, t))$ , which means we are using flux variables for reconstruction in finite difference case.

In this thesis project for finite difference case Lax-Friedrichs flux splitting is used. The flux splitting is formulated as follows:

$$f(u) = f^+(u) + f^-(u) \quad (4.33)$$

where

$$\frac{df^+(u)}{du} \geq 0, \quad \frac{df^-(u)}{du} \leq 0 \quad (4.34)$$

Lax-Friedrichs flux splitting is:

$$f^\pm(u) = \frac{1}{2}(f(u) \pm \alpha u) \quad (4.35)$$

where  $\alpha = \max_u |f'(u)|$  is a constant. The maximum is taken over the relevant range of  $u$ .

### 4.3.3 WENO in Systems with Component-wise and Characteristic-wise Decompositions

We only consider hyperbolic  $m \times m$  systems, where the Jacobian  $f'(u)$  has  $m$  real eigenvalues (4.36a) and a complete set of independent eigenvectors (4.36b).

$$\lambda_1(u) \leq \dots \leq \lambda_m(u) \quad (4.36a)$$

$$r_1(u), \dots, r_m(u) \quad (4.36b)$$

We denote the matrix whose columns are eigenvectors (4.36b) by

$$R(u) = (r_1(u), \dots, r_m(u)) \quad (4.37)$$

Then clearly

$$R^{-1}(u) f'(u) R(u) = \Lambda(u) \quad (4.38)$$

where  $\Lambda(u)$  is the diagonal matrix with  $\lambda_1(u), \dots, \lambda_m(u)$  on the diagonal. Notice that the rows of  $R^{-1}(u)$ , denoted by  $l_1(u), \dots, l_m(u)$  (row vectors), are left eigenvectors of  $f'(u)$ :

$$l_i(u) f'(u) = \lambda_i(u) l_i(u), \quad i = 1, \dots, m. \quad (4.39)$$

The easiest way to generalize scalar ENO and WENO schemes to systems is to apply a component-by-component procedure. For finite volume formulation, this means that we make the reconstruction using ENO and WENO for each components of  $u$  separately.

In the generated Fortran code for this thesis project, in order to find the numerical flux in systems at the cell boundaries, again Lax-Friedrichs approximate Riemann solver is used. Except that now the constant  $\alpha$  is taken as

$$\alpha = \max_u \max_{1 \leq j \leq m} |\lambda_j(u)| \quad (4.40)$$

For finite difference formulation, a smooth flux splitting (4.33) is again needed. The condition (4.34) now becomes that the two Jacobians

$$\frac{df^+(u)}{du}, \quad \frac{df^-(u)}{du} \quad (4.41)$$

are still diagonalizable, and have only non-negative/non-positive eigenvalues, respectively. The Lax-Friedrichs flux splitting (4.35) is again used, where  $\alpha$  is given by (4.40) [1].

For example, for global Lax-Friedrichs flux and finite volume formulation,  $\alpha$  should be taken as the maximum value of all the three characteristic speeds over whole cells in the direction you are sweeping, however, for finite difference formulation the maximum value of each of the three characteristic speeds should be used in each component of the numerical flux. For local Lax-Friedrichs flux,  $\alpha$  should be computed in a desired number of cells in the sweep direction, i.e. for two point local Lax-Friedrichs flux  $\alpha_i = \max_u |f'(u_i), f'(u_{i+1})|$ .

An alternative to component-wise solution algorithm is the characteristic-wise solution algorithm. Characteristic-wise decompositions work well for more demanding problem cases, or when the order of accuracy is high. Comparison of component-wise and characteristic-wise decompositions in test cases will be given in results and discussion section. Additionally, an introduction to characteristic decomposition has already given in section 2.1.1.

We will consider a simple example where  $f(u) = Au$  in (4.26) is linear and  $A$  is a constant matrix. In this situation, the eigenvalues (4.36a), the eigenvectors (4.36b), and the related matrices  $R$ ,  $R^{-1}$ , and  $\Lambda$  (4.37)-(4.38), are all constant matrices. If we define a change of variable

$$v = R^{-1}u \quad (4.42)$$

then the PDE (4.26) becomes diagonal

$$v_t + \Lambda v_x = 0 \quad (4.43)$$

that is, the  $m$  equations in the previous expression are decoupled, they are independent from other fields from now on, and each one is a scalar linear convection equation in the form below

$$w_t + \lambda_j w_x = 0 \quad (4.44)$$

After projecting to characteristic field, we can use the reconstruction or flux evaluation techniques for the scalar equations, discussed in sections 4.3.1 and 4.3.2, to handle each of the equations in the previous expression. After we do the reconstruction and obtain the results in the characteristic field, we can project back to the physical field and can handle the numerical flux evaluation.

$$u = Rv \quad (4.45)$$

We now come to the situation where  $f'(u)$  is not constant. The trouble is that now all the matrices  $R(u)$ ,  $R^{-1}(u)$ , and  $\Lambda(u)$  are dependent upon  $u$ . We must freeze them locally in order to carry out a similar procedure as in the constant coefficient case. A detailed discussion about this and the Roe averaging process, which is used in the code, is already given in section 2.1.2.

In summary, to build a finite volume (FV) or finite difference (FD) WENO scheme, either scalar or vectorial, and either using component-wise or characteristic-wise decompositions, is given by the following procedures. They are covering merely the related procedures for the code generated for this thesis project, i.e. using only the Lax-Friedrichs flux for finite volume, and the Lax-Friedrichs flux splitting for finite difference schemes. More procedures can be found in the references [1,2].

#### 4.3.4 Procedure 4.2 - FV 1D Scalar WENO Scheme

- Follow the Procedure 4.1, to obtain  $k^{th}$  order reconstructed values  $u_{i+1/2}^-$  and  $u_{i+1/2}^+$  for all  $i$ .

- Compute the monotone flux  $\hat{f}_{i+1/2}$  by (4.31).
- Form the scheme (4.29).

#### 4.3.5 Procedure 4.3 - FD 1D Scalar Flux Splitting WENO Scheme

- Perform the flux splitting (4.35).
- Identify  $\bar{v}_i = f^+(u_i)$  and follow the Procedure 4.1, to obtain the cell boundary values  $v_{i+1/2}^-$  for all  $i$ .
- Take the positive numerical flux as  $\hat{f}_{i+1/2}^+ = v_{i+1/2}^-$ .
- Identify  $\bar{v}_i = f^-(u_i)$  and follow the Procedure 4.1, to obtain the cell boundary values  $v_{i+1/2}^+$  for all  $i$ .
- Take the negative numerical flux as  $\hat{f}_{i+1/2}^- = v_{i+1/2}^+$ .
- Form the numerical flux as  $\hat{f}_{i+1/2} = \hat{f}_{i+1/2}^- + \hat{f}_{i+1/2}^+$ .
- Form the scheme (4.32).

An alternative way for this procedure is to use differences  $da_i = a_{i+1} - a_i$  both for flux and conservative variables. This will reduce coding steps and computations, results in a more speedy algorithm in computers, this variation of finite difference flux splitting is discussed in Appendix-A.

#### 4.3.6 Procedure 4.4 - Component-wise FV 1D Vectorial WENO Scheme

- For each component of the solution  $\bar{u}$ , apply the scalar WENO Procedure 4.1 to reconstruct the corresponding component of the solution at the cell interfaces,  $u_{i+1/2}^\pm$  for all  $i$ .
- Apply the approximate Riemann solver to compute the flux  $\hat{f}_{i+1/2}$  by (4.31).
- Form the scheme (4.29).

#### 4.3.7 Procedure 4.5 - Component-wise FD 1D Vectorial WENO Scheme

- Perform the flux splitting (4.35), with  $\alpha$  given by (4.40).
- For each component of the solution  $u$ , apply the scalar Procedure 4.3 to reconstruct the corresponding component of the numerical flux  $\hat{f}_{i+1/2}$ .
- Form the scheme (4.32).

#### 4.3.8 Procedure 4.6 - Characteristic-wise FV 1D Vectorial WENO Scheme

- Perform the Roe averaging procedure to find an average state  $u_{i+1/2}$ .
- Compute the left and right eigenvectors, and the eigenvalues of the Jacobian  $f'(u_{i+1/2})$ , (4.36)-(4.38).
- Project these conservative variables, which are in the potential stencil of WENO reconstruction, to the characteristic field by using (4.42)

$$\bar{v}_j = R^{-1} \bar{u}_j, \quad j \text{ in a neighborhood of } i.$$

- For each component of the characteristic variable  $\bar{v}$ , apply the scalar WENO Procedure 4.1 to reconstruct the corresponding component of the solution at the cell interfaces,  $v_{i+1/2}^\pm$  for all  $i$ .
- Project back to physical field by using (4.45).

$$u_{i+1/2}^\pm = R v_{i+1/2}^\pm$$

- Apply the approximate Riemann solver to compute the flux  $\hat{f}_{i+1/2}$  by (4.31), with  $\alpha$  given by (4.40).
- Form the scheme (4.29).

#### 4.3.9 Procedure 4.7 - Characteristic-wise FD 1D Vectorial WENO Scheme

- Perform the flux splitting (4.35), with viscosity coefficient  $\alpha$  for the  $l$ -th component is given by  $\alpha = \max_{1 \leq j \leq N} |\lambda_l(u_j)|$ , which means the dissipation is

added in each field according to the maximum size of eigenvalues in that field, not globally.

- Perform the Roe averaging procedure to find an average state  $u_{i+1/2}$ .
- Compute the left and right eigenvectors, and the eigenvalues of the Jacobian  $f'(u_{i+1/2})$ , (4.36)-(4.38).
- Project these fluxes, which are in the potential stencil of WENO reconstruction, to the characteristic field by using (4.42)

$$\hat{g}_j = R^{-1} \hat{f}_j, \quad j \text{ in a neighborhood of } i.$$

- For each component of the characteristic variable  $\hat{g}$ , apply the scalar WENO Procedure 4.1 to reconstruct the corresponding component of the solution  $\hat{g}_{i+1/2}^\pm$  for all  $i$ .
- Project back to physical field by using (4.45).

$$\hat{f}_{i+1/2}^\pm = R \hat{g}_{i+1/2}^\pm$$

- Form the flux by taking

$$\hat{f}_{i+1/2} = \hat{f}_{i+1/2}^- + \hat{f}_{i+1/2}^+$$

- From the scheme (4.32).

#### 4.4 Time Discretization for ENO and WENO Schemes

Up to now, only the spatial discretizations are discussed, leaving the time continuous. In this subsection time discretization is discussed. For time discretization in ENO and WENO schemes, TVD or non-TVD Runge-Kutta procedures, or Lax-Wendroff procedure can be used. In the code generated for this thesis project, a 3<sup>rd</sup> order TVD and a 4<sup>th</sup> order non-TVD Runge-Kutta methods implemented, so these two methods are given in this subsection.

#### 4.4.1 TVD Runge-Kutta Methods

A class of TVD (total variation diminishing) high order Runge-Kutta methods is developed in [3] and further in [15].

These Runge-Kutta methods are used to solve a system of initial value problems of ODEs written as

$$u_t = L(u) \quad (4.46)$$

resulting from a method of lines spatial approximation to a PDE such as

$$u_t = -f(u)_x \quad (4.47)$$

If we assume that a first order Euler forward time stepping:

$$u^{n+1} = u^n + \Delta t L(u^n) \quad (4.48)$$

is stable in a certain norm:

$$\|u^{n+1}\| \leq \|u^n\| \quad (4.49)$$

under a suitable restriction on  $\Delta t$  :

$$\Delta t \leq \Delta t_1 \quad (4.50)$$

then we look for higher order in time Runge-Kutta methods such that the same stability result (4.49) holds, under a perhaps different restriction on  $\Delta t$  :

$$\Delta t \leq c\Delta t_1 \quad (4.51)$$

where  $c$  is termed the CFL coefficient for the high order time discretization.

In [3], a general Runge-Kutta method for (4.46) is written in the form:

$$u^{(i)} = \sum_{k=0}^{i-1} (\alpha_{ik} u^{(k)} + \Delta t \beta_{ik} L(u^{(k)})), \quad i = 1, \dots, m \quad (4.52a)$$

$$u^{(0)} = u^n, \quad u^{(m)} = u^{n+1}. \quad (4.52b)$$

The Runge-Kutta method (4.52a) is TVD under the CFL coefficient (4.51):

$$c = \min_{i,k} \frac{\alpha_{ik}}{\beta_{ik}} \quad (4.53)$$

provided that  $\alpha_{ik} \geq 0$  and  $\beta_{ik} \geq 0$  [1].

Finally, the optimal  $3^{rd}$  order TVD Runge-Kutta is given by [3,15]:

$$u^{(1)} = u^n + \Delta t L(u^n) \quad (4.54a)$$

$$u^{(2)} = \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}) \quad (4.54b)$$

$$u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}) \quad (4.54c)$$

and the  $4^{th}$  order non-TVD Runge-Kutta is given by:

$$u^{(1)} = u^n + \frac{1}{2}\Delta t L(u^n) \quad (4.55a)$$

$$u^{(2)} = u^n + \frac{1}{2}\Delta t L(u^{(1)}) \quad (4.55b)$$

$$u^{(3)} = u^n + \Delta t L(u^{(2)}) \quad (4.55c)$$

$$u^{n+1} = \frac{1}{3} \left( -u^n + u^{(1)} + 2u^{(2)} + u^{(3)} + \frac{1}{2}\Delta t L(u^{(3)}) \right) \quad (4.55d)$$

## 5. TWO DIMENSIONAL APPROXIMATION AND RECONSTRUCTION

In the previous two sections approximation and reconstruction algorithms for one-dimensional space is discussed, and in this section these discussions will be extended to two-dimensional space.

Only the structured decomposition is in this scope of this thesis project, unstructured decomposition is not in the scope so we will not discuss it, however an unstructured decomposition is possible for WENO scheme and related information can be found in references, i.e. [1,2,16].

In this section we will consider Cartesian grids, that is, the domain is rectangle

$$[a,b] \times [c,d] \quad (5.1)$$

covered by cells

$$I_{ij} = \left[ x_{i-1/2}, x_{i+1/2} \right] \times \left[ y_{j-1/2}, y_{j+1/2} \right], \quad 1 \leq i \leq N_x, \quad 1 \leq j \leq N_y \quad (5.2)$$

where

$$a = x_{1/2} < x_{3/2} < \dots < x_{N_x-1/2} < x_{N_x+1/2} = b \quad (5.3a)$$

$$c = y_{1/2} < y_{3/2} < \dots < y_{N_y-1/2} < y_{N_y+1/2} = d \quad (5.3b)$$

The centers of cells and cells sizes are given respectively as the following equations (5.4) and (5.5).

$$x_i = \frac{1}{2} \left( x_{i-1/2} + x_{i+1/2} \right) \quad (5.4a)$$

$$y_j = \frac{1}{2} \left( y_{j-1/2} + y_{j+1/2} \right) \quad (5.4b)$$

$$\Delta x_i = x_{i+1/2} - x_{i-1/2}, \quad i = 1, 2, \dots, N_x \quad (5.5a)$$

$$\Delta y_j = y_{j+1/2} - y_{j-1/2}, \quad j = 1, 2, \dots, N_y \quad (5.5b)$$

## 5.1 WENO Scheme for Two Dimensional Conservation Laws

In this subsection WENO scheme is described for 2D conservation laws, which are described briefly in section 2.2. We will use:

$$u_t(x, y, t) + f_x(u(x, y, t)) + g_y(u(x, y, t)) = 0 \quad (5.6)$$

again equipped with suitable initial and boundary conditions.

We will leave the time continuous (methods-of-lines-approach), as time discretization has already discussed in section 4.4. Third order TVD and fourth order non-TVD Runge-Kutta time discretizations are also sufficient for the two dimensional case.

For structured meshes, the computational domain should be rectangular, given by (5.1). In such cases our grid will be Cartesian, given by (5.2)-(5.5).

For rectangular meshes, we can proceed with using one-dimensional results. For the reconstruction, we first use the one-dimensional WENO reconstruction algorithm, Procedure 4.1, on the two-dimensional cell averages.

For example, to perform 2D structured dimension-by-dimension reconstruction for WENO, firstly Procedure 4.1 should be applied in one direction, say in  $x$  direction, to obtain one-dimensional cell averages in  $x$  only. Then, another reconstruction in the remaining direction, say  $y$  direction, is performed to recover the function itself, again using one-dimensional methodology [1].

### 5.1.1 Finite Volume Formulation in the Scalar Case

In finite volume formulations, we deal with the cell averages of the property. We do not solve (5.6) directly, but its integrated version. The equation (5.6) is integrated over the cell  $I_{ij}$  to obtain the following equations:

$$\begin{aligned} \frac{d\bar{u}_{ij}(t)}{dt} = & -\frac{1}{\Delta x_i \Delta y_j} \left( \int_{y_{j-1/2}}^{y_{j+1/2}} f\left(u\left(x_{i+1/2}, y, t\right)\right) dy - \int_{y_{j-1/2}}^{y_{j+1/2}} f\left(u\left(x_{i-1/2}, y, t\right)\right) dy \right) \\ & - \frac{1}{\Delta x_i \Delta y_j} \left( \int_{x_{i-1/2}}^{x_{i+1/2}} g\left(u\left(x, y_{j+1/2}, t\right)\right) dx - \int_{x_{i-1/2}}^{x_{i+1/2}} g\left(u\left(x, y_{j-1/2}, t\right)\right) dx \right) \end{aligned} \quad (5.7)$$

where the cell average is

$$\bar{u}_{ij}(t) = \frac{1}{\Delta x_i \Delta y_j} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} u(\xi, \eta, t) d\xi d\eta \quad (5.8)$$

Approximation to the equation (5.7) is accomplished by the following conservative scheme

$$\frac{d\bar{u}_{ij}(t)}{dt} = -\frac{1}{\Delta x_i} \left( \hat{f}_{i+1/2,j} - \hat{f}_{i-1/2,j} \right) - \frac{1}{\Delta y_j} \left( \hat{g}_{i,j+1/2} - \hat{g}_{i,j-1/2} \right) \quad (5.9)$$

where the numerical flux  $\hat{f}_{i+1/2,j}$  is defined by

$$\hat{f}_{i+1/2,j} = \sum_{\alpha} \omega_{\alpha} h \left( u_{i+1/2, y_j + \beta_{\alpha} \Delta y_j}^{-}, u_{i+1/2, y_j + \beta_{\alpha} \Delta y_j}^{+} \right) \quad (5.10)$$

where  $\beta_{\alpha}$  and  $\omega_{\alpha}$  are Gaussian quadrature nodes and weights, for approximating the

integration in  $y$ , which is:  $\frac{1}{\Delta y_j} \int_{y_{j-1/2}}^{y_{j+1/2}} f\left(u\left(x_{i+1/2}, y, t\right)\right) dy$ , inside the integral form of

PDE (5.7), and  $u_{i+1/2, y}^{\pm}$  are the  $k^{th}$  order accurate reconstructed values obtained by

WENO reconstruction procedure. The numerical flux  $\hat{g}_{i,j+1/2}$  is defined similarly by

$$\hat{g}_{i,j+1/2} = \sum_{\alpha} \omega_{\alpha} h \left( u_{x_i + \beta_{\alpha} \Delta x_i, j+1/2}^{-}, u_{x_i + \beta_{\alpha} \Delta x_i, j+1/2}^{+} \right) \quad (5.11)$$

for approximating the integration in  $x$ , which is:  $\frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} g\left(u\left(x, y_{j+1/2}, t\right)\right) dx$ , inside the integral form of PDE (5.7), and  $u_{x,j+1/2}^{\pm}$  are the  $k^{th}$  order accurate reconstructed values obtained by WENO reconstruction procedure. The  $h$  is again a one-dimensional monotone flux.

Using 2D finite volume WENO reconstruction for order of accuracy higher than two has some extra costs (order of accuracy below and equal to two doesn't need extra Gaussian point calculations). Because, for example; for a two point Gaussian quadrature point reconstruction, the one-dimensional reconstruction processes should be applied three times, that is, say in  $x$  direction, one from cell centers to cell boundaries, after that two in  $y$  direction to reconstruct the two Gaussian points. So, taking into account the other sweep direction, we have to make six WENO reconstructions for finite volume, however, we have to make only two WENO reconstructions for finite difference in each time step.

Although the cost is high for finite volume WENO, we choose to implement and use finite volume case for the code, because it is conservative in non-uniform grids, and it is one step ahead from finite difference case in the way to an unstructured case.

The Gaussian quadrature approach for piece-wise quadratic WENO reconstruction will be discussed in detail in the following section 5.2.

### 5.1.2 Finite Difference Formulation in the Scalar Case

Here we assume a uniform grid and solve (5.6) directly using a conservative approximation to the spatial derivative:

$$\frac{du_{ij}(t)}{dt} = -\frac{1}{\Delta x} \left( \hat{f}_{i+1/2,j} - \hat{f}_{i-1/2,j} \right) - \frac{1}{\Delta y} \left( \hat{g}_{i,j+1/2} - \hat{g}_{i,j-1/2} \right) \quad (5.12)$$

where  $u_{ij}(t)$  is the numerical approximation to the point value  $u(x_i, y_j, t)$ .

The numerical flux  $\hat{f}_{i+1/2,j}$  is obtained by one-dimensional WENO algorithm, Procedure 4.1, with  $v(x) = u(x, y_j, t)$  and with  $j$  fixed, and the numerical flux

$\hat{g}_{i,j+\frac{1}{2}}$  is obtained by one-dimensional WENO algorithm, Procedure 4.1, with  $v(y) = u(x_i, y, t)$  and with  $i$  fixed. In addition to that, Lax-Friedrichs flux splitting can be applied here dimension-by-dimension.

As stated before, in terms of operation count, the finite difference WENO scheme is less than the finite volume counterpart of the same order [1]. This is because we don't need any WENO reconstruction for the Gaussian points in the other direction to obtain the high order of accuracy. But the disadvantage of finite difference WENO scheme is the restriction to use a uniform grid.

### 5.1.3 WENO in Systems

Implementing 2D WENO in vectorial problems is the same as implementing the 1D one. The advice here that, when the fluxes are computed along a cell boundary a one-dimensional local characteristic decomposition normal to the boundary is performed. The monotone flux is replaced with a one-dimensional approximate Riemann solver (also for flux computations exact solvers can be used but this is not in the scope of this thesis project). Thus, the discussion in section 4.3.3 can be applied here.

### 5.1.4 Procedure 5.1 - FV 2D WENO Scheme

- Follow the corresponding one-dimensional finite volume procedures, which are given in section 4, up to the reconstruction of the solution at cell interfaces.
- Change the sweep direction, and for each Gaussian integration point perform one-dimensional WENO reconstructions to the values found in the previous step.
- Compute the numerical flux at the cell interfaces by (5.10) or (5.11) for each Gaussian point (without summing), then use Gaussian integration rule to find the corresponding flux for that cell boundary.
- Do the computations for the other sweep direction (from step one to three again).
- Form the scheme (5.9).

### 5.1.5 Procedure 5.2 - FD 2D WENO Scheme

- Follow the corresponding one dimensional finite difference procedures, which are given in section 4, in order to find numerical fluxes at cell interfaces.
- Perform the previous step again in the other sweep direction.
- Form the scheme (5.12).

## 5.2 Gaussian Integration Point Rule for WENO Scheme

This subsection discusses and describes the implementation of two point Gaussian integration point rule to WENO scheme.

As stated before, if the order of accuracy is more than two, one needs more than one point for the numerical flux calculation at the cell boundary for the finite volume case. Thus, to achieve a piece-wise parabolic ( $5^{th}$  order) WENO, which is used in the code, we have to increase the number of reconstruction points at the cell boundaries.

Reference [17] suggests to use a two point Gaussian integration point for piece-wise parabolic reconstruction. They state that, their numerical experiments show that the best result in terms of accuracy and computational cost for piece-wise parabolic  $k = 3$ , and piece-wise cubic  $k = 4$ , are obtained if the following two point ( $4^{th}$  order) Gaussian quadrature is used:

$$\int_{-1}^1 \phi(\xi) d\xi = \phi\left(-\frac{1}{\sqrt{3}}\right) + \phi\left(+\frac{1}{\sqrt{3}}\right) \quad (5.13)$$

even though the use of (5.13) leads to formal fourth order spatial accuracy.

Figure 5.1 shows the usage of two point Gaussian integration point (GIP) rule for WENO reconstruction;  $1^{st}$  sweep direction from cell centers to cell boundary centers and then  $2^{nd}$  sweep direction from cell boundary centers to Gaussian integration points. These points locate  $\frac{\Delta\eta}{2\sqrt{3}}$  away from cell boundary centers, if cell boundary has a length of  $\Delta\eta$ .

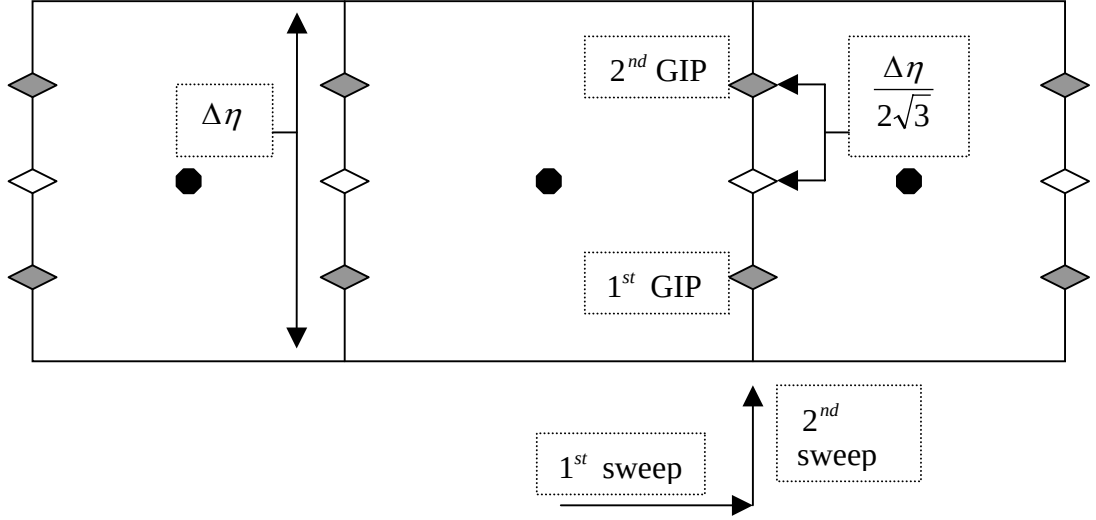


**Figure 5.1 :** Usage of Two Point Gaussian Integration Point Rule for WENO

### 5.3 Piece-wise Parabolic WENO Reconstruction ( $5^{th}$ Order)

This subsection prepared for giving an overview for reconstructing a piece-wise parabolic polynomial ( $k = 3$ ) for finite difference and finite volume WENO scheme, which uses uniform grid. Information to build a  $5^{th}$  order WENO has already given throughout the text, in this subsection we collect these information together. In addition to that, we will present the new constants for two point Gaussian integration point rule for a uniform grid case. The constants to use, in non-uniform grid case, will be discussed in section 6.

Consider a cell  $I_i = [\xi_{i-1/2}, \xi_{i+1/2}]$ , the three candidate stencils for the reconstruction, which shown in figure 4.2 are:

$$S_0 = \{i, i+1, i+2\}, \quad S_1 = \{i-1, i, i+1\}, \quad S_2 = \{i-2, i-1, i\} \quad (5.14)$$

and the corresponding smoothness indicators for these stencils given in equation (4.23). The optimal (linear) weights for left extrapolated values  $u_{i+1/2}^-$ , see figure 3.1, are:

$$d_0 = \frac{3}{10}, \quad d_1 = \frac{3}{5}, \quad d_2 = \frac{1}{10} \quad (5.15)$$

then, the reconstruction on the left side of the cell boundary is, see table 3.1,

$$\begin{aligned}
u_{i+1/2}^- &= u\left(\xi_{i+1/2} - 0\right) \\
&= \frac{1}{6}w_0(-u_{i+2} + 5u_{i+1} + 2u_i) + \frac{1}{6}w_1(2u_{i+1} + 5u_i - u_{i-1}) \\
&\quad + \frac{1}{6}w_2(11u_i - 7u_{i-1} + 2u_{i-2})
\end{aligned} \tag{5.16}$$

The optimal (linear) weights for right extrapolated values  $u_{i-1/2}^+$ , are:

$$d_0 = \frac{1}{10}, \quad d_1 = \frac{3}{5}, \quad d_2 = \frac{3}{10} \tag{5.17}$$

then, the reconstruction on the left side of the cell boundary is,

$$\begin{aligned}
u_{i-1/2}^+ &= u\left(\xi_{i-1/2} + 0\right) \\
&= \frac{1}{6}w_0(2u_{i+2} - 7u_{i+1} + 11u_i) + \frac{1}{6}w_1(-u_{i+1} + 5u_i + 2u_{i-1}) \\
&\quad + \frac{1}{6}w_2(2u_i + 5u_{i-1} - u_{i-2})
\end{aligned} \tag{5.18}$$

For the first Gaussian integration point  $\xi = \xi_i - \frac{\Delta\xi}{2\sqrt{3}}$  the optimal weights are

$$d_0 = \frac{210 - \sqrt{3}}{1080}, \quad d_1 = \frac{11}{18}, \quad d_2 = \frac{210 + \sqrt{3}}{1080} \tag{5.19}$$

and the reconstructed value is given by

$$\begin{aligned}
u\left(\xi_i - \frac{\Delta\xi}{2\sqrt{3}}\right) &= w_0\left[u_i + (3u_i - 4u_{i+1} + u_{i+2})\frac{\sqrt{3}}{12}\right] + w_1\left[u_i - (-u_{i-1} + u_{i+1})\frac{\sqrt{3}}{12}\right] \\
&\quad + w_2\left[u_i - (3u_i - 4u_{i-1} + u_{i-2})\frac{\sqrt{3}}{12}\right]
\end{aligned} \tag{5.20}$$

For the second Gaussian integration point  $\xi = \xi_i + \frac{\Delta\xi}{2\sqrt{3}}$  the optimal weights are obtained from symmetry, which are given by:

$$d_0 = \frac{210 + \sqrt{3}}{1080}, \quad d_1 = \frac{11}{18}, \quad d_2 = \frac{210 - \sqrt{3}}{1080} \quad (5.21)$$

and the reconstructed value is given by:

$$\begin{aligned} u\left(\xi_i + \frac{\Delta\xi}{2\sqrt{3}}\right) = & w_0 \left[ u_i - (3u_i - 4u_{i+1} + u_{i+2}) \frac{\sqrt{3}}{12} \right] + w_1 \left[ u_i + (-u_{i-1} + u_{i+1}) \frac{\sqrt{3}}{12} \right] \\ & + w_2 \left[ u_i - (-3u_i + 4u_{i-1} - u_{i-2}) \frac{\sqrt{3}}{12} \right] \end{aligned} \quad (5.22)$$

Note that the nonlinear weights  $w_r$  must be computed according to equation (4.19) separately for each of the points  $\xi = \xi_i \pm \frac{\Delta\xi}{2\sqrt{3}}$  [17].

## 6. WENO SCHEME FOR NON-UNIFORM GRIDS

This section gives all the necessary information to build a piece-wise linear ( $3^{rd}$  order), and piece-wise parabolic ( $5^{th}$  order) WENO reconstruction for non-uniform, rectangular, structured meshes.

In reference [18] non-uniform grids had already investigated, but they gave results merely for the cell boundaries. In this section, the results for the piece-wise parabolic and piece-wise linear WENO are given for an arbitrary point in the line along a 1D cell. This is because, the Gaussian integration point rule, which is discussed in the previous section, requires reconstructions in some number of points inside the cell (in the interval along the boundary for 2D). Hence, the ability to make a reconstruction in an arbitrary point along the cell boundary is vital. In summary, you can use these expressions below to reconstruct piece-wise parabolic polynomials for Gaussian integration rule, which uses any number of point for integration.

Suppose that the grid used in figure 3.1 is now non-uniform, like the figure 6.1 given below.

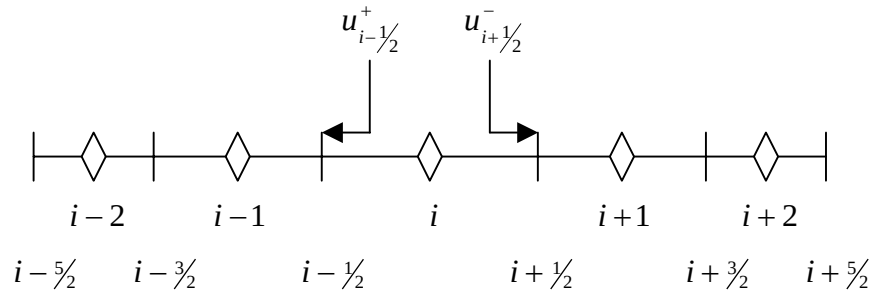


**Figure 6.1 : A Non-Uniform Grid for the Stencil**

We have to construct a Lagrange interpolation polynomial around  $x = x_i$ , the interpolation polynomial is already given in (3.23), but let's write it in another notation, which is given in [18].

$$u_r(x)|_i = \sum_{j=0}^{k-1} \left[ \frac{\sum_{m=j+1}^k \prod_{p=0, p \neq m}^k \left( x - x_{i-r+q-\frac{1}{2}} \right)}{\prod_{p=0, p \neq m}^k \left( x_{i-r+m-\frac{1}{2}} - x_{i-r+p-\frac{1}{2}} \right)} \right] \left( x_{i-r+j+\frac{1}{2}} - x_{i-r+j-\frac{1}{2}} \right) \bar{u}_{i-r+j} \quad (6.1)$$

Here  $r = 0, \dots, k-1$  indicates the stencil  $\{x_{i-r-\frac{1}{2}} \dots x_{i-r+k-\frac{1}{2}}\}$  used for the construction of the polynomial and  $k$  indicates the order of accuracy of the polynomial.

We can find the linear (optimal) weights by assuming that we are in a smooth region, in addition, the nonlinear weights should be equal to linear weights in that smooth region. Thus, the higher order polynomial,  $(2k-1)^{th}$ , is constructed from a linear combination of some lower order,  $(k)^{th}$ , polynomials.

$$u_{i+\frac{1}{2}, (2k-1)}^- = \sum_{r=0}^{k-1} d_{i+\frac{1}{2}, (k)}^- u_{i+\frac{1}{2}, (k)}^- \quad (6.2)$$

and from equation (4.16),  $\sum_{r=0}^{k-1} d_r = 1$ .

The smoothness indicators  $\beta_r$  are the sum of the  $L_2$ -norm of all derivatives of the interpolation polynomial  $u_r(x)|_i$ , over the interval  $(x_{i-\frac{1}{2}} \dots x_{i+\frac{1}{2}})$ , defined as

$$\beta_{i,r} = \sum_{m=1}^{k-1} \left[ \left( x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}} \right)^{2m-1} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left( \frac{\partial^m (u_r(x)|_i)}{\partial^m (x)} \right)^2 dx \right] \quad (6.3)$$

## 6.1 Piece-wise Parabolic WENO for Non-Uniform Grids ( $5^{th}$ Order)

In order to reconstruct a piece-wise parabolic WENO from three parabolic reconstructions,  $k = 3$ , one has to solve the linear system given below.

$$\begin{bmatrix} 0 & 0 & u_{i-2,r=2,(k)} \\ 0 & u_{i-1,r=1,(k)} & u_{i-1,r=2,(k)} \\ u_{i,r=0,(k)} & u_{i,r=1,(k)} & u_{i,r=2,(k)} \\ u_{i+1,r=0,(k)} & u_{i+1,r=1,(k)} & 0 \\ u_{i+2,r=0,(k)} & 0 & 0 \end{bmatrix}_{5 \times 3} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix}_{3 \times 1} = \begin{bmatrix} u_{i-2,r=2,(2k-1)} \\ u_{i-1,r=2,(2k-1)} \\ u_{i,r=2,(2k-1)} \\ u_{i+1,r=2,(2k-1)} \\ u_{i+2,r=2,(2k-1)} \end{bmatrix}_{5 \times 1} \quad (6.4)$$

The three parabolic ( $3^{rd}$  order) reconstructions for three candidate stencils, see figure 4.2, around  $x = x_i$  lead to the following expressions, and using (6.5) for the notation:

$$a = x_{i+5/2}, \quad b = x_{i+3/2}, \quad c = x_{i+1/2}, \quad d = x_{i-1/2}, \quad e = x_{i-3/2}, \quad f = x_{i-5/2} \quad (6.5)$$

- For  $r = 0$ :

$$A_1 = \frac{(x-b)(x-a) + (x-d)(x-a) + (x-d)(x-b)}{(c-d)(c-b)(c-a)} \quad (6.6a)$$

$$A_2 = \frac{(x-c)(x-a) + (x-d)(x-a) + (x-d)(x-c)}{(b-d)(b-c)(b-a)} \quad (6.6b)$$

$$A_3 = \frac{(x-c)(x-b) + (x-d)(x-b) + (x-d)(x-c)}{(a-d)(a-c)(a-b)} \quad (6.6c)$$

$$\begin{aligned} u_0(x)|_i &= (A_1 + A_2 + A_3) \left( x_{i+1/2} - x_{i-1/2} \right) \bar{u}_i \\ &\quad + (A_2 + A_3) \left( x_{i+3/2} - x_{i+1/2} \right) \bar{u}_{i+1} \\ &\quad + A_3 \left( x_{i+5/2} - x_{i+3/2} \right) \bar{u}_{i+2} \end{aligned} \quad (6.7)$$

- For  $r = 1$ :

$$A_1 = \frac{(x-c)(x-b) + (x-e)(x-b) + (x-e)(x-c)}{(d-e)(d-c)(d-b)} \quad (6.8a)$$

$$A_2 = \frac{(x-d)(x-b) + (x-e)(x-b) + (x-e)(x-d)}{(c-e)(c-d)(c-b)} \quad (6.8b)$$

$$A_3 = \frac{(x-d)(x-c) + (x-e)(x-c) + (x-e)(x-d)}{(b-e)(b-d)(b-c)} \quad (6.8c)$$

$$\begin{aligned} u_1(x)|_i = & (A_1 + A_2 + A_3) \left( x_{i-1/2} - x_{i-3/2} \right) \bar{u}_{i-1} \\ & + (A_2 + A_3) \left( x_{i+1/2} - x_{i-1/2} \right) \bar{u}_i \\ & + A_3 \left( x_{i+3/2} - x_{i+1/2} \right) \bar{u}_{i+1} \end{aligned} \quad (6.9)$$

- For  $r = 2$ :

$$A_1 = \frac{(x-d)(x-c) + (x-f)(x-c) + (x-f)(x-d)}{(e-f)(e-d)(e-c)} \quad (6.10a)$$

$$A_2 = \frac{(x-e)(x-c) + (x-f)(x-c) + (x-f)(x-e)}{(d-f)(d-e)(d-c)} \quad (6.10b)$$

$$A_3 = \frac{(x-e)(x-d) + (x-f)(x-d) + (x-f)(x-e)}{(c-f)(c-e)(c-d)} \quad (6.10c)$$

$$\begin{aligned} u_2(x)|_i = & (A_1 + A_2 + A_3) \left( x_{i-3/2} - x_{i-5/2} \right) \bar{u}_{i-2} \\ & + (A_2 + A_3) \left( x_{i-1/2} - x_{i-3/2} \right) \bar{u}_{i-1} \\ & + A_3 \left( x_{i+1/2} - x_{i-1/2} \right) \bar{u}_i \end{aligned} \quad (6.11)$$

The 5<sup>th</sup> order reconstruction around  $x = x_i$  with  $r = 2$  leads to the following expression, using (6.5):

$$\begin{aligned} & (x-d)(x-c)(x-b)(x-a) \\ & + (x-f)(x-c)(x-b)(x-a) \\ & + (x-f)(x-d)(x-b)(x-a) \\ & + (x-f)(x-d)(x-c)(x-a) \\ & + (x-f)(x-d)(x-c)(x-b) \\ A_1 = & \frac{(x-f)(x-d)(x-c)(x-b)}{(e-f)(e-d)(e-c)(e-b)(e-a)} \end{aligned} \quad (6.12a)$$

$$\begin{aligned}
& (x-e)(x-c)(x-b)(x-a) \\
& + (x-f)(x-c)(x-b)(x-a) \\
& + (x-f)(x-e)(x-b)(x-a) \\
& + (x-f)(x-e)(x-c)(x-a) \\
& + (x-f)(x-e)(x-c)(x-b) \\
A_2 = & \frac{(d-f)(d-e)(d-c)(d-b)(d-a)}{(d-f)(d-e)(d-c)(d-b)(d-a)}
\end{aligned} \tag{6.12b}$$

$$\begin{aligned}
& (x-e)(x-d)(x-b)(x-a) \\
& + (x-f)(x-d)(x-b)(x-a) \\
& + (x-f)(x-e)(x-b)(x-a) \\
& + (x-f)(x-e)(x-d)(x-a) \\
& + (x-f)(x-e)(x-d)(x-b) \\
A_3 = & \frac{(c-f)(c-e)(c-d)(c-b)(c-a)}{(c-f)(c-e)(c-d)(c-b)(c-a)}
\end{aligned} \tag{6.12c}$$

$$\begin{aligned}
& (x-e)(x-d)(x-c)(x-a) \\
& + (x-f)(x-d)(x-c)(x-a) \\
& + (x-f)(x-e)(x-c)(x-a) \\
& + (x-f)(x-e)(x-d)(x-a) \\
& + (x-f)(x-e)(x-d)(x-c) \\
A_4 = & \frac{(b-f)(b-e)(b-d)(b-c)(b-a)}{(b-f)(b-e)(b-d)(b-c)(b-a)}
\end{aligned} \tag{6.12d}$$

$$\begin{aligned}
& (x-e)(x-d)(x-c)(x-b) \\
& + (x-f)(x-d)(x-c)(x-b) \\
& + (x-f)(x-e)(x-c)(x-b) \\
& + (x-f)(x-e)(x-d)(x-b) \\
& + (x-f)(x-e)(x-d)(x-c) \\
A_5 = & \frac{(a-f)(a-e)(a-d)(a-c)(a-b)}{(a-f)(a-e)(a-d)(a-c)(a-b)}
\end{aligned} \tag{6.12e}$$

$$\begin{aligned}
u_2(x)|_i = & (A_1 + A_2 + A_3 + A_4 + A_5) \left( x_{i-3/2} - x_{i-5/2} \right) \bar{u}_{i-2} \\
& + (A_2 + A_3 + A_4 + A_5) \left( x_{i-1/2} - x_{i-3/2} \right) \bar{u}_{i-1} \\
& + (A_3 + A_4 + A_5) \left( x_{i+1/2} - x_{i-1/2} \right) \bar{u}_i \\
& + (A_4 + A_5) \left( x_{i+3/2} - x_{i+1/2} \right) \bar{u}_{i+1} \\
& + A_5 \left( x_{i+5/2} - x_{i+3/2} \right) \bar{u}_{i+2}
\end{aligned} \tag{6.13}$$

Solving the linear system (6.4) lead us to the following linear (optimal) weights, (system is solved by Mathematica):

$$\begin{aligned}
& (x-b)(x-c)(x-d)(x-e) \\
& + (x-b)(x-c)(x-d)(x-f) \\
& + (x-b)(x-c)(x-e)(x-f) \\
& + (x-b)(x-d)(x-e)(x-f) \\
& + (x-c)(x-d)(x-e)(x-f) \\
d_0 = & \frac{(x-b)(x-c)(x-d)(x-e) + (x-b)(x-c)(x-d)(x-f) + (x-b)(x-c)(x-e)(x-f) + (x-b)(x-d)(x-e)(x-f) + (x-c)(x-d)(x-e)(x-f)}{(a-e)(a-f)[(x-b)(x-c) + (x-b)(x-d) + (x-c)(x-d)]}
\end{aligned} \tag{6.14a}$$

$$\begin{aligned}
& (x-a)(x-b)(x-c)(x-d) \\
& + (x-a)(x-b)(x-c)(x-e) \\
& + (x-a)(x-b)(x-d)(x-e) \\
& + (x-a)(x-c)(x-d)(x-e) \\
& + (x-b)(x-c)(x-d)(x-e) \\
d_2 = & \frac{(x-a)(x-b)(x-c)(x-d) + (x-a)(x-b)(x-c)(x-e) + (x-a)(x-b)(x-d)(x-e) + (x-a)(x-c)(x-d)(x-e) + (x-b)(x-c)(x-d)(x-e)}{(a-f)(b-f)[(x-c)(x-d) + (x-c)(x-e) + (x-d)(x-e)]}
\end{aligned} \tag{6.14b}$$

$$d_1 = 1 - (d_0 + d_2) \tag{6.14c}$$

The smoothness indicators, from expression (6.3), for a piecewise-parabolic WENO reconstruction are found to be:

$$\begin{aligned}
\beta_0 = & 36(c-d) \frac{(c-d)^3 ((a-c)u_i - ((a-c) + (b-d))u_{i+1} + (b-d)u_{i+2})^2}{(a-c)^2 (a-d)^2 (b-d)^2} \\
& + 36(c-d) \frac{(((a-c)((a-d) + (b-d) + (c-d))u_i - (((a-d) + (b-d))^2 - (c-d)^2 - (b-d)(a-d))u_{i+1} + (b-d)((b-d) + (c-d))u_{i+2})^3}{81(a-c)^2 (a-d)^2 (b-d)^2 ((a-c)u_i - ((a-c) + (b-d))u_{i+1} + (b-d)u_{i+2})} \\
& - 36(c-d) \frac{(((a-c)((a-c) + (b-c))u_i - (((a-c) + (b-c))^2 - (c-d)^2 - (b-c)(a-c))u_{i+1} + (b-d)((b-c) - (c-d))u_{i+2})^3}{81(a-c)^2 (a-d)^2 (b-d)^2 ((a-c)u_i - ((a-c) + (b-d))u_{i+1} + (b-d)u_{i+2})} \quad (6.15a)
\end{aligned}$$

$$\begin{aligned}
\beta_1 = & 36(c-d) \frac{(c-d)^3 ((b-d)u_{i-1} - ((b-d) + (c-e))u_i + (c-e)u_{i+1})^2}{(b-d)^2 (b-e)^2 (c-e)^2} \\
& + 36(c-d) \frac{(((b-d)((b-d) + (c-d))u_{i-1} - (((b-d) + (c-d))^2 - (d-e)^2 - (c-d)(b-d))u_i + (c-e)((c-d) - (d-e))u_{i+1})^3}{81(b-d)^2 (b-e)^2 (c-e)^2 ((b-d)u_{i-1} - ((b-d) + (c-e))u_i + (c-e)u_{i+1})} \\
& - 36(c-d) \frac{(((b-d)((b-c) - (c-d))u_{i-1} + (((c-d) + (c-e))^2 - (b-c)^2 - (c-e)(c-d))u_i - (c-e)((c-d) + (c-e))u_{i+1})^3}{81(b-d)^2 (b-e)^2 (c-e)^2 ((b-d)u_{i-1} - ((b-d) + (c-e))u_i + (c-e)u_{i+1})} \quad (6.15b)
\end{aligned}$$

$$\begin{aligned}
\beta_2 = & 36(c-d) \frac{(c-d)^3 ((c-e)u_{i-2} - ((c-e) + (d-f))u_{i-1} + (d-f)u_i)^2}{(c-e)^2 (c-f)^2 (d-f)^2} \\
& + 36(c-d) \frac{(((c-e)((c-d) - (d-e))u_{i-2} + (((d-e) + (d-f))^2 - (c-d)^2 - (d-e)(d-f))u_{i-1} - (d-f)((d-e) + (d-f))u_i)^3}{81(c-e)^2 (c-f)^2 (d-f)^2 ((c-e)u_{i-2} - ((c-e) + (d-f))u_{i-1} + (d-f)u_i)} \\
& + 36(c-d) \frac{(((c-e)((c-d) + (c-e))u_{i-2} - (((c-e) + (c-f))^2 - (c-d)^2 - (c-e)(c-f))u_{i-1} + (d-f)((c-d) + (c-e) + (c-f))u_i)^3}{81(c-e)^2 (c-f)^2 (d-f)^2 ((c-e)u_{i-2} - ((c-e) + (d-f))u_{i-1} + (d-f)u_i)} \quad (6.15c)
\end{aligned}$$

## 6.2 Piece-wise Linear WENO for Non-Uniform Grids ( $3^{rd}$ Order)

In order to reconstruct a piece-wise linear WENO from two linear reconstructions,  $k = 2$ , one has to solve the linear system given below.

$$\begin{bmatrix} 0 & u_{i-1,r=1,(k)} \\ u_{i,r=0,(k)} & u_{i,r=1,(k)} \\ u_{i+1,r=0,(k)} & 0 \end{bmatrix}_{3 \times 2} \begin{bmatrix} d_0 \\ d_1 \end{bmatrix}_{2 \times 1} = \begin{bmatrix} u_{i-1,r=1,(2k-1)} \\ u_{i,r=1,(2k-1)} \\ u_{i+1,r=1,(2k-1)} \end{bmatrix}_{3 \times 1} \quad (6.16)$$

The two linear ( $2^{nd}$  order) reconstructions for two candidate stencils, around  $x = x_i$  lead to the following expressions, again using (6.5):

- For  $r = 0$ :

$$A_1 = -\left( \frac{(x-b) + (x-d)}{(b-c)(c-d)} \right) \quad (6.17a)$$

$$A_2 = \frac{(x-c) + (x-d)}{(b-c)(b-d)} \quad (6.17b)$$

$$\begin{aligned} u_0(x)|_i &= (A_1 + A_2) \left( x_{i+1/2} - x_{i-1/2} \right) \bar{u}_i \\ &\quad + A_2 \left( x_{i+3/2} - x_{i+1/2} \right) \bar{u}_{i+1} \end{aligned} \quad (6.18)$$

- For  $r = 1$ :

$$A_1 = -\left( \frac{(x-c) + (x-e)}{(c-d)(d-e)} \right) \quad (6.19a)$$

$$A_2 = \frac{(x-d) + (x-e)}{(c-d)(c-e)} \quad (6.19b)$$

$$\begin{aligned} u_1(x)|_i &= (A_1 + A_2) \left( x_{i-1/2} - x_{i-3/2} \right) \bar{u}_{i-1} \\ &\quad + A_2 \left( x_{i+1/2} - x_{i-1/2} \right) \bar{u}_i \end{aligned} \quad (6.20)$$

Solving the linear system (6.16) lead us to the following linear (optimal) weights, (system is solved by Mathematica):

$$d_0 = \frac{(x-c)(x-d) + (x-c)(x-e) + (x-d)(x-e)}{(b-e) + ((x-c) + (x-d))} \quad (6.21a)$$

$$d_1 = -\left( \frac{(x-b)(x-c) + (x-b)(x-d) + (x-c)(x-d)}{(b-e) + ((x-c) + (x-d))} \right) \quad (6.21b)$$

or

$$d_1 = 1 - d_0 \quad (6.21c)$$

The smoothness indicators, from expression (6.3), for a piecewise-linear WENO reconstruction are found to be:

$$\beta_0 = 4 \left( \frac{(c-d)(u_i - u_{i+1})}{(b-d)} \right)^2 \quad (6.22a)$$

$$\beta_1 = 4 \left( \frac{(c-d)(u_{i-1} - u_i)}{(c-e)} \right)^2 \quad (6.22b)$$

Consequently, with the help of these equations given in this section, and for a provided coordinate  $x$  around  $x_i$ , one can construct the polynomials required for piece-wise linear and piece-wise parabolic, structured, Cartesian, rectangular, WENO reconstruction. For instance, to find reconstructions in the cell boundaries, put  $x = x_{i-1/2}$  then  $x = x_{i+1/2}$ , or to find reconstructions in the two point Gaussian

points, put  $x = x_i - \frac{\Delta x}{2\sqrt{3}}$  then  $x = x_i + \frac{\Delta x}{2\sqrt{3}}$ .

## 7. BOUNDARY CONDITIONS

This section is briefly discussing the implementation of boundary conditions. Boundary conditions can be set in various ways, and one can find a lot of different approaches in the literature for this issue. Thus, honestly speaking choosing an appropriate way for any kind of boundary conditions and implementing it is not the easiest part of writing a code, but of course not the most challenging part. In addition, different ways you choose for a boundary condition can lead you different results, so extreme care must be given for it.

In the code, we choose to implement boundary conditions via ghost cells approach. This approach is discussed briefly in the next subsection.

### 7.1 Ghost (Dummy) Cells Approach

In the case of implementing a boundary condition, one approach is to develop special formulas for use near the boundaries, which will depend both on what type of boundary conditions are specified and on what sort of method you are trying to match. However, in general it is much easier to think of extending the computational domain to include a few additional cells on either end, called ghost cells, whose values are set at the beginning of each time step in some manner that depends on the boundary conditions and perhaps the interior solution [19, page129].

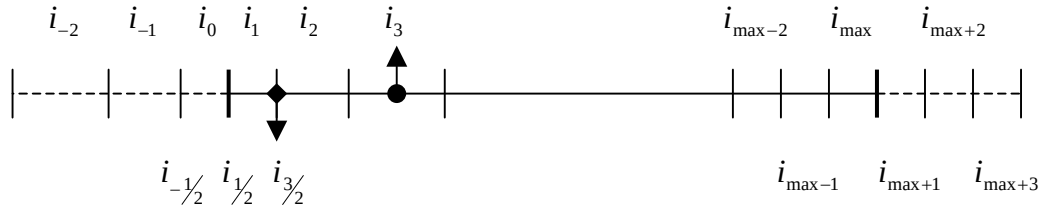


**Figure 7.1 :** Concept of Ghost Cells Approach

In figure 7.1 above, the concept of ghost cell approach in one-dimensional grid is given in a case of a non-uniform beginning and a uniform ending section. As can be seen, on either end, the symmetric of some number of cells, which are located

inside the domain and adjacent to the boundaries, are created. In addition, two-dimensional case uses the same concept with one-dimensional case.

The advantage of using ghost cell approach is; the same discretization scheme can be employed at the boundaries like inside the physical domain. Thus, we can solve the governing equations in the same way in all physical grid points. This makes the discretization scheme much easier to implement. Furthermore, all grid points of a structured grid can be accessed in a single loop. In addition, in a case of multiblock approach, the variables can be transferred between adjacent blocks via ghost cells. [11, page268]. Also, the number of ghost cell layers must be such that the part of the stencil outside the physical domain is completely covered. For example, in 5<sup>th</sup> order WENO, if one wants to use the same order of reconstruction in the boundaries as well as the interior, has to use three dummy cell layers on either end. This is not a small amount to create, because in the case of narrow sections, like a leading edge of a NACA0012 airfoil, which is one of the test cases of this thesis project, dummy cells can degenerate too much. In order to overcome this difficulty, one can reduce the order of reconstruction at the boundaries, like a 3<sup>rd</sup> order WENO or MUSCL. In the generated code order reduction is not implemented for the boundaries, the same order of reconstruction is used for the ghost cell layers, and the solutions show that there is no clue to claim that the deviation of a ghost cell from its symmetric counterpart does affect the solution.

Creating the symmetric of interior grid nodes with respect to physical boundary is accomplished with the help of the expressions given below, where the geometric information is supplied by vector relations [20].

$$x_g = x_b + [(x_i - x_b) - 2Xn_x] \quad (7.1a)$$

$$y_g = y_b + [(y_i - y_b) - 2Xn_y] \quad (7.1b)$$

where, subscripts  $g$ ,  $i$  and  $b$  denote; ghost, interior and boundary nodes respectively. In equations (7.1a) and (7.1b),  $n_x$  and  $n_y$  denote surface normal vectors of the boundary cell.

$$X = (x_i - x_b)n_x + (y_i - y_b)n_y \quad (7.1c)$$

Equation (7.1c) is the contravariant vector component of distance, thus, here  $n_x$  and  $n_y$  denote normal vectors between two nodes  $b$  and  $i$ .

The boundary conditions, with respect to figure 7.1, which are used in the code, are briefly discussed in the following subsections. These boundary conditions assume an inviscid flow, and cell centered decomposition, and if the boundary condition does not using a characteristic variable concept (i.e. farfield boundary condition) it can be implemented by using each component of conservative variables separately.

## 7.2 $0^{th}$ Order Extrapolation Boundary Condition

Here we will assume that there will only be outgoing waves at the boundary of the domain, thus there will be no contribution from outside to inside the domain. Means, there should be no incoming waves. In that case, our goal is to impose boundary conditions on the computational domain that are non-reflecting, or absorbing, and allow any outgoing waves to disappear without generating spurious oscillations [19, page488]. This condition is achieved by simply extrapolating the cell value, which is adjacent to boundary, to the ghost cells.

$$\bar{u}_0 = \bar{u}_1, \quad \bar{u}_{-1} = \bar{u}_1, \quad \bar{u}_{-2} = \bar{u}_1 \quad (7.2a)$$

$$\bar{u}_{i_{\max}+1} = \bar{u}_{i_{\max}}, \quad \bar{u}_{i_{\max}+2} = \bar{u}_{i_{\max}}, \quad \bar{u}_{i_{\max}+3} = \bar{u}_{i_{\max}} \quad (7.2b)$$

## 7.3 Linear Extrapolation Boundary Condition

This boundary condition can be implemented if one wants to extrapolate the value linearly to the ghost cells, for example:

$$\bar{u}_{i_{\max}+1} = \bar{u}_{i_{\max}} + (\bar{u}_{i_{\max}} - \bar{u}_{i_{\max}-1}) \quad (7.3a)$$

$$\bar{u}_{i_{\max}+2} = \bar{u}_{i_{\max}} + 2(\bar{u}_{i_{\max}} - \bar{u}_{i_{\max}-1}) \quad (7.3b)$$

$$\bar{u}_{i_{\max}+3} = \bar{u}_{i_{\max}} + 3(\bar{u}_{i_{\max}} - \bar{u}_{i_{\max}-1}) \quad (7.3c)$$

## 7.4 Periodic/Cut Boundary Condition

There are certain practical applications where the flow field is periodic with respect to one or multiple coordinate directions. In such a case, it is sufficient to simulate the flow only within one of the repeating regions. One example of it implemented in the test case of vortex evolution.

The other case, which is the cut boundary, is encountered only in structured grids. The coordinate cut represents an artificial (not physical) boundary. The grid is folded such that it touches itself. The flow variables and their gradients have to stay continuous across the cut [11, page 286]. One example of it is a C-Grid, which is used for airfoil cross section calculations in this thesis project. For cut boundary the source layer of the cut must be taken into account, like for a periodic source:

$$\bar{u}_0 = \bar{u}_{i_{\max}}, \quad \bar{u}_{-1} = \bar{u}_{i_{\max}-1}, \quad \bar{u}_{-2} = \bar{u}_{i_{\max}-2} \quad (7.4a)$$

$$\bar{u}_{i_{\max}+1} = \bar{u}_1, \quad \bar{u}_{i_{\max}+2} = \bar{u}_2, \quad \bar{u}_{i_{\max}+3} = \bar{u}_3 \quad (7.4b)$$

## 7.5 Farfield Boundary Condition

The farfield boundary condition is implemented from the reference [11, page 277]

The numerical simulations of external flows past airfoils, wings, cars and other configurations have to be conducted within a bounded domain. For this reason, artificial boundary conditions become necessary. The numerical implementation of the farfield boundary conditions has to fulfill two basic requirements. First, the truncation of the domain should have no notable effects on the flow solution as compared to the infinite domain. Second, any outgoing disturbances must not reflect back into the flow field [21]. Due to their elliptic nature, sub- and transonic flow problems are particularly sensitive to the farfield boundary conditions.

The concept of characteristic variables appears here. Depending on the sign of the eigenvalues of the convective flux Jacobians, the information is transported out of or into the computational domain along the characteristics. For example, in the case of a two-dimensional subsonic inflow there are three incoming characteristics and one outgoing, thus, in the case of a two dimensional subsonic outflow there are

three outgoing characteristics and one incoming. Figure 7.2 shows the incoming and outgoing characteristics at subsonic flow case.

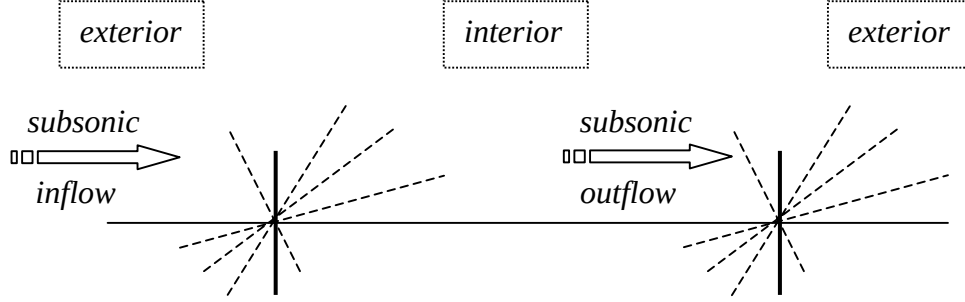


**Figure 7.2 :** Concept of Characteristic Variables at Boundaries (2D)

Depending on the local Mach number, four different types of farfield boundary conditions have to be treated:

- Subsonic Inflow
- Subsonic Outflow
- Supersonic Inflow
- Supersonic Outflow

### 7.5.1 Subsonic Inflow

This flow case is sketched in the front part of figure 7.2. Here, three characteristics enter and one characteristic leaves the physical domain. Therefore, three characteristic variables are prescribed based on freestream values. One characteristic variable is extrapolated from the interior of the physical domain.

This leads to the following set of boundary conditions [22].

$$p_b = \frac{1}{2} \left( p_e + p_i - \rho_0 a_0 \left[ n_x (u_e - u_i) + n_y (v_e - v_i) \right] \right) \quad (7.5a)$$

$$\rho_b = \rho_e + \frac{p_b - p_e}{a_0^2} \quad (7.5b)$$

$$u_b = u_e - n_x \frac{p_e - p_b}{\rho_0 a_0} \quad (7.5c)$$

$$v_b = v_e - n_y \frac{p_e - p_b}{\rho_0 a_0} \quad (7.5d)$$

where  $\rho_0$  and  $a_0$  represents reference state of density and sound speed, and  $b$ ,  $e$ ,  $i$  denote boundary, exterior and interior points respectively. The reference state is normally set equal to the state at the interior point. The values in exterior point are determined from the freestream state.

### 7.5.2 Subsonic Outflow

In the case of subsonic outflow, three flow variables (density and two velocity components) have to be extrapolated from the interior of the physical domain. The remaining fourth variable (pressure) must be specified externally.

The primitive variables at the farfield boundary are obtained from [22]

$$p_b = p_e \quad (7.6a)$$

$$\rho_b = \rho_i + \frac{p_b - p_i}{a_0^2} \quad (7.6b)$$

$$u_b = u_i + n_x \frac{p_i - p_b}{\rho_0 a_0} \quad (7.6c)$$

$$v_b = v_i + n_y \frac{p_i - p_b}{\rho_0 a_0} \quad (7.6d)$$

with exterior node being the prescribed static pressure.

Once physical properties of boundary nodes are found, the properties of ghost nodes (cells) can be obtained by linearly extrapolating from the states  $b$  and  $i$ .

### 7.5.3 Supersonic Inflow

For supersonic inflow all the characteristics are entering the domain. Means there will not be any contribution to ghost cells from the interior. Thus, the conservative variables on the boundary are determined by freestream values only.

$$U_b = U_e \quad (7.7)$$

The values  $U_e$  are specified based on the given freestream Mach number  $M_\infty$  and on two flow angles (angle of attack, side-slip angle).

#### 7.5.4 Supersonic Outflow

For supersonic outflow all characteristics have the same sign like supersonic inflow. They are all leaving the domain, results in no contribution from the ghost cells to interior. All conservative variables at the boundary must be determined from the solution inside the domain.

$$U_b = U_i \quad (7.8)$$

In addition to these discussions, in farfield boundary condition case these characteristic expressions assume zero circulation, which is not correct for a lifting body. For this reason, the farfield boundary has to be located far away from the body. But, the distance to the farfield can be significantly shortened if the freestream flow includes the effect of a single vortex. These modifications for lifting bodies is implemented to the generated code and the related expressions can be found from reference [11, page279].

#### 7.6 Wall (Solid) Boundary Condition

In the case of inviscid flow there is no friction force, and the fluid slips over the surface (the velocity vector must be tangent to the wall surface).

In wall boundary condition case, we apply the symmetric counterpart technique, which is similar to the ghost cells creation. The conservative variables in the interior cell are given to its symmetric counterpart ghost cell. Thus, we create a symmetric flow in the ghost cells. However, this is not sufficient because we have to create an exact mirror image of the flow. In order to accomplish this, we have to give the negative of the momentum vector to the ghost cells, thus, making the mirror of the corresponding momentum vector will give us an image of the flow in the ghost cells. Eventually, flow tangency condition is achieved by this way.

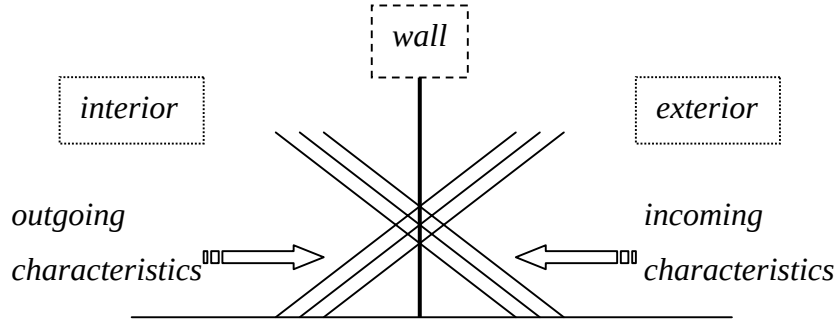


**Figure 7.3 :** Wall Boundary Condition by Creating the Image of the Flow

As can be seen from figure 7.3, creating the image of the flow in the ghost cells imposes the wall boundary condition. As a matter of fact, we allow the interior characteristics to pass through the wall to outside, and the mirror image to pass through wall to inside of the domain. Thus, we simulate a flow, which acts like hitting the wall and returning back.

In general coordinates, this is achieved by assigning each vector of conservative variables to its counterpart ghost cell, then subtracting twice of the contravariant velocity (normal to the surface) from its counterpart ghost cell for momentums.

$$\rho_0 = \rho_1, \quad \rho_{-1} = \rho_2, \quad \rho_{-2} = \rho_3 \quad (7.9a)$$

$$\rho u_0 = \rho u_1 - 2V_1 n_x, \quad \rho u_{-1} = \rho u_2 - 2V_2 n_x, \quad \rho u_{-2} = \rho u_3 - 2V_3 n_x \quad (7.9b)$$

$$\rho v_0 = \rho v_1 - 2V_1 n_y, \quad \rho v_{-1} = \rho v_2 - 2V_2 n_y, \quad \rho v_{-2} = \rho v_3 - 2V_3 n_y \quad (7.9c)$$

$$\rho E_0 = \rho E_1, \quad \rho E_{-1} = \rho E_2, \quad \rho E_{-2} = \rho E_3 \quad (7.9d)$$

Where contravariant velocity is from (2.23), the  $n_x$  and  $n_y$  in (7.9) denote surface normal vectors of the boundary cell, and for right end boundary the expressions are similar to these.

## 7.7 Multiblock Interface Boundary Condition

Sometimes it is not possible to cover a domain with a single grid in structured cases. At these circumstances some number of structured grids can connect to each other and use to cover a more complex geometry, which is called a multiblock technique.

This technique is implemented to the code generated for this thesis project, and the forward facing step test case uses this technique for the solution.

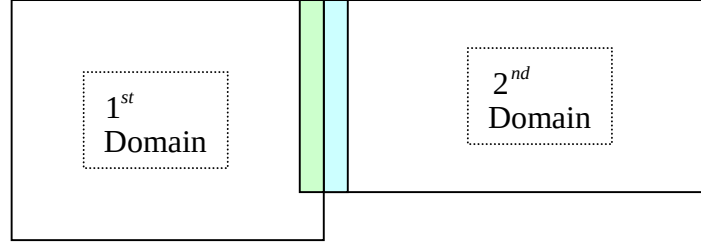


**Figure 7.4 :** Sketch of Multiblock Interfaces

The figure 7.4 shows a sketch of multiblock interface boundary condition. In fact, two domains are not overlapping physically, but a ghost cell layer of one domain is overlapping to an interior cell layer of the other domain. Light blue layer is the ghost cell layer of some part of the right side of 1<sup>st</sup> domain, and the light green layer is the ghost cell layer of the left side of 2<sup>nd</sup> domain.

In order to implement the boundary condition, we put the values from one domain to another in the overlapping layers of cells. For example, in figure 7.4, we start solving from 1<sup>st</sup> domain and at the end of the 1<sup>st</sup> Runge-Kutta step we put the values of the light green layer to the 2<sup>nd</sup> domain as its right boundary condition, and start the 1<sup>st</sup> Runge-Kutta step again in the 2<sup>nd</sup> domain. After this step, we put the values of light blue layer to 1<sup>st</sup> domain as its boundary condition and start 2<sup>nd</sup> Runge-Kutta step there, and the process goes on in this manner.

Besides these boundary conditions discussed so far, we implemented some other boundary conditions to the code, which are inlet and outlet boundary conditions for internal flows and symmetry boundary condition, the discussions for these can be found in reference [11].

In addition to that, some other boundary conditions like fixed to the initial data boundary condition, and problem specific boundary conditions, which are vital for some test cases, i.e. simulating a moving shock wave in ghost cells, had implemented to the code.

## 8. COORDINATE TRANSFORMATIONS

This section discusses Euler equations and metrics calculations for coordinate transformations.

Coordinate transformation is necessary for structured decompositions of WENO reconstructions. The finite volume and finite difference WENO reconstructions for structured grids can be applied on Cartesian grids without a need to a coordinate transformation. However, for a general geometry, which can be covered by a smooth curvilinear grid, coordinate transformation has to be done.

If the domain is smooth enough, it is usually can be mapped smoothly to a rectangle (or at least to a union of non-overlapping) rectangles. That is, the transformation

$$\xi = \xi(x, y), \quad \eta = \eta(x, y) \quad (8.1)$$

maps the physical domain to a rectangular computational domain

$$a \leq \xi \leq b, \quad c \leq \eta \leq d \quad (8.2)$$

We require the transformation functions (8.1) to be smooth (i.e. it has as many derivatives as the scheme calls for). Using chain rule, we could write

$$v_x = \xi_x v_\xi + \eta_x v_\eta \quad (8.3)$$

We can then apply WENO approximations on  $v_\xi$  and  $v_\eta$ , as they are now defined in rectangular domains. The smoothness of  $\xi_x$  and  $\eta_x$  will guarantee that this leads to a high order approximation to  $v_x$  as well through (8.3) [1].

## 8.1 Obtaining Metrics for Coordinate Transformations

Writing the equation (8.3) in a more generalized two dimensional form:

$$\frac{\partial}{\partial x} = \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} \quad (8.4a)$$

$$\frac{\partial}{\partial y} = \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta} \quad (8.4b)$$

The metrics  $\xi_x$ ,  $\eta_x$ ,  $\xi_y$  and  $\eta_y$  appearing in the equations above can be determined in the following manner

$$d\xi = \xi_x dx + \xi_y dy \quad (8.5a)$$

$$d\eta = \eta_x dx + \eta_y dy \quad (8.5b)$$

which in matrix form become

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \quad (8.6)$$

reversing the dependent and independent variables in (8.1) will lead us to the following expressions (8.7)-(8.9)

$$x = x(\xi, \eta), \quad y = y(\xi, \eta) \quad (8.7)$$

$$dx = x_\xi d\xi + x_\eta d\eta \quad (8.8a)$$

$$dy = y_\xi d\xi + y_\eta d\eta \quad (8.8b)$$

which in matrix form

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \quad (8.9)$$

therefore it is evident that

$$\begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}^{-1} \quad (8.10)$$

After some manipulations, which can be found in reference [24, page258], the Jacobian of the transformation is defined to be

$$J = \frac{\partial(\xi, \eta)}{\partial(x, y)} = \begin{vmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{vmatrix} \quad (8.11a)$$

$$\frac{1}{J} = \frac{\partial(x, y)}{\partial(\xi, \eta)} = \begin{vmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{vmatrix} \quad (8.11b)$$

so that the Jacobian and the inverse are found to be

$$J = \xi_x \eta_y - \xi_y \eta_x \quad (8.12a)$$

$$\frac{1}{J} = x_\xi y_\eta - y_\xi x_\eta \quad (8.12b)$$

therefore the metrics are

$$\xi_x = J y_\eta \quad (8.13a)$$

$$\xi_y = -J x_\eta \quad (8.13b)$$

$$\eta_x = -J y_\xi \quad (8.13c)$$

$$\eta_y = J x_\xi \quad (8.13d)$$

It is clear from the equations above that, in order to calculate metrics and Jacobians, we have to find derivatives of  $x$  and  $y$  with respect to  $\xi$  and  $\eta$ . This can be done in two ways. First one is; if the analytic transformation of grid points is available then derivatives can be found in an analytical manner, and the second one is; if analytic transformation of grid points isn't available, the transformation can be constructed by numerically computing the second-order central differences of the points in the physical plane along constant  $\xi$  and  $\eta$  lines [23, page682].

According to the previous paragraph, these derivatives can be constructed as the following

$$x_{\xi} = \frac{x_{i+1} - x_{i-1}}{\xi_{i+1} - \xi_{i-1}} \quad (8.14a)$$

$$y_{\xi} = \frac{y_{i+1} - y_{i-1}}{\xi_{i+1} - \xi_{i-1}} \quad (8.14b)$$

$$x_{\eta} = \frac{x_{j+1} - x_{j-1}}{\eta_{j+1} - \eta_{j-1}} \quad (8.14c)$$

$$y_{\eta} = \frac{y_{j+1} - y_{j-1}}{\eta_{j+1} - \eta_{j-1}} \quad (8.14d)$$

For other points like cell boundaries or Gaussian integration points, the metrics and Jacobians can be found by interpolations [23, page400], for example:

$$(\xi_x)_{i+\frac{1}{2},j} = \frac{1}{2}[(\xi_x)_{i,j} + (\xi_x)_{i+1,j}] \quad (8.15a)$$

$$(J)_{i,j+\frac{1}{2}} = \frac{1}{2}[(J)_{i,j} + (J)_{i,j+1}] \quad (8.15b)$$

## 8.2 2D Governing Equations for Coordinate Transformations

In section 2.2 general form of Euler equations in conservative form was written as

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \quad (8.16)$$

For transformation to a general coordinate system, the conservation equation becomes [23, page398]

$$\frac{\partial \tilde{Q}}{\partial t} + \frac{\partial \tilde{F}}{\partial \xi} + \frac{\partial \tilde{G}}{\partial \eta} = 0 \quad (8.17)$$

where

$$\tilde{Q} = \frac{1}{J} Q \quad (8.18a)$$

$$\tilde{F} = \frac{1}{J} (\xi_x F + \xi_y G) \quad (8.18b)$$

$$\tilde{G} = \frac{1}{J} (\eta_x F + \eta_y G) \quad (8.18c)$$

After some manipulations, which can be found in reference [24, page260], the conservative variables and flux vectors of transformation is found to be

$$\tilde{Q} = \frac{1}{J} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \quad (8.19a)$$

$$\tilde{F} = \frac{1}{J} \begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ (\rho E + p) U \end{bmatrix} \quad (8.19b)$$

$$\tilde{G} = \frac{1}{J} \begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ (\rho E + p) V \end{bmatrix} \quad (8.19c)$$

where

$$U = \xi_x u + \xi_y v \quad (8.20a)$$

$$V = \eta_x u + \eta_y v \quad (8.20b)$$

For a characteristic WENO reconstruction eigenvalues and eigenvectors must be defined. The eigenvalues of  $\frac{\partial \tilde{F}}{\partial \tilde{Q}}$  are  $U - a\sqrt{\xi_x^2 + \xi_y^2}$ ,  $U$ ,  $U$ , and  $U + a\sqrt{\xi_x^2 + \xi_y^2}$ , the right eigenvectors are the columns of

$$R = \begin{bmatrix} 1 & 0 & 1 & 1 \\ u_{RL} - \hat{\xi}_x a_{RL} & \hat{\xi}_y & u_{RL} & u_{RL} + \hat{\xi}_x a_{RL} \\ v_{RL} - \hat{\xi}_y a_{RL} & -\hat{\xi}_x & v_{RL} & v_{RL} + \hat{\xi}_y a_{RL} \\ H_{RL} - \hat{U}_{RL} a_{RL} & u_{RL} \hat{\xi}_y - v_{RL} \hat{\xi}_x & q_{RL} & H_{RL} + \hat{U}_{RL} a_{RL} \end{bmatrix} \quad (8.21a)$$

and the left eigenvectors are the rows of

$$R^{-1} = \begin{bmatrix} \frac{1}{2} \left( b_2 + \frac{\hat{U}_{RL}}{a_{RL}} \right) & -\frac{1}{2} \left( b_1 u_{RL} + \frac{\hat{\xi}_x}{a_{RL}} \right) & -\frac{1}{2} \left( b_1 v_{RL} + \frac{\hat{\xi}_y}{a_{RL}} \right) & \frac{1}{2} b_1 \\ -u_{RL} \hat{\xi}_y + v_{RL} \hat{\xi}_x & \hat{\xi}_y & -\hat{\xi}_x & 0 \\ 1 - b_2 & b_1 u_{RL} & b_1 v_{RL} & -b_1 \\ \frac{1}{2} \left( b_2 - \frac{\hat{U}_{RL}}{a_{RL}} \right) & -\frac{1}{2} \left( b_1 u_{RL} - \frac{\hat{\xi}_x}{a_{RL}} \right) & -\frac{1}{2} \left( b_1 v_{RL} - \frac{\hat{\xi}_y}{a_{RL}} \right) & \frac{1}{2} b_1 \end{bmatrix} \quad (8.21b)$$

where  $b_1$ ,  $b_2$ , and  $q_{RL}$  defined in section 2.2, and

$$\hat{\xi}_x = \frac{\xi_x}{\sqrt{\xi_x^2 + \xi_y^2}} \quad (8.22a)$$

$$\hat{\xi}_y = \frac{\xi_y}{\sqrt{\xi_x^2 + \xi_y^2}} \quad (8.22b)$$

$$\hat{U}_{RL} = \hat{\xi}_x u + \hat{\xi}_y v \quad (8.22c)$$

The explicit definition of  $\frac{\partial \tilde{G}}{\partial \tilde{Q}}$  can easily be found from the expressions above [25].

## 9. ONE DIMENSIONAL RESULTS AND DISCUSSION

This section gives one dimensional test case results for finite difference and finite volume WENO scheme. Only 5<sup>th</sup> order WENO is considered, with 3<sup>rd</sup> order TVD and 4<sup>th</sup> order non-TVD Runge-Kutta time discretizations, with Lax-Friedrichs flux-splitting for finite difference, and with Lax-Friedrichs numerical flux for finite volume cases. They are denoted by FD-WENO5-LF-RK# and FV-WENO5-LF-RK# respectively (local Lax-Friedrichs is denoted by LLF instead of LF). In the next subsection, scalar problems are considered, which are linear advection equation and nonlinear Burger's equation, and in the subsequent subsection vectorial results are presented for Euler equations, which use component- and characteristic-wise reconstructions, and at the end of the section CPU time comparison is given. In the figures, number of nodes has to be treated as number of cells.

### 9.1 1D Results for Scalar Conservation Laws

First problem we solve is the following linear scalar problem, called the advection equation

$$u_t + au_x = 0 \quad (9.1)$$

with  $a = 1$ , which is the speed of the information (wave), and the second one is the nonlinear scalar problem, called the Burger's equation, which is

$$u_t + f(u)_x = 0 \quad (9.2)$$

where  $f(u)$  is a scalar function of  $u$ , and equal to  $\frac{1}{2}u^2$ .

#### 9.1.1 Piecewise Smooth Problem with Advection and Burger's Equations

Here, the initial conditions are given as (9.3), the domain is  $[-1,1]$ , and zero-order extrapolation boundary condition used at each end.

$$u(x,0)=1, \quad \text{for } x \leq 0 \quad (9.3a)$$

$$u(x,0)=0, \quad \text{for } x > 0 \quad (9.3b)$$

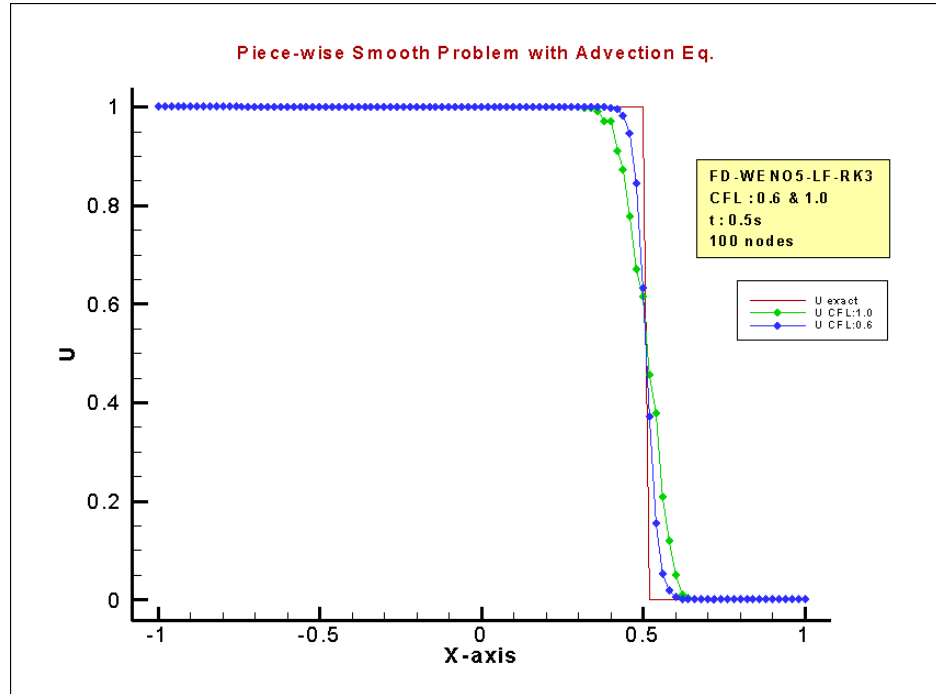


**Figure 9.1 :** Piece-wise Smooth Problem with Advection Equation (FD)

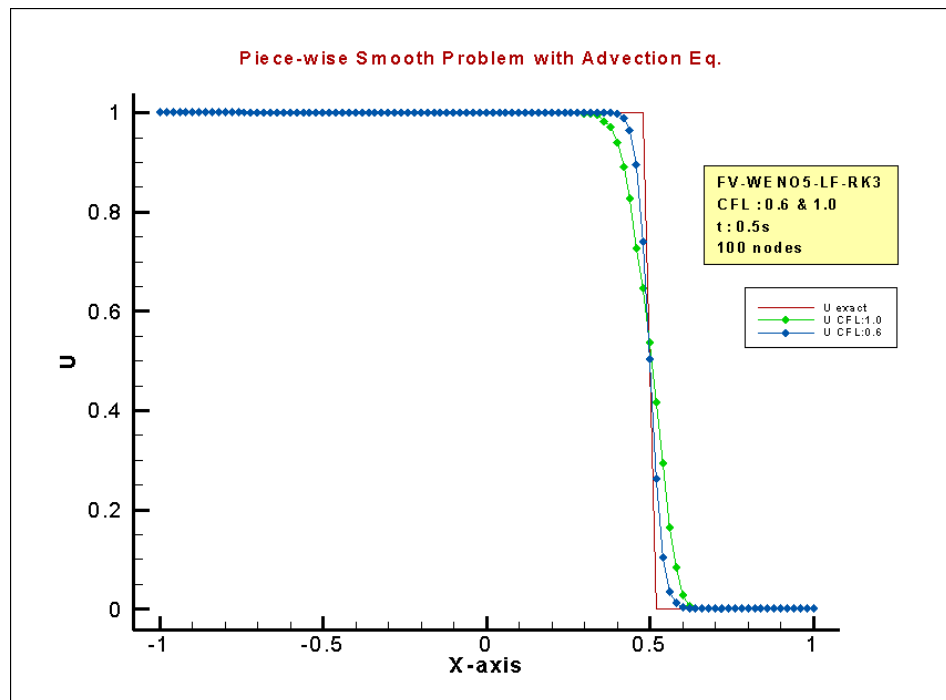


**Figure 9.2 :** Piece-wise Smooth Problem with Advection Equation (FV)

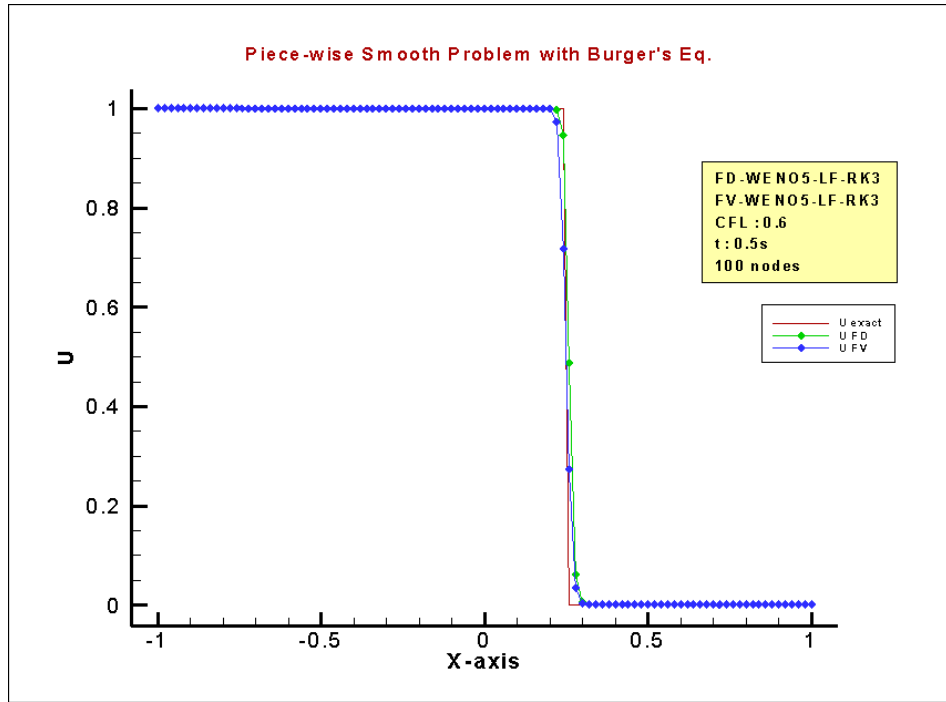


**Figure 9.3 :** Piece-wise Smooth Problem with Burger's Equation (FD & FV)

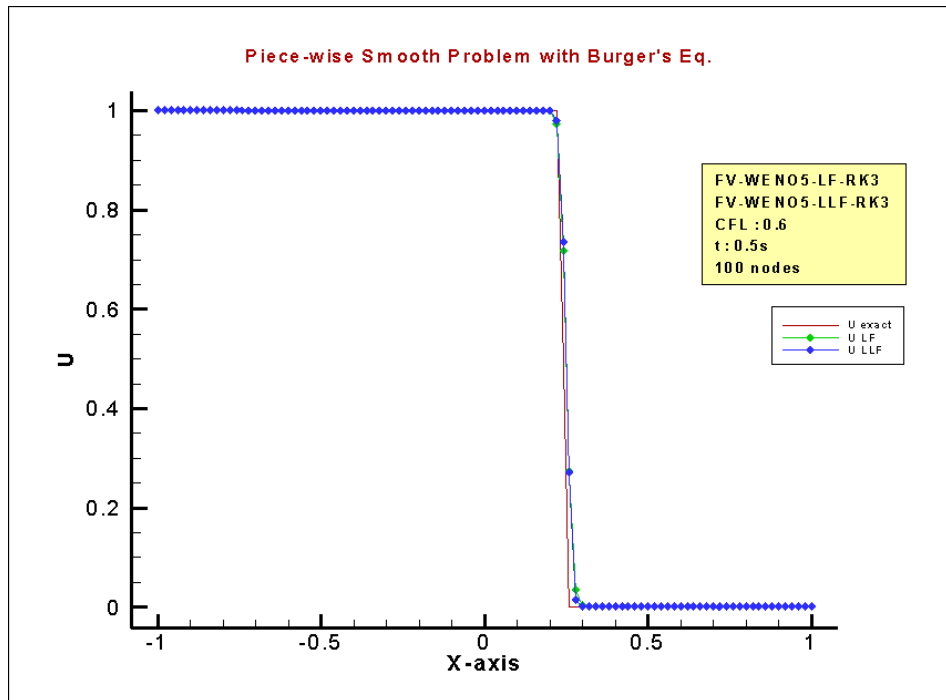


**Figure 9.4 :** Piece-wise Smooth Problem with Burger's Equation (LF & LLF)

From figures 9.1 to 9.4, the results showing that the smearing in advection equation is significantly reduced with a CFL number 0.6 instead of 1.0. Using LLF instead of LF gives only a slight improvement for Burger's equation.

### 9.1.2 Sin-wave Problems with Advection and Burger's Equations

Two sinus waves are used, which have the initial  $\sin(\pi x)$  and  $\sin^4(\pi x)$ . The domain is  $[-1,1]$ , and periodic boundary condition used at each end.

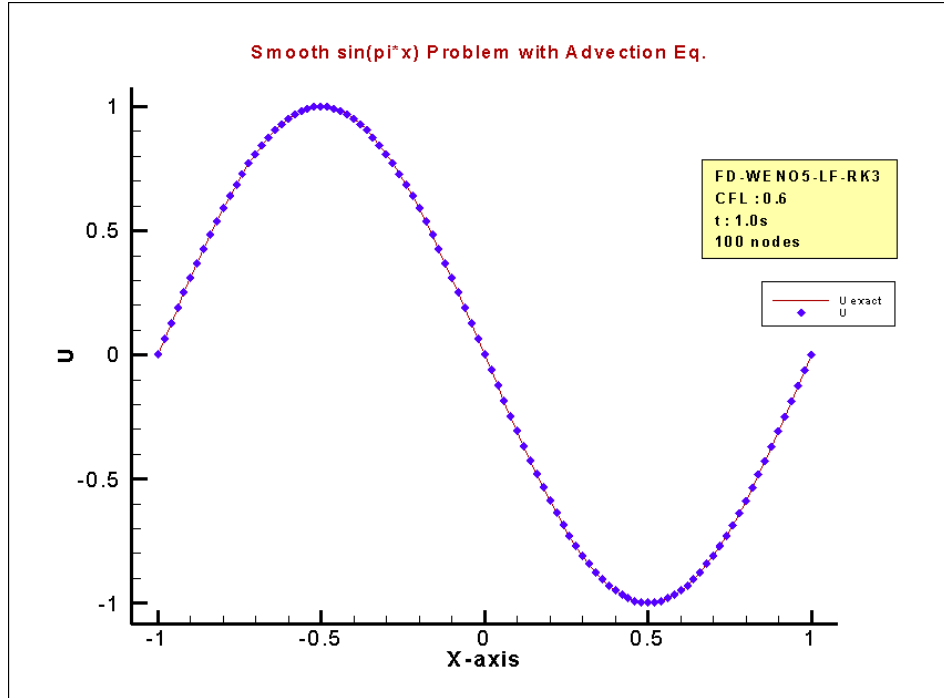


**Figure 9.5 :**  $\sin(\pi x)$  Problem with Advection Equation

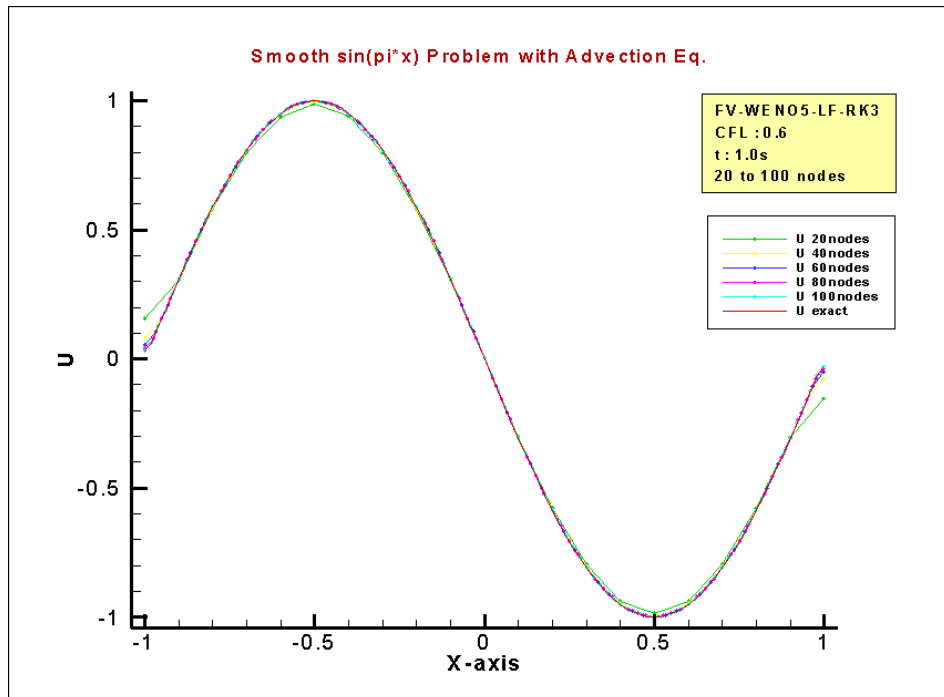


**Figure 9.6 :**  $\sin(\pi x)$  Problem with Advection Equation (20 to 100 cells)

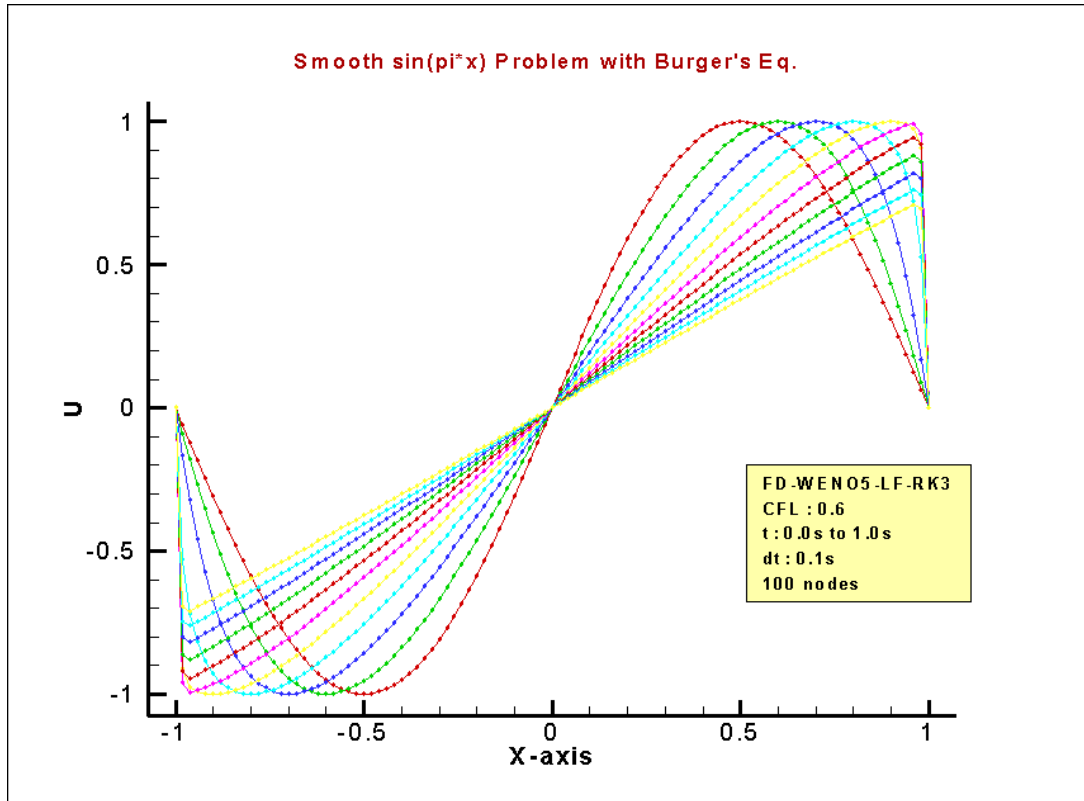


**Figure 9.7 :**  $\sin(\pi x)$  Problem with Burger's Equation ( $t : 0.0s$  to  $1.0s$ )

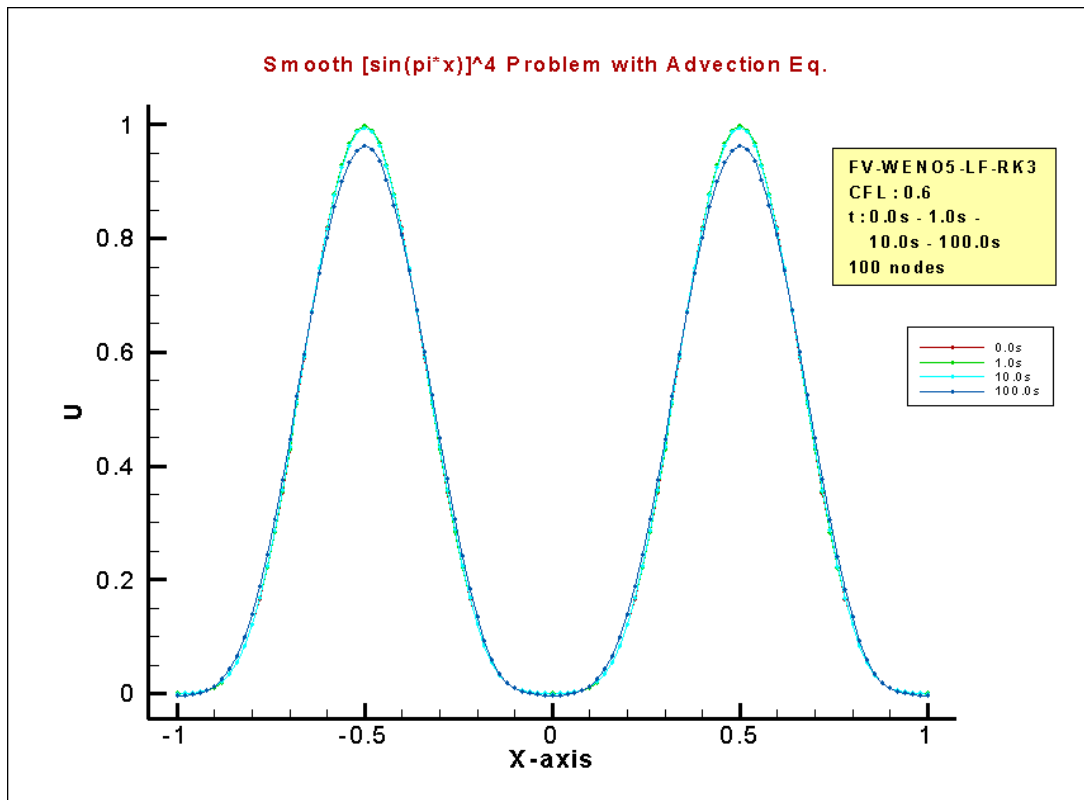


**Figure 9.8 :**  $\sin^4(\pi x)$  Problem with Advection Equation ( $t : 0.0s$  to  $100.0s$ )

In figure 9.5, the initial  $\sin(\pi x)$  problem is calculated for advection equation, and in figure 9.6 this case is calculated for different cell numbers in the domain. In figure 9.7 this case is calculated for Burger's equation and a time advancing plot is given, it can be seen that discontinuities are emerging near the boundaries. In figure 9.8, the initial  $\sin^4(\pi x)$  is calculated with advection equation for a long time simulation, and in figure 9.9 below, figure 9.8 is zoomed-in.

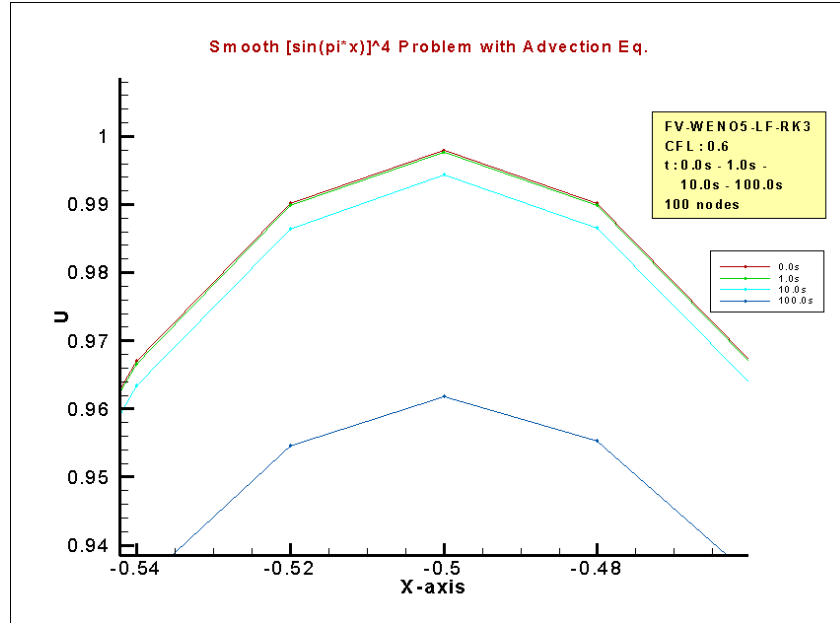


**Figure 9.9 :**  $\sin^4(\pi x)$  Problem with Advection Equation (Figure 9.8 zoomed-in)

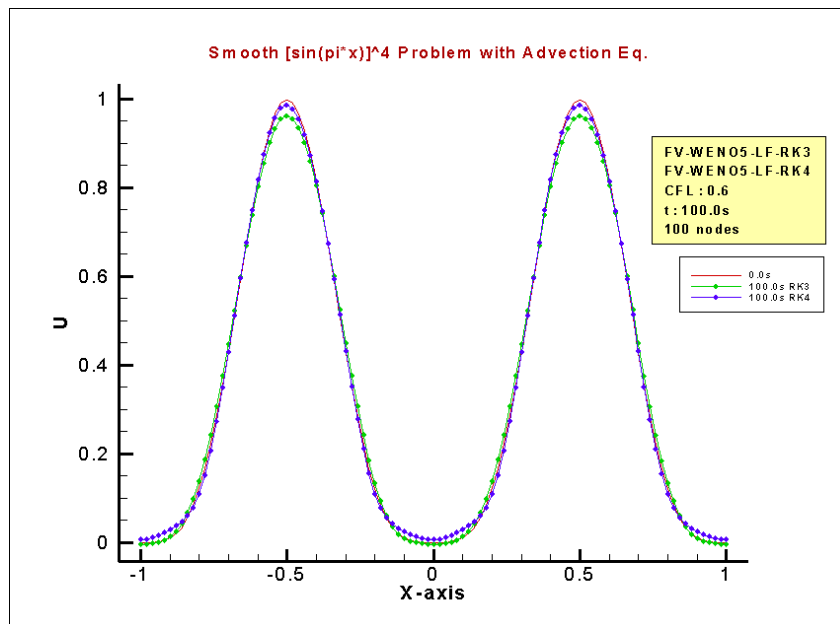


**Figure 9.10 :**  $\sin^4(\pi x)$  Problem with Advection Equation (FV RK3 & RK4)

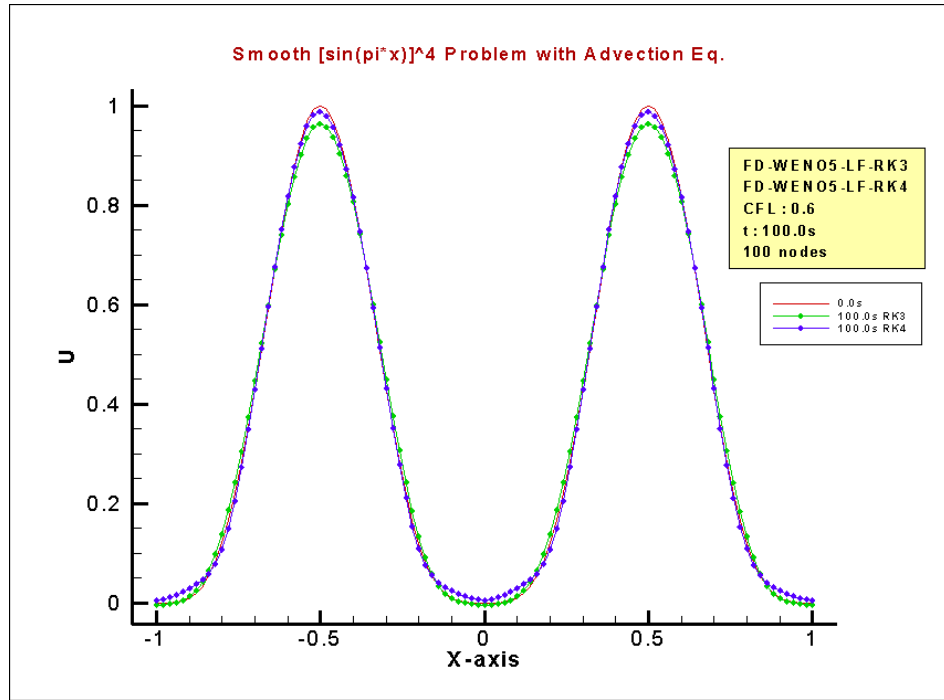


**Figure 9.11 :**  $\sin^4(\pi x)$  Problem with Advection Equation (FD RK3 & RK4)

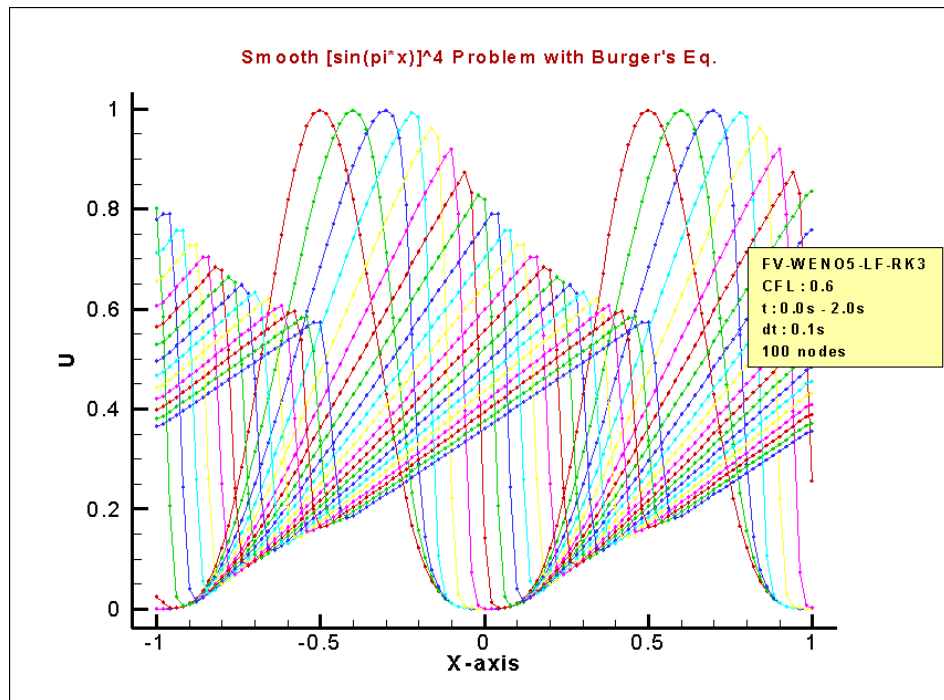


**Figure 9.12 :**  $\sin^4(\pi x)$  Problem with Burger's Equation ( $t : 0.0s$  to  $2.0s$ )

From figures 9.10 and 9.11, it is observed that using RK4 results in less smearing than RK3 for this problem case. In figure 9.12, this case is applied for Burger's equation and a time advancing plot is given.

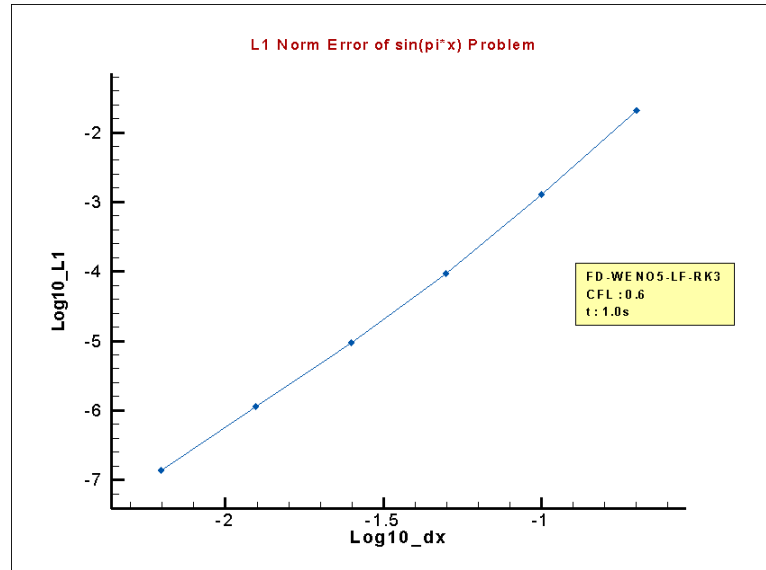


**Figure 9.13 :**  $L_1$  Norm Errors of  $\sin(\pi x)$  Problem with Advection Equation (RK3)

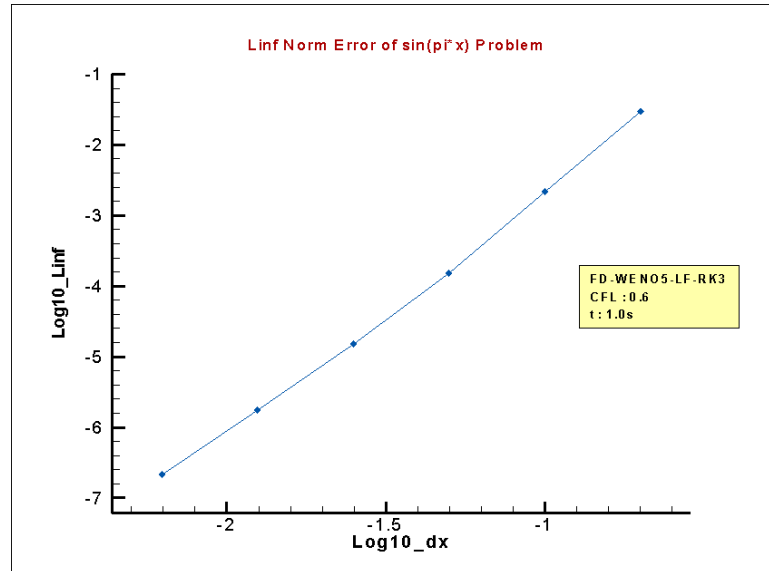


**Figure 9.14 :**  $L_\infty$  Norm Errors of  $\sin(\pi x)$  Problem with Advection Equation (RK3)

**Table 9.1 :** Norm Errors of  $u_0(x) = \sin(\pi x)$  with Advection Equation (FD-LF-RK3)

| Nodes | $L_1$ Error | $L_1$ Order | $L_\infty$ Error | $L_\infty$ Order |
|-------|-------------|-------------|------------------|------------------|
| 10    | 2,10E-02    |             | 3,00E-02         |                  |
| 20    | 1,26E-03    | 4,06        | 2,17E-03         | 3,79             |
| 40    | 9,17E-05    | 3,78        | 1,49E-04         | 3,87             |
| 80    | 9,37E-06    | 3,29        | 1,47E-05         | 3,33             |
| 160   | 1,11E-06    | 3,08        | 1,74E-06         | 3,08             |
| 320   | 1,37E-07    | 3,02        | 2,14E-07         | 3,02             |

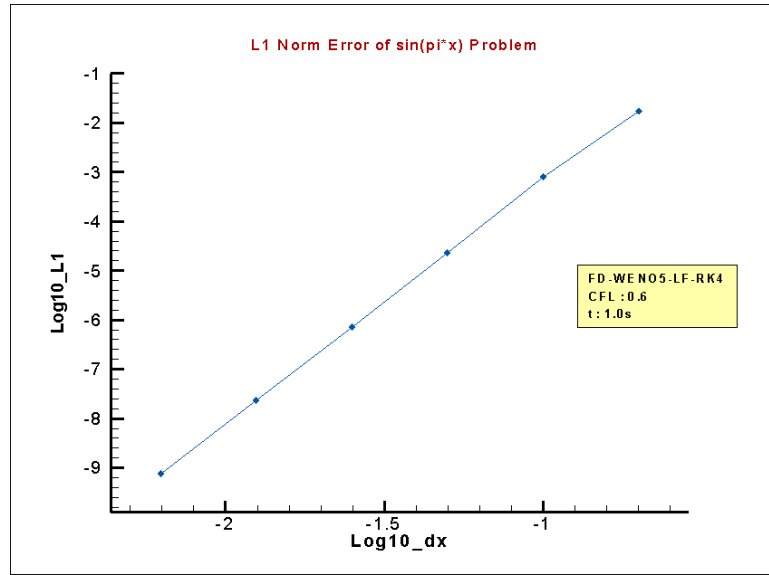


**Figure 9.15 :**  $L_1$  Norm Errors of  $\sin(\pi x)$  Problem with Advection Equation (RK4)

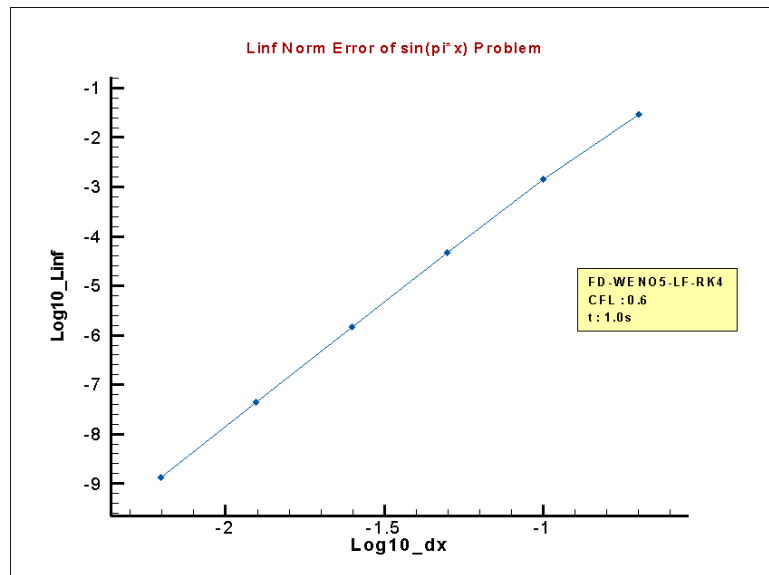


**Figure 9.16 :**  $L_\infty$  Norm Errors of  $\sin(\pi x)$  Problem with Advection Equation (RK4)

**Table 9.2 :** Norm Errors of  $u_0(x) = \sin(\pi x)$  with Advection Equation (FD-LF-RK4)

| Nodes | $L_1$ Error | $L_1$ Order | $L_\infty$ Error | $L_\infty$ Order |
|-------|-------------|-------------|------------------|------------------|
| 10    | 1,75E-02    |             | 2,97E-02         |                  |
| 20    | 7,80E-04    | 4,48        | 1,43E-03         | 4,38             |
| 40    | 2,30E-05    | 5,09        | 4,52E-05         | 4,98             |
| 80    | 7,10E-07    | 5,02        | 1,45E-06         | 4,96             |
| 160   | 2,26E-08    | 4,97        | 4,32E-08         | 5,07             |
| 320   | 7,58E-10    | 4,90        | 1,32E-09         | 5,03             |

**Table 9.3 :** Norm Errors of  $u_0(x) = \sin(\pi x)$  with Advection Equation (FV-LF-RK3)

| <b>Nodes</b> | <b><math>L_1</math> Error</b> | <b><math>L_1</math> Order</b> | <b><math>L_\infty</math> Error</b> | <b><math>L_\infty</math> Order</b> |
|--------------|-------------------------------|-------------------------------|------------------------------------|------------------------------------|
| 10           | 2,03E-02                      |                               | 2,98E-02                           |                                    |
| 20           | 1,25E-03                      | 4,02                          | 2,11E-03                           | 3,82                               |
| 40           | 9,16E-05                      | 3,77                          | 1,49E-04                           | 3,82                               |
| 80           | 9,36E-06                      | 3,29                          | 1,48E-05                           | 3,33                               |
| 160          | 1,11E-06                      | 3,08                          | 1,74E-06                           | 3,09                               |
| 320          | 1,37E-07                      | 3,02                          | 2,14E-07                           | 3,02                               |

**Table 9.4 :** Norm Errors of  $u_0(x) = \sin(\pi x)$  with Advection Equation (FV-LF-RK4)

| <b>Nodes</b> | <b><math>L_1</math> Error</b> | <b><math>L_1</math> Order</b> | <b><math>L_\infty</math> Error</b> | <b><math>L_\infty</math> Order</b> |
|--------------|-------------------------------|-------------------------------|------------------------------------|------------------------------------|
| 10           | 1,63E-02                      |                               | 2,44E-02                           |                                    |
| 20           | 7,49E-04                      | 4,44                          | 1,31E-03                           | 4,22                               |
| 40           | 2,28E-05                      | 5,04                          | 4,69E-05                           | 4,80                               |
| 80           | 7,08E-07                      | 5,01                          | 1,41E-06                           | 5,06                               |
| 160          | 2,26E-08                      | 4,97                          | 4,36E-08                           | 5,01                               |
| 320          | 7,56E-10                      | 4,90                          | 1,31E-09                           | 5,05                               |

**Table 9.5 :** Norm Errors of  $u_0(x) = \sin^4(\pi x)$  with Advection Equation (FV-LF-RK3)

| <b>Nodes</b> | <b><math>L_1</math> Error</b> | <b><math>L_1</math> Order</b> | <b><math>L_\infty</math> Error</b> | <b><math>L_\infty</math> Order</b> |
|--------------|-------------------------------|-------------------------------|------------------------------------|------------------------------------|
| 10           | 1,95E-01                      |                               | 4,05E-01                           |                                    |
| 20           | 5,73E-02                      | 1,77                          | 1,30E-01                           | 1,64                               |
| 40           | 5,04E-03                      | 3,51                          | 9,95E-03                           | 3,70                               |
| 80           | 5,92E-04                      | 3,09                          | 1,51E-03                           | 2,72                               |
| 160          | 3,99E-05                      | 3,89                          | 8,07E-05                           | 4,22                               |
| 320          | 4,41E-06                      | 3,18                          | 8,91E-06                           | 3,18                               |
| 640          | 5,54E-07                      | 2,99                          | 1,08E-06                           | 3,05                               |
| 1280         | 6,93E-08                      | 3,00                          | 1,34E-07                           | 3,01                               |

**Table 9.6 :** Norm Errors of  $u_0(x) = \sin^4(\pi x)$  with Advection Eq. (FV-LLF-RK3)

| <b>Nodes</b> | <b><math>L_1</math> Error</b> | <b><math>L_1</math> Order</b> | <b><math>L_\infty</math> Error</b> | <b><math>L_\infty</math> Order</b> |
|--------------|-------------------------------|-------------------------------|------------------------------------|------------------------------------|
| 10           | 1,77E-01                      |                               | 3,69E-01                           |                                    |
| 20           | 4,88E-02                      | 1,86                          | 1,06E-01                           | 1,80                               |
| 40           | 7,07E-03                      | 2,79                          | 2,02E-02                           | 2,39                               |
| 80           | 9,69E-04                      | 2,87                          | 5,11E-03                           | 1,98                               |
| 160          | 3,86E-05                      | 4,65                          | 8,02E-05                           | 5,99                               |
| 320          | 4,52E-06                      | 3,09                          | 8,91E-06                           | 3,17                               |
| 640          | 5,56E-07                      | 3,02                          | 1,08E-06                           | 3,05                               |
| 1280         | 6,93E-08                      | 3,01                          | 1,34E-07                           | 3,01                               |

Up to now norm errors are given in tables and figures above, it is observed that  $3^{rd}$  order TVD Runge-Kutta time discretization is achieving  $3^{rd}$  order accuracy.

**Table 9.7 :** Norm Errors of  $u_0(x) = \sin^4(\pi x)$  with Advection Eq. (FV-LF-RK4)

| Nodes | $L_1$ Error | $L_1$ Order | $L_\infty$ Error | $L_\infty$ Order |
|-------|-------------|-------------|------------------|------------------|
| 10    | 1,97E-01    |             | 3,79E-01         |                  |
| 20    | 5,39E-02    | 1,87        | 1,30E-01         | 1,55             |
| 40    | 3,79E-03    | 3,83        | 9,61E-03         | 3,75             |
| 80    | 4,75E-04    | 3,00        | 1,52E-03         | 2,66             |
| 160   | 1,49E-05    | 5,00        | 6,63E-05         | 4,52             |
| 320   | 3,90E-07    | 5,25        | 1,77E-06         | 5,23             |
| 640   | 9,41E-09    | 5,37        | 3,16E-08         | 5,81             |
| 1280  | 2,50E-10    | 5,23        | 5,76E-10         | 5,78             |

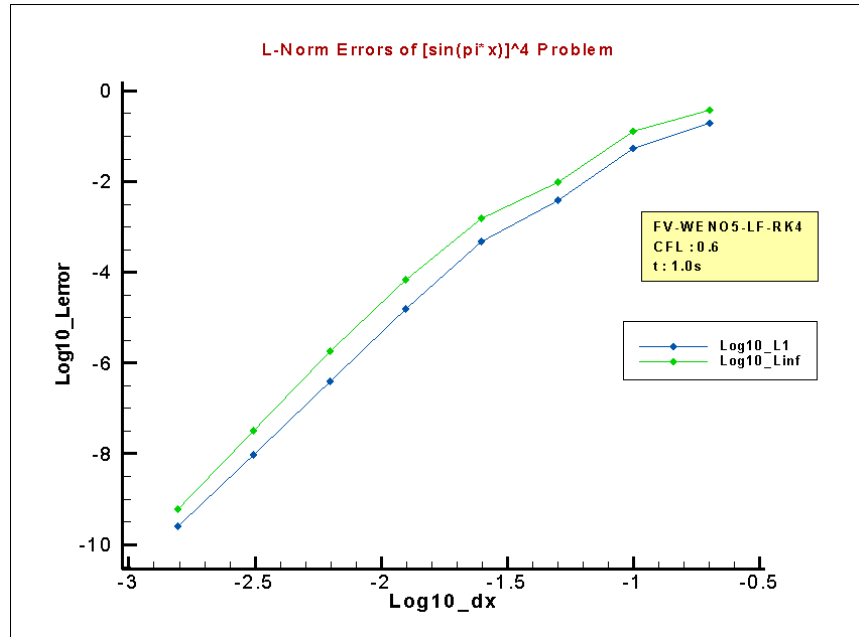


**Figure 9.17 :**  $L$ -Norm Errors of  $\sin^4(\pi x)$  Problem with Advection Eq. (LF-RK4)

**Table 9.8 :** Norm Errors of  $u_0(x) = \sin^4(\pi x)$  with Advection Eq. (FV-LLF-RK4)

| Nodes | $L_1$ Error | $L_1$ Order | $L_\infty$ Error | $L_\infty$ Order |
|-------|-------------|-------------|------------------|------------------|
| 10    | 1,86E-01    |             | 3,36E-01         |                  |
| 20    | 4,46E-02    | 2,06        | 1,05E-01         | 1,68             |
| 40    | 5,77E-03    | 2,95        | 1,85E-02         | 2,50             |
| 80    | 8,27E-04    | 2,80        | 5,07E-03         | 1,87             |
| 160   | 7,20E-06    | 6,84        | 2,50E-05         | 7,67             |
| 320   | 1,57E-07    | 5,52        | 5,30E-07         | 5,56             |
| 640   | 4,80E-09    | 5,03        | 1,69E-08         | 4,97             |
| 1280  | 2,01E-10    | 4,58        | 5,37E-10         | 4,98             |

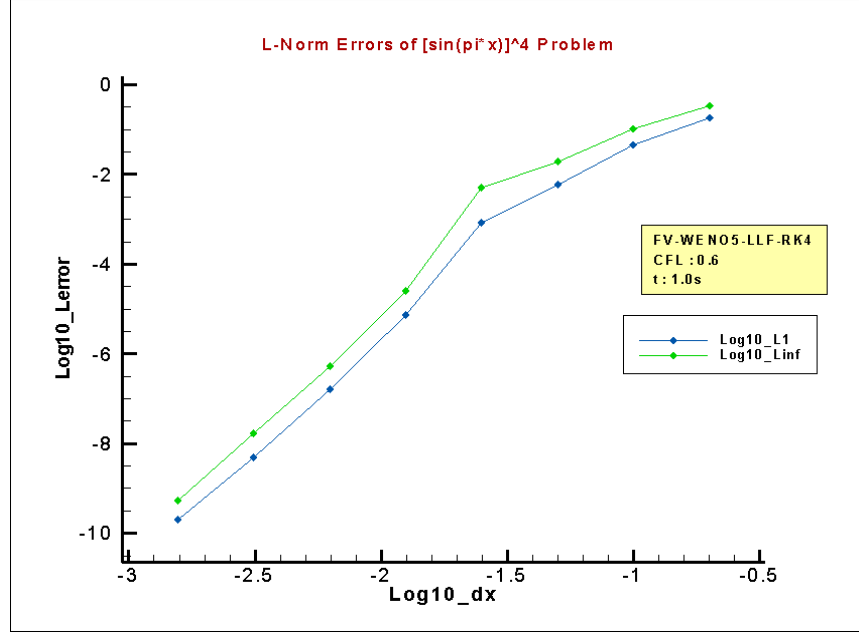


**Figure 9.18 :**  $L$ -Norm Errors of  $\sin^4(\pi x)$  Problem with Advection Eq. (LLF-RK4)

Up to now, it is observed that both FD and FV versions of the code give very similar results, as we expected for one dimension.  $L_1$  and  $L_\infty$  error norms and error orders are given in the tables above. This accuracy test shows that, the TVD RK3 converges to  $3^{rd}$  order while the non-TVD RK4 converges to  $5^{th}$  order accuracy. We know that, for a completely  $5^{th}$  order scheme we have to use  $5^{th}$  order approximations both for spatial and time domain. But, despite the sacrifice in overall accuracy, it is observed that, it will be sufficient to use RK3 for problems (especially in 2D). In addition to that, if the figures 9.17 and 9.18 are compared, it can be seen that using LLF instead of LF results in a more oscillatory convergence, and the tables show that using LLF sacrifices overall accuracy compared to LF.

### 9.1.3 Riemann Problems with Nonconvex Flux Function

In these two problem cases, a nonconvex flux used to solve 1D Riemann problem with two different initial conditions. The domain is  $[-1,1]$ , and zero-order extrapolation boundary condition used at each end. The nonconvex flux, which is used in equation (9.2), is the following

$$f(u) = \frac{1}{4}(u^2 - 1)(u^2 - 4) \quad (9.4)$$

with initial conditions for the first case as (9.5) and for the second case as (9.6)

$$u(x,0) = 2, \quad \text{for } x \leq 0 \quad (9.5a)$$

$$u(x,0) = -2, \quad \text{for } x > 0 \quad (9.5b)$$

$$u(x,0) = -3, \quad \text{for } x \leq 0 \quad (9.6a)$$

$$u(x,0) = 3, \quad \text{for } x > 0 \quad (9.6b)$$

Results for different versions of ENO can be found in [9,27] and for WENO in [10].

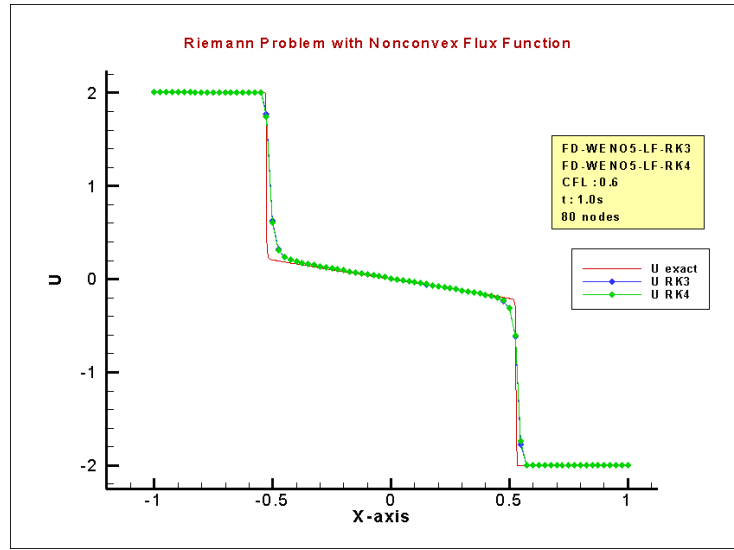


**Figure 9.19 :** Riemann Problem with Nonconvex Flux Function (Case 1 - FD)

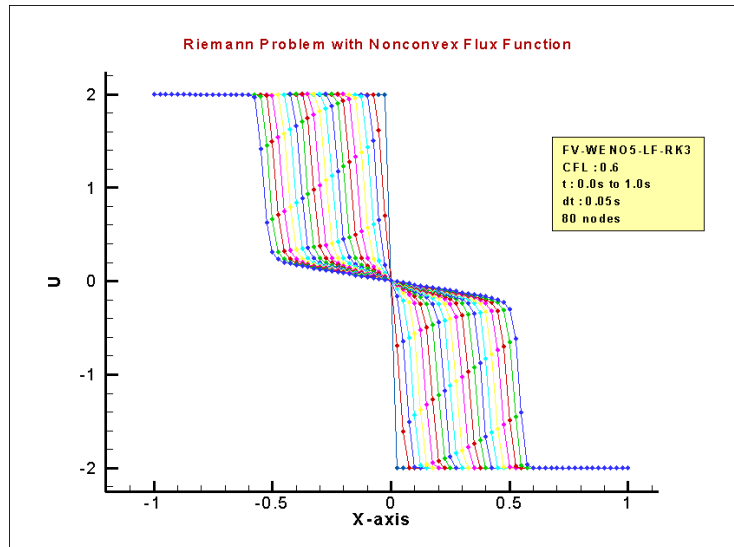


**Figure 9.20 :** Riemann Problem with Nonconvex Flux Function (Case 1 - FV)

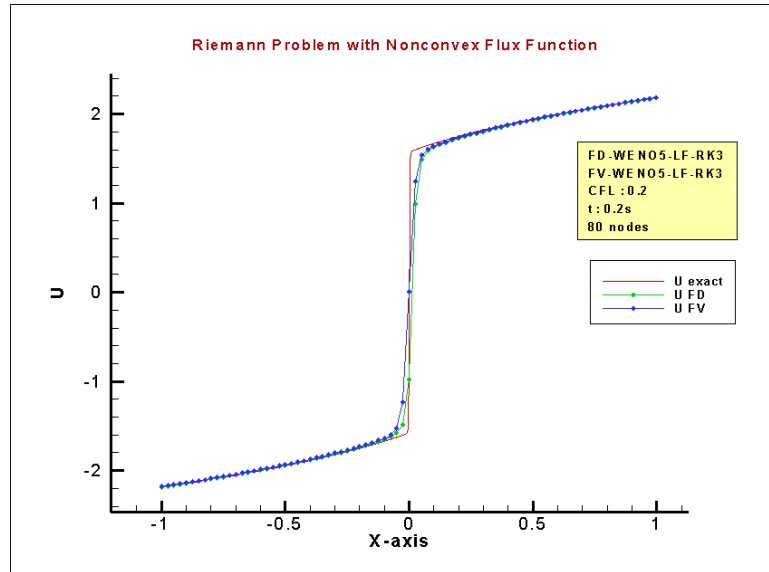


**Figure 9.21 :** Riemann Problem with Nonconvex Flux Function (Case 2 - FD & FV)

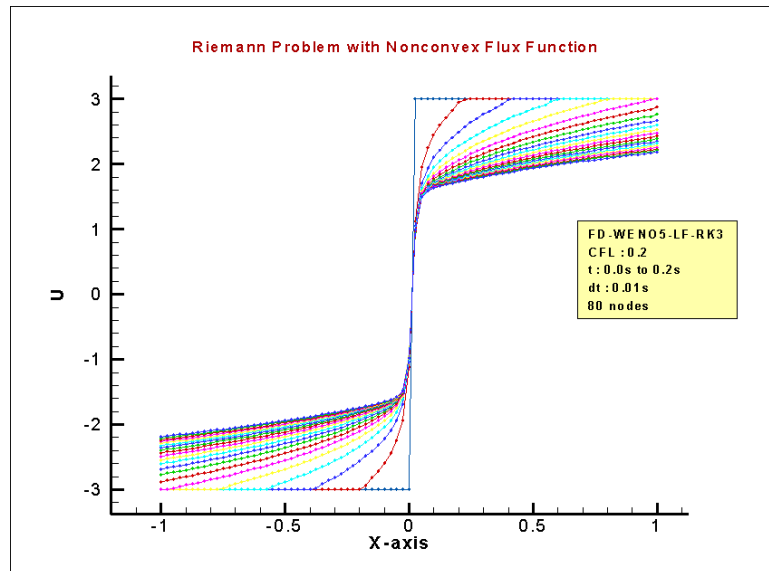


**Figure 9.22 :** Riemann Problem with Nonconvex Flux Function (Case 2 - FD)

In figure 9.19, first case is solved with FD type decomposition and RK3 and RK4 time decompositions are compared, it is observed that both of them give very similar results. In figure 9.20, this case is solved using FV type decomposition and a time advancing plot is given, it can be seen from the figure that two discontinuities emerge and advances in opposite directions. In figure 9.21, second case is solved and a comparison of FD and FV is given, and it is observed that both of them working well and giving similar results. In figure 9.22, this case is given with a time advancing plot.

### 9.1.4 Buckley-Leverett Problem

In this problem case another nonconvex flux function different from (9.4) is used, which is the following

$$f(u) = \frac{4u^2}{4u^2 + (1-u)^2} \quad (9.7)$$

with initial conditions as (9.8)

$$u(x,0) = 0, \quad \text{for } x < -0.5 \quad (9.8a)$$

$$u(x,0) = 1, \quad \text{for } -0.5 \leq x \leq 0 \quad (9.8b)$$

$$u(x,0) = 0, \quad \text{for } x > 0 \quad (9.8c)$$

The domain is  $[-1,1]$ , and zero-order extrapolation boundary condition used at each end.

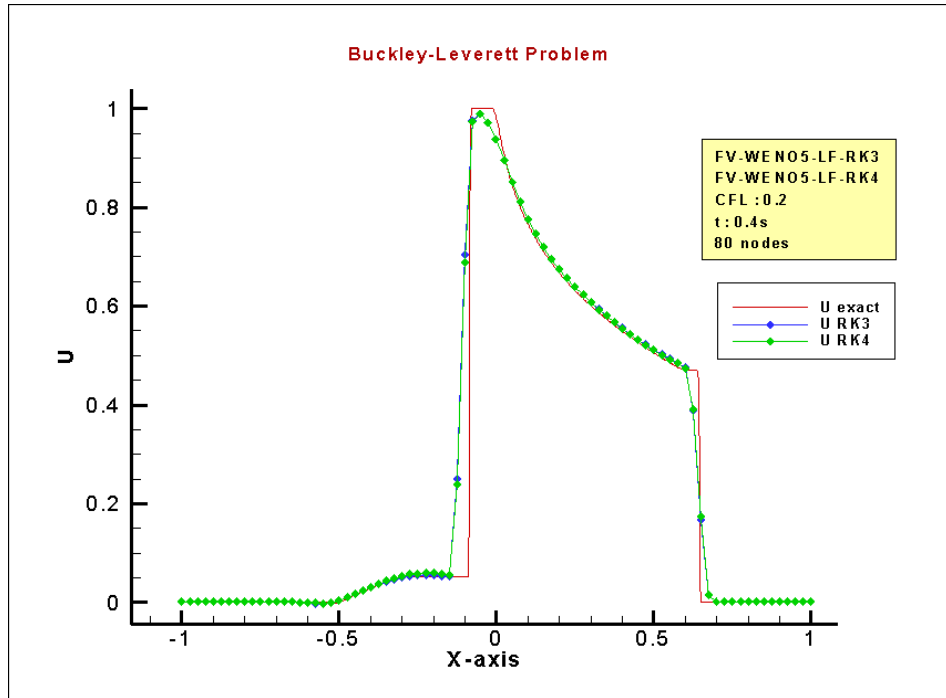


**Figure 9.23 :** Buckley-Leverett Problem (RK3 & RK4)

Figure 9.23 presents the solution for  $t = 0.4$  s. It can be seen that both RK3 and RK4 give very similar results.

From the results for scalar problems given throughout this section, we observe good convergence and resolution for all test cases. For the nonconvex flux functions, it is observed that using either RK3 or RK4 give very similar results. In the sections 9.1.3 and 9.1.4, the exact (reference) solutions are found by a very fine grid, which uses 800 cells.

## 9.2 1D Results for Systems of Conservation Laws

In this subsection, the results of one dimensional conservation laws for Euler equations are presented. Sod and Lax type shock tubes, the problem of Shu, and two interacting blast waves problem are considered.

### 9.2.1 Sod Shock Tube Problem

This is a classic test problem especially for one-dimensional conservation laws, which are applied to fluid flow with discontinuities. In this problem case, after giving a discontinuous initial condition as (9.9), one rarefaction wave, one contact discontinuity and one shock wave are forming, and rarefaction wave is moving to the high dense side while the others are moving to the opposite side.

$$\rho_L = 1.0, \quad u_L = 0.0, \quad p_L = 1.0 \quad (9.9a)$$

$$\rho_R = 0.125, \quad u_R = 0.0, \quad p_R = 0.1 \quad (9.9b)$$

The domain is  $[-5, 5]$ , and wall boundary condition used at each end.

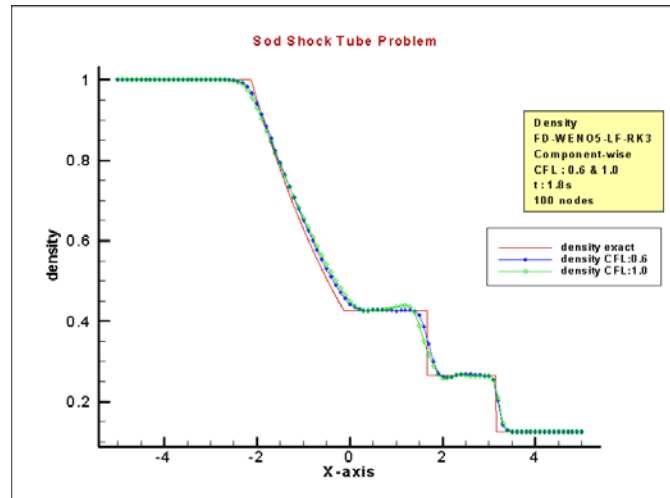


**Figure 9.24 :** Sod Shock Tube Problem (Component-wise - FD - CFL:0.6 & 1.0)

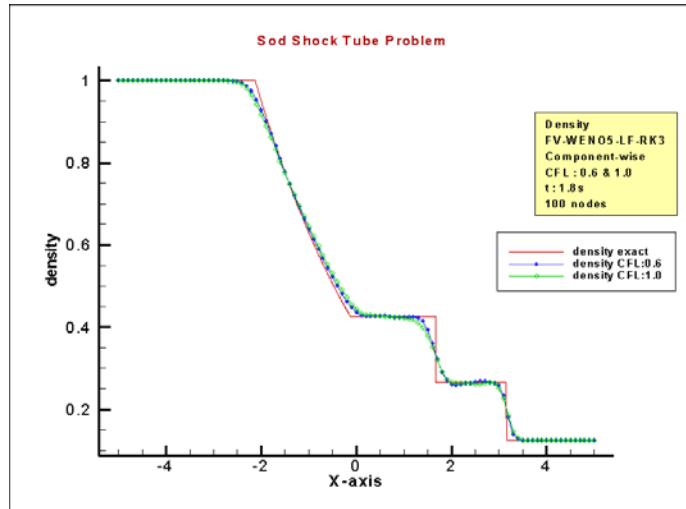


**Figure 9.25 :** Sod Shock Tube Problem (Component-wise - FV - CFL:0.6 & 1.0)

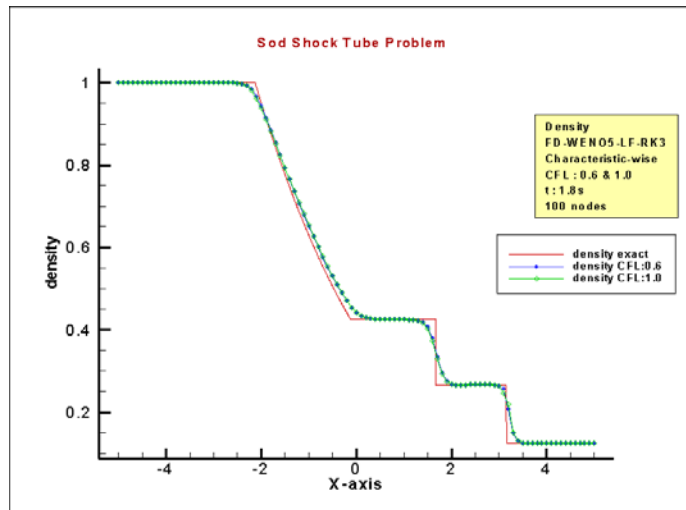


**Figure 9.26 :** Sod Shock Tube Problem (Characteristic-wise - FD - CFL:0.6 & 1.0)

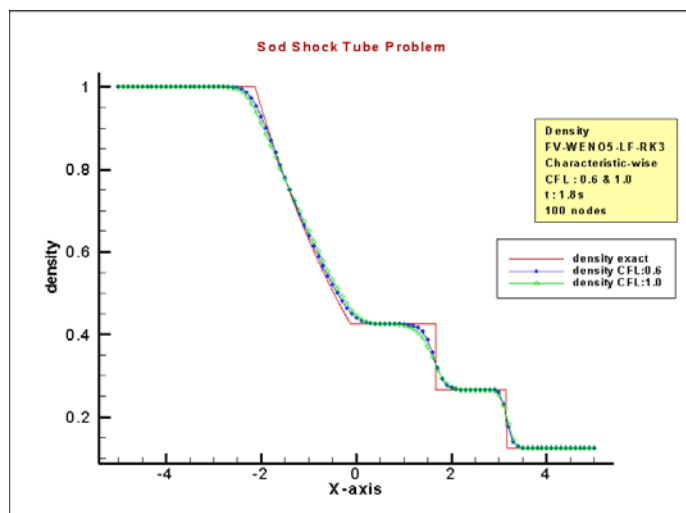


**Figure 9.27 :** Sod Shock Tube Problem (Characteristic-wise - FV - CFL:0.6 & 1.0)

From figures 9.24 to 9.27, it is observed that component-wise solutions gives noises compared to characteristic-wise solutions. CFL number of 0.6 gives better results compared to 1.0. FV decomposition gives more smearing in contact discontinuities, shocks, and in rarefaction waves compared to FD decomposition, however, the position of the rarefaction waves and shocks are captured more accurately with FV.

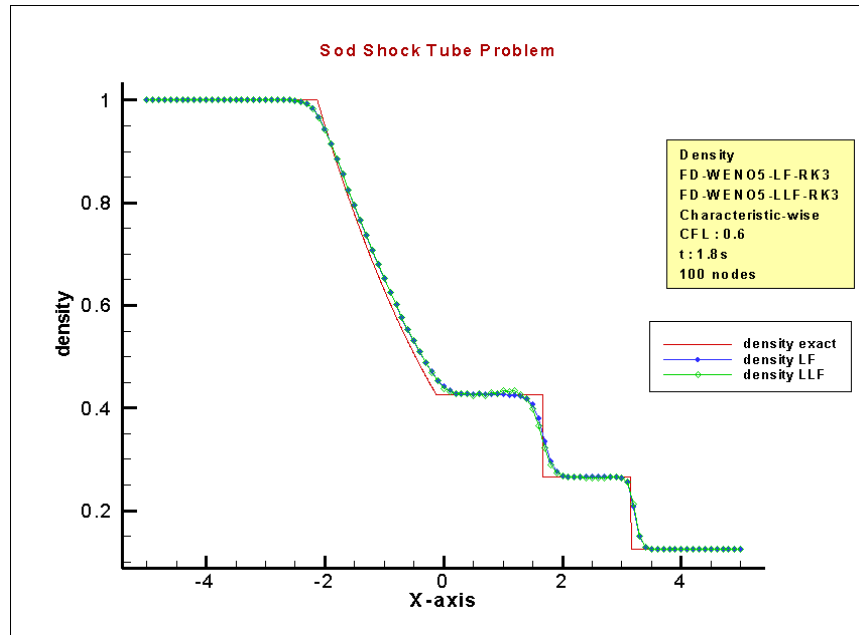


**Figure 9.28 : Sod Shock Tube Problem (FD - LF & LLF)**

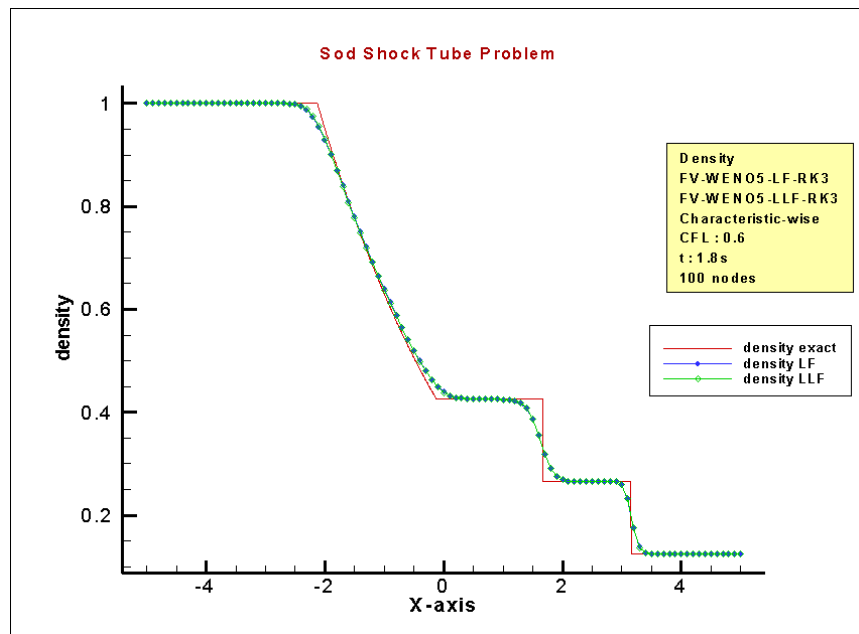


**Figure 9.29 : Sod Shock Tube Problem (FV - LF & LLF)**

From figures 9.28 and 9.29 it is seen that LLF with FD decomposition gives noises near the contact discontinuity, but with FV decomposition, there is not such a noise appears there. LLF gives very minor advantage in results than LF for FV kind of decomposition.

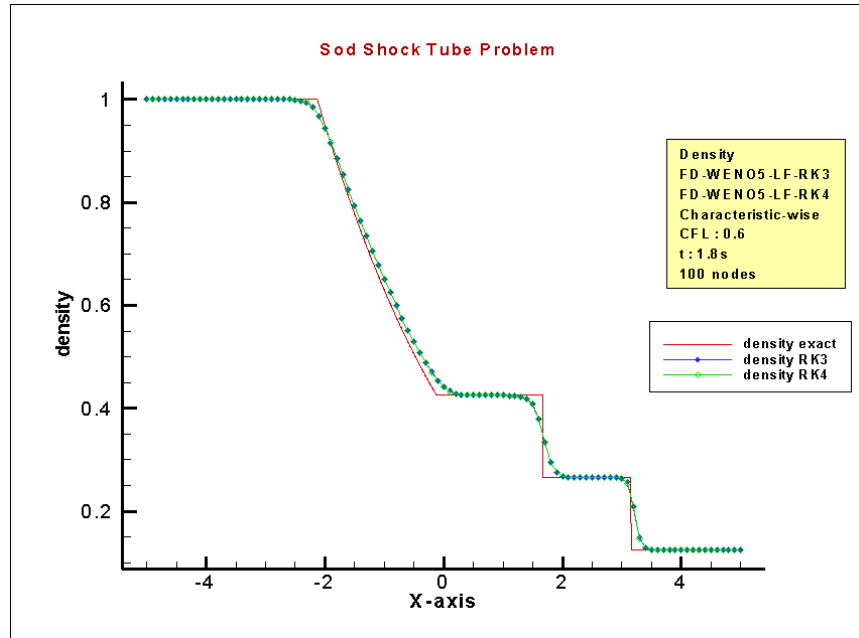


**Figure 9.30 : Sod Shock Tube Problem (FD - RK3 & RK4)**

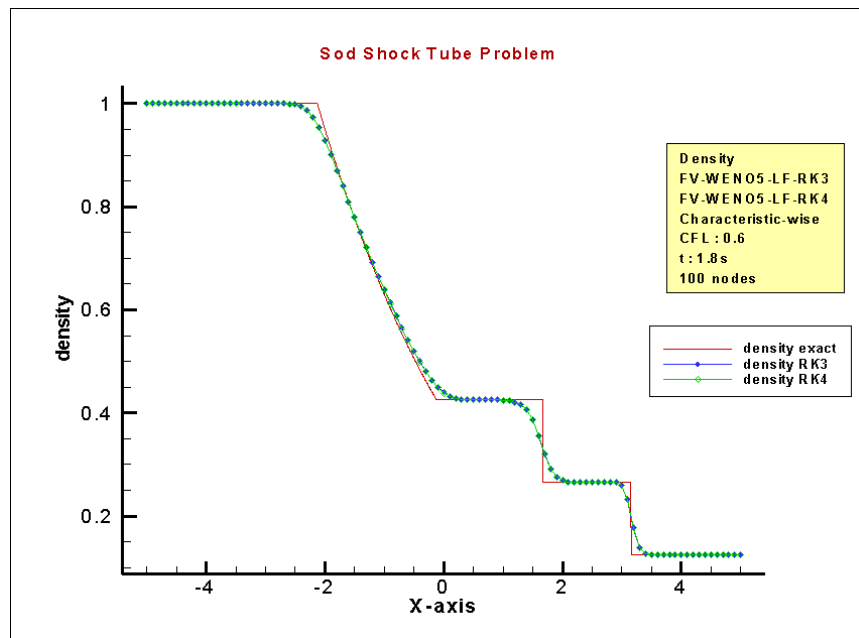


**Figure 9.31 : Sod Shock Tube Problem (FV - RK3 & RK4)**

In figures 9.30 and 9.31, RK3 and RK4 type time decompositions are compared. Either with FD or FV, no such significant difference is observed.

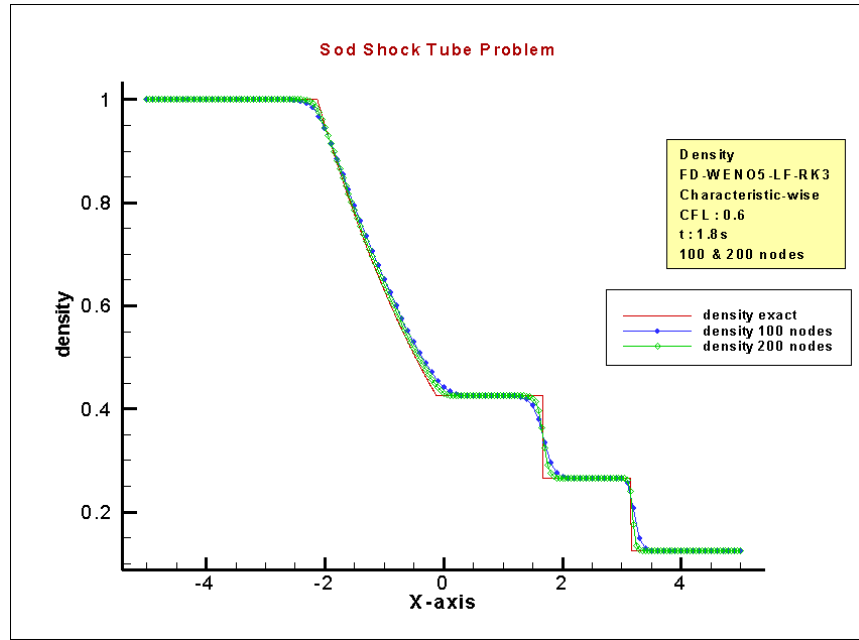


**Figure 9.32 :** Sod Shock Tube Problem (FD - 100 & 200 cells)

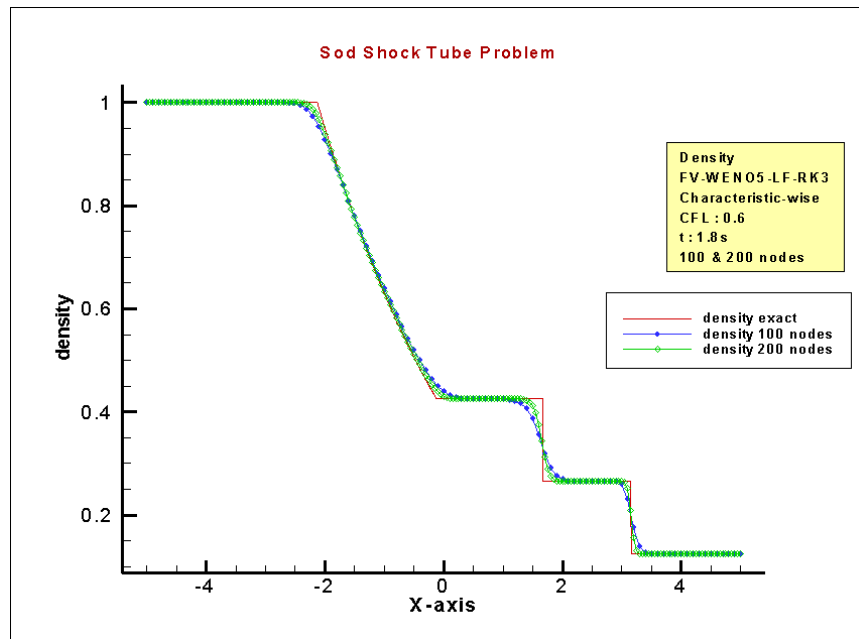


**Figure 9.33 :** Sod Shock Tube Problem (FV - 100 & 200 cells)

The differences between using 100 cells and 200 cells both for FD and FV type decompositions can be observed from the figures 9.32 and 9.33 above.

In this section, the figures given so far are investigated the difference in reconstruction types used for 5<sup>th</sup> order WENO with the density variable. The following figures will present other variables used with FD-WENO5-LF-RK3 and FV-WENO5-LF-RK3.

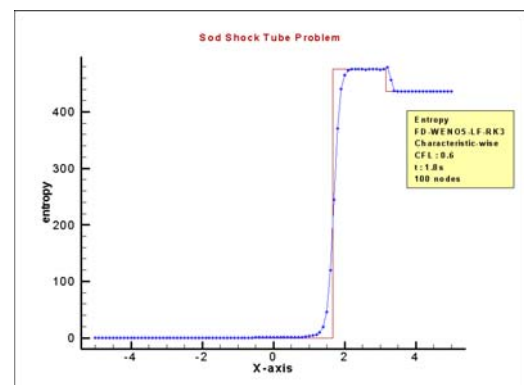
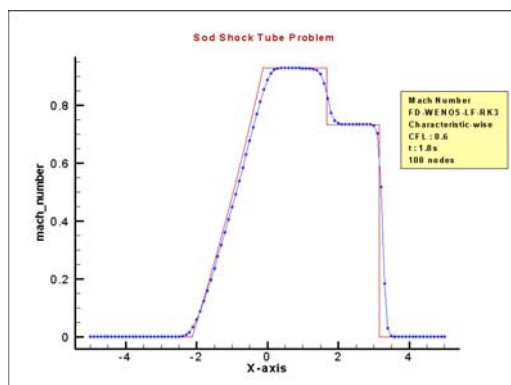
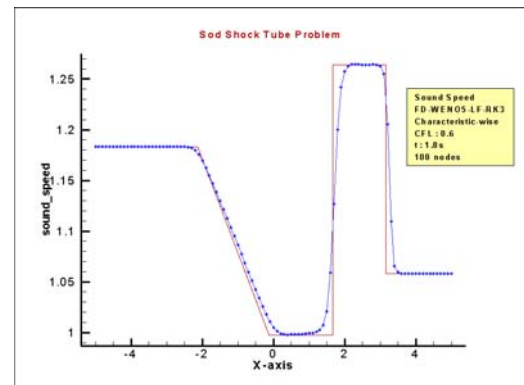
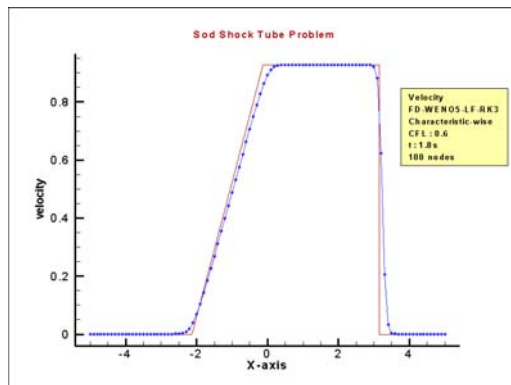
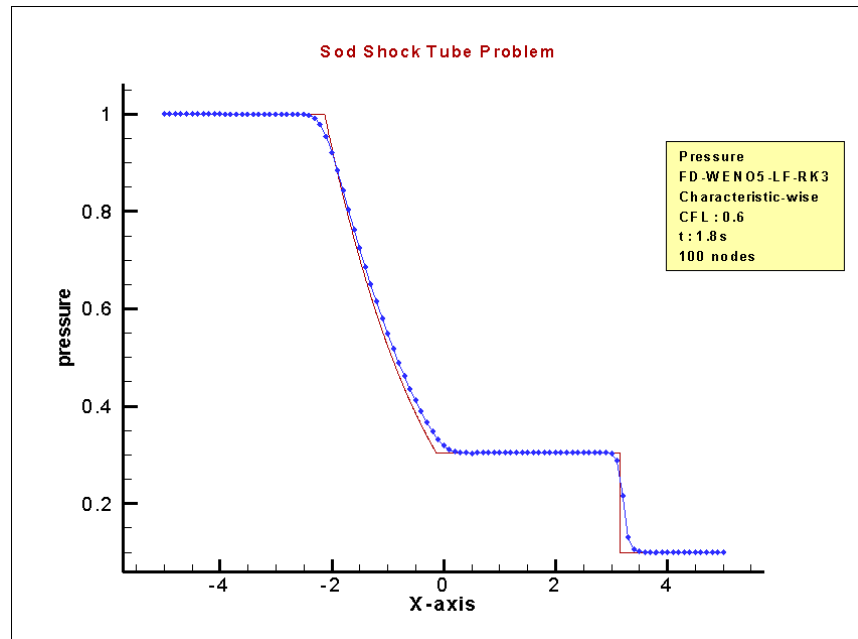


**Figure 9.34 : Sod Shock Tube Problem, Other Variables (FD)**

From the figure 9.34, which presents other variables; pressure, velocity, speed of sound, Mach number, and entropy, it can be claimed that FD type decomposition works well with WENO. However, there is a small amount of overshoot in entropy solution of the shock wave.

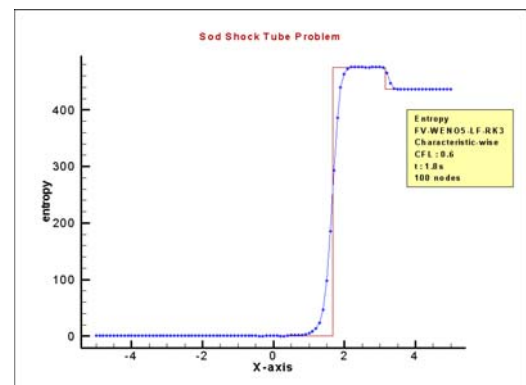
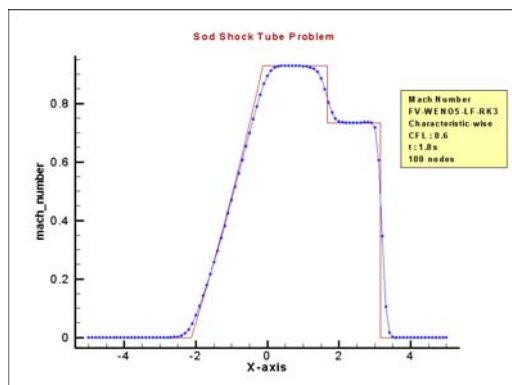
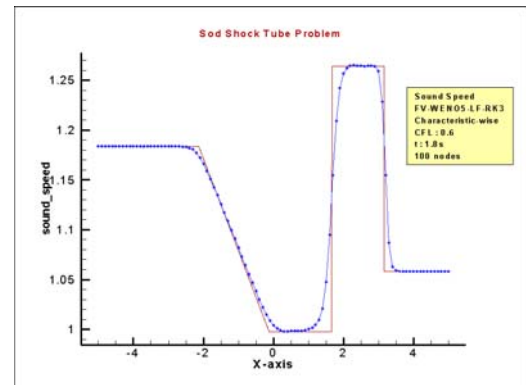
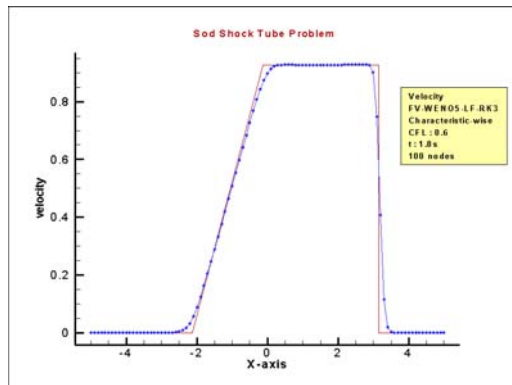
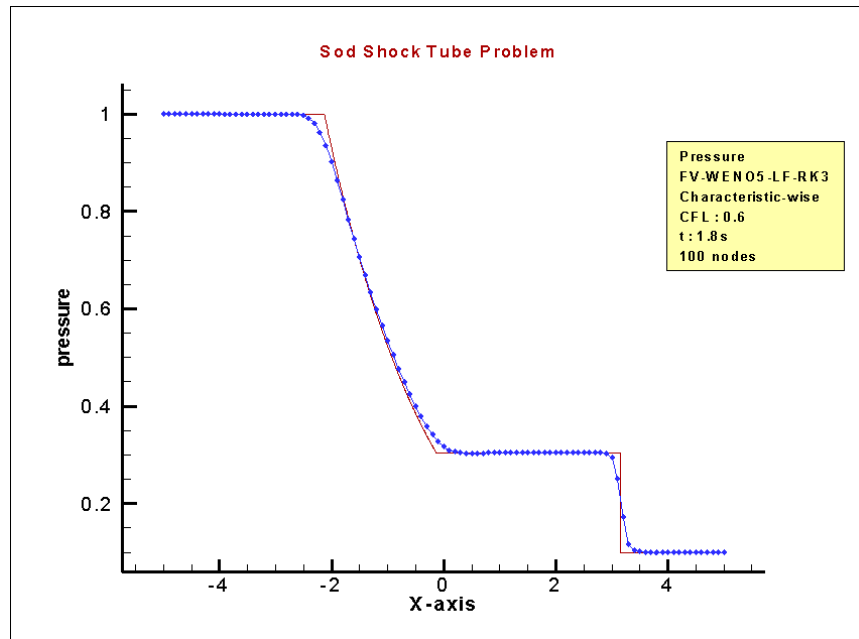


**Figure 9.35 : Sod Shock Tube Problem, Other Variables (FV)**

It is observed from the figure 9.35 that, FV type decomposition works properly with WENO as well.

### 9.2.2 Lax Shock Tube Problem

This is another Riemann shock tube problem with initial conditions as the following

$$\rho_L = 0.445, \quad u_L = 0.698, \quad p_L = 3.528 \quad (9.10a)$$

$$\rho_R = 0.5, \quad u_R = 0.0, \quad p_R = 0.571 \quad (9.10b)$$

Domain is  $[-5, 5]$ , and zero-order extrapolation boundary condition used at each end.

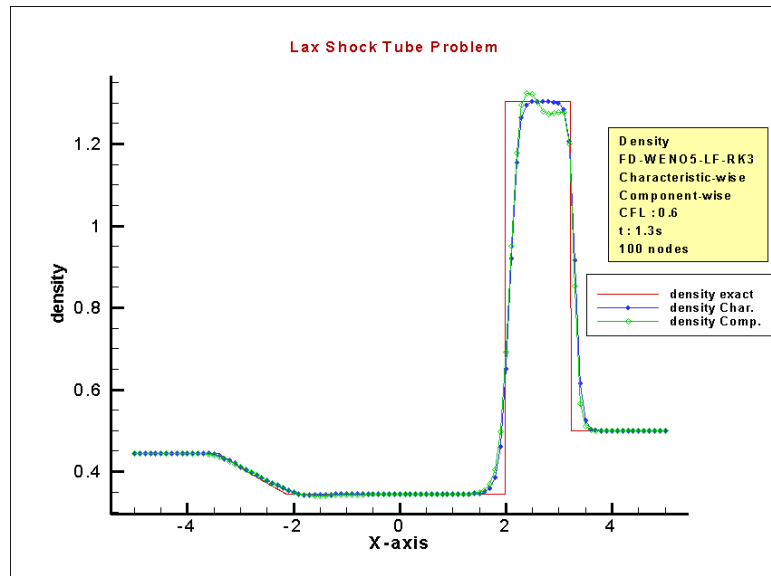


**Figure 9.36 :** Lax Shock Tube Problem (FD - Char. & Comp.)

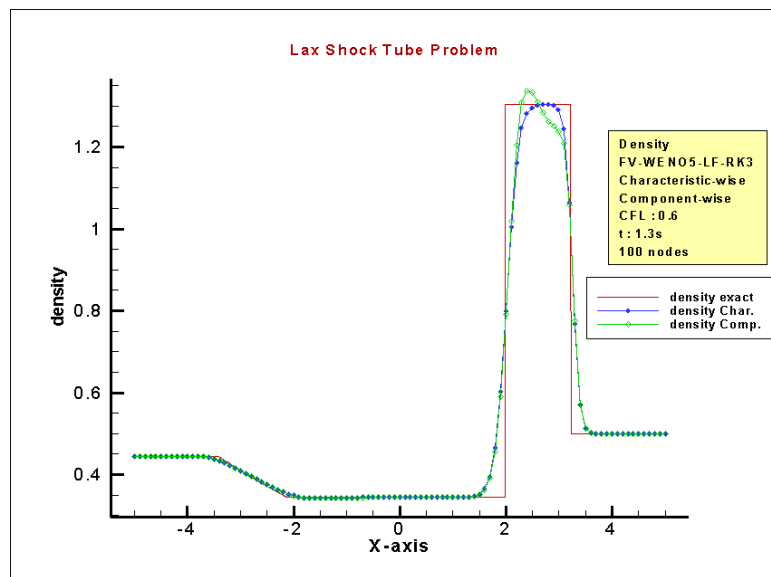


**Figure 9.37 :** Lax Shock Tube Problem (FV - Char. & Comp.)

From figures 9.36 and 9.37, it is observed that using a component-wise decomposition gives overestimations and underestimations especially between contact discontinuity and shock wave. However, characteristic-wise decomposition can catch the trend of exact solution accurately.

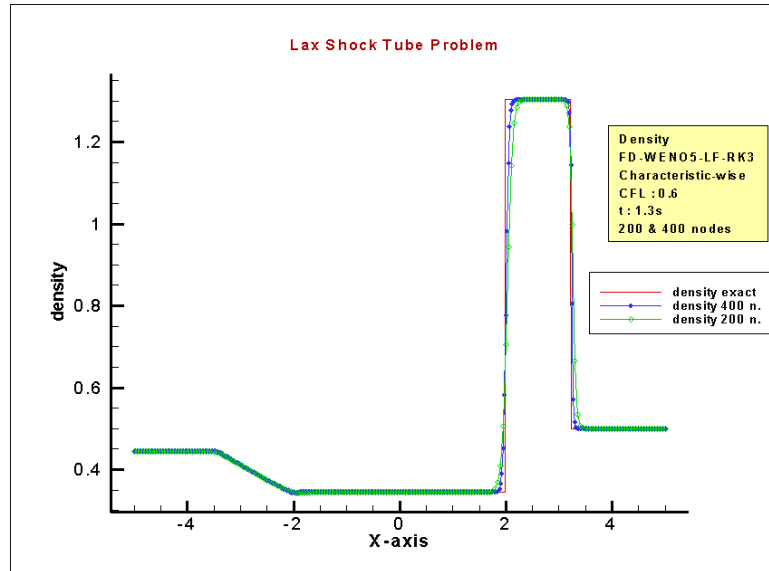


**Figure 9.38 :** Lax Shock Tube Problem (FD - 200 & 400 cells)

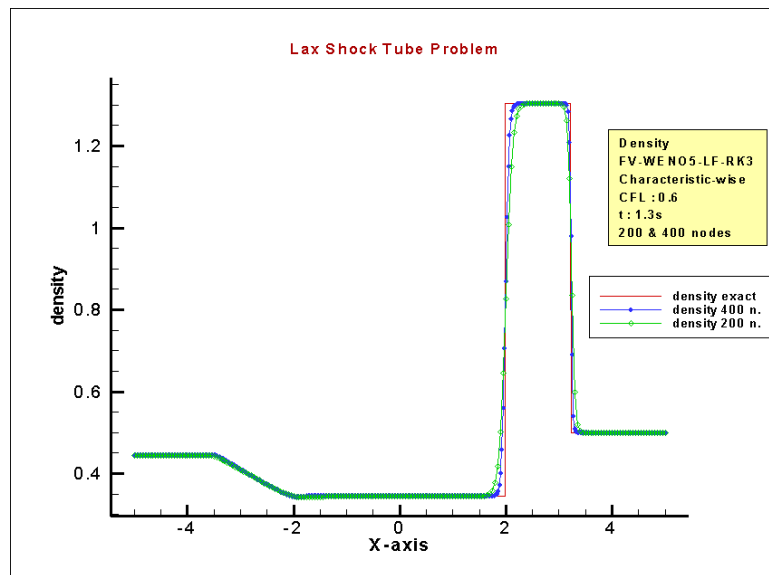


**Figure 9.39 :** Lax Shock Tube Problem (FV - 200 & 400 cells)

From figures 9.38 and 9.39 above, one can observe the difference between using 200 and 400 cells for FD and FV decompositions.

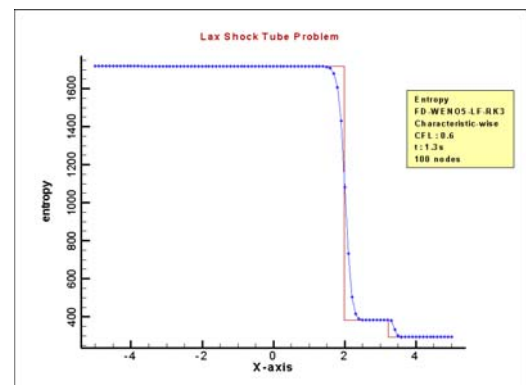
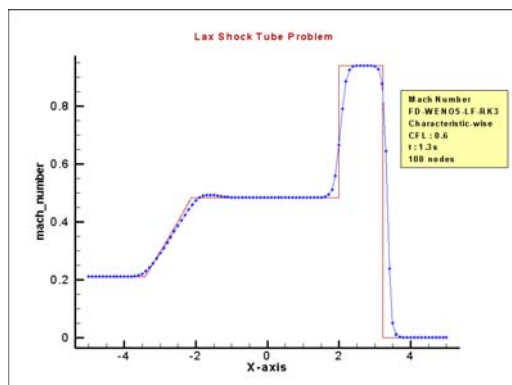
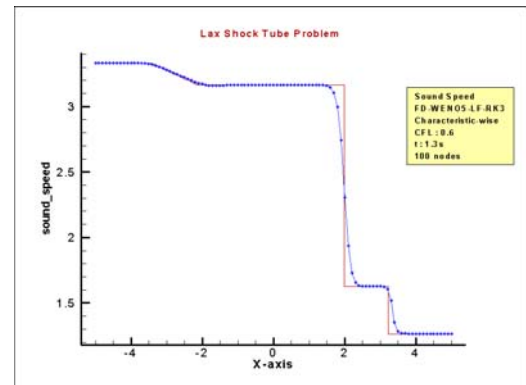
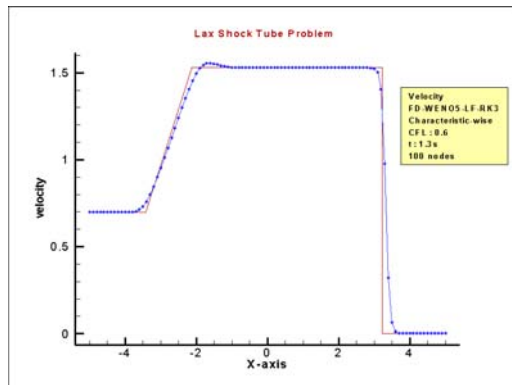
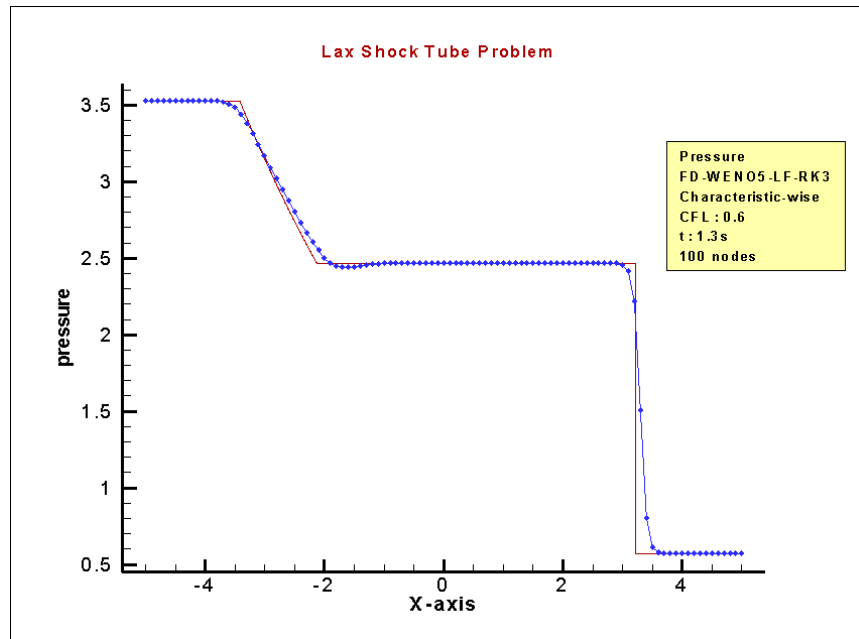


**Figure 9.40 : Lax Shock Tube Problem, Other Variables (FD)**

Figure 9.40 shows pressure, velocity, speed of sound, Mach number, and entropy solutions with FD decomposition for WENO5.

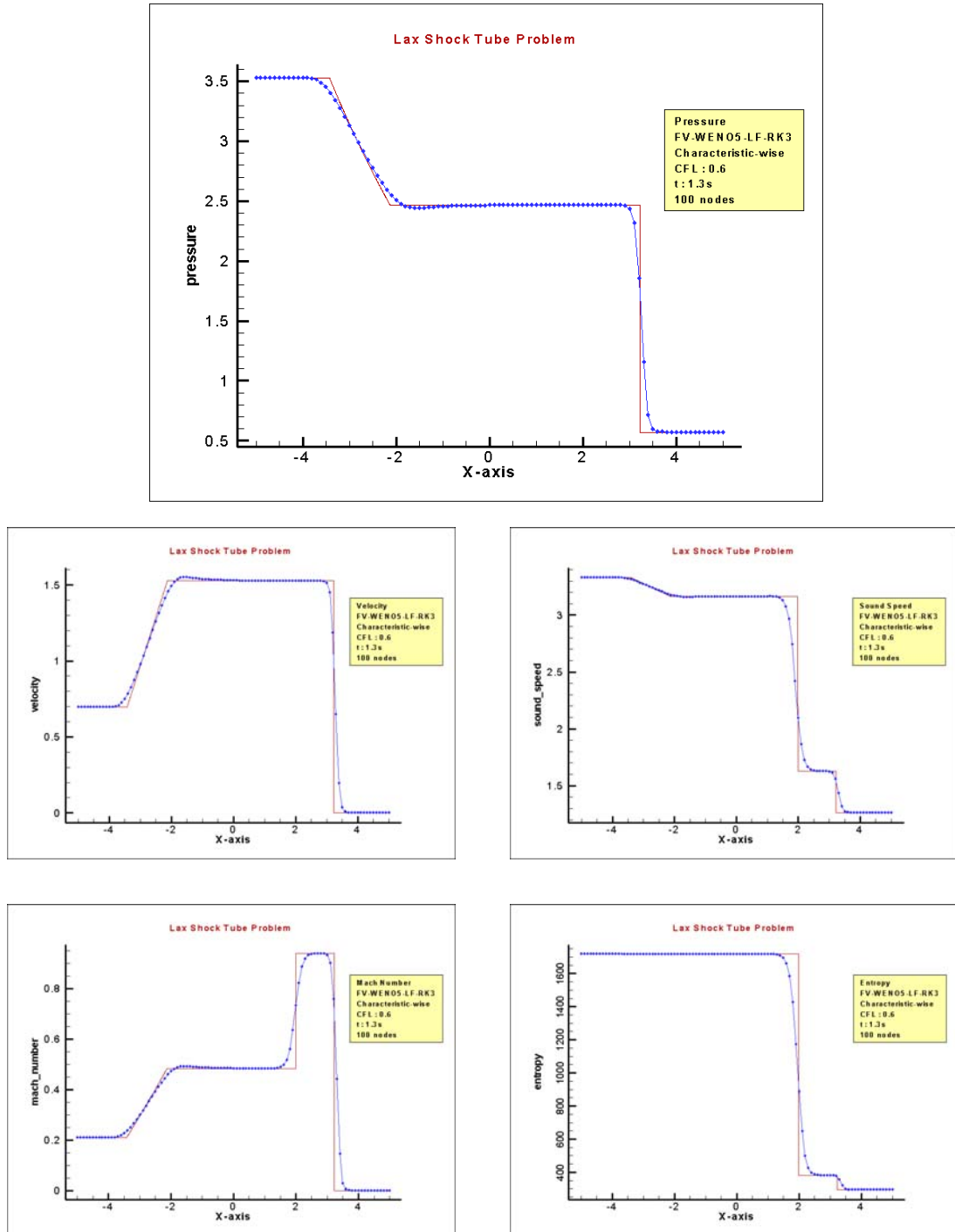


**Figure 9.41 : Lax Shock Tube Problem, Other Variables (FV)**

Figure 9.41 presents pressure, velocity, speed of sound, Mach number, and entropy solutions with FV decomposition for WENO5. From both of the previous two figures we can claim that, characteristic-wise WENO performs well for this test case. In addition to that, these solutions are in very good harmony with references.

### 9.2.3 Moving Shock Wave Interacting with Fluctuations (Shu's Problem)

Up to now the test cases include simple smooth region solutions and only shocks, for which shock resolution is the main concern and a second-order non-oscillatory scheme would give satisfactory results. There is a little advantage of using high order schemes with these kinds of problems. They are used in order to demonstrate the non-oscillatory property of this scheme. A higher order scheme would show its advantage when the solution contains both shocks and complex smooth region structures [13]. One example for this kind of flow structure is the shock wave interaction with entropy waves [27]. This kind of problem is also a model for shock/turbulence interaction [28].

Initial conditions used for this test case are

$$\rho_L = 3.857143, \quad u_L = 2.629369, \quad p_L = 10.33333 \quad \text{for } x < -4.0 \quad (9.11a)$$

$$\rho_R = 1 + \varepsilon \sin(5x), \quad u_R = 0.0, \quad p_R = 1.0 \quad \text{for } x \geq -4.0 \quad (9.11b)$$

Here  $\varepsilon = 0.2$ , if  $\varepsilon$  chosen to be zero, there will be a pure Mach 3 shock moving to the right. The domain is  $[-5, 5]$ , and zero-order extrapolation boundary condition used at each end. Exact (reference) solutions found with a very fine grid, consists of 1600 cells.

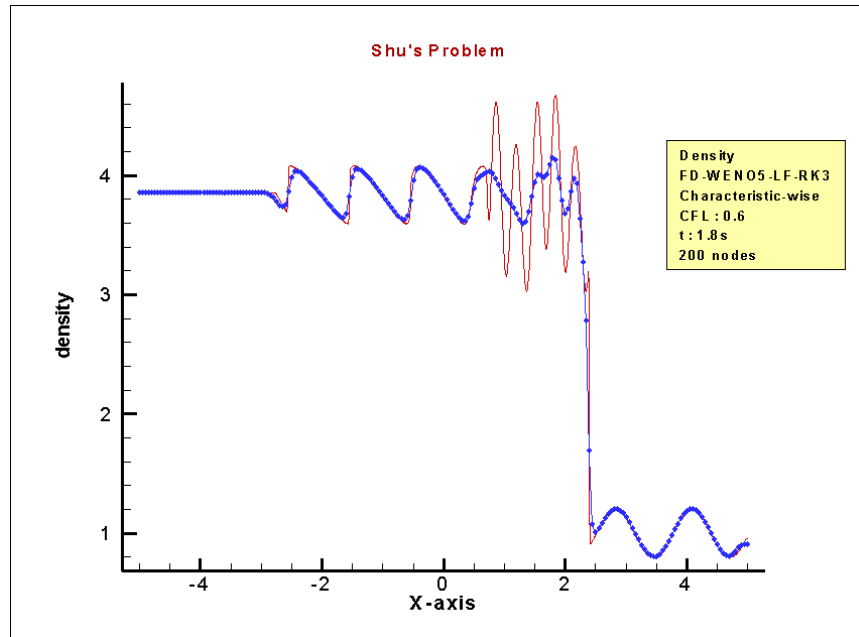


**Figure 9.42 : Shu's Problem (FD - 200 cells)**

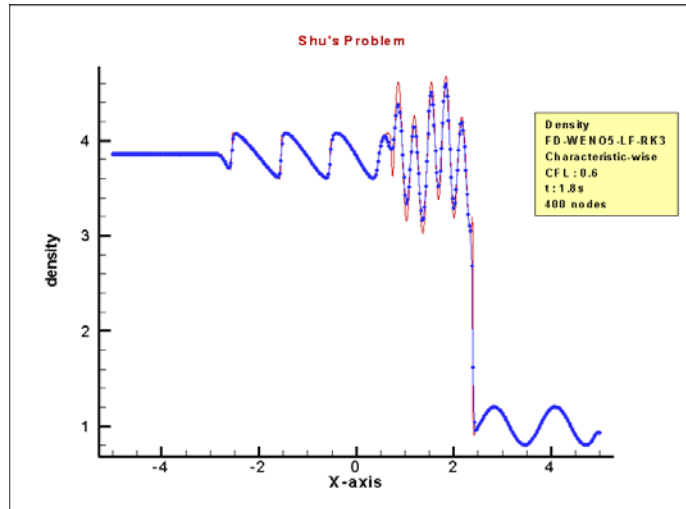


**Figure 9.43 :** Shu's Problem (FD - 400 cells)

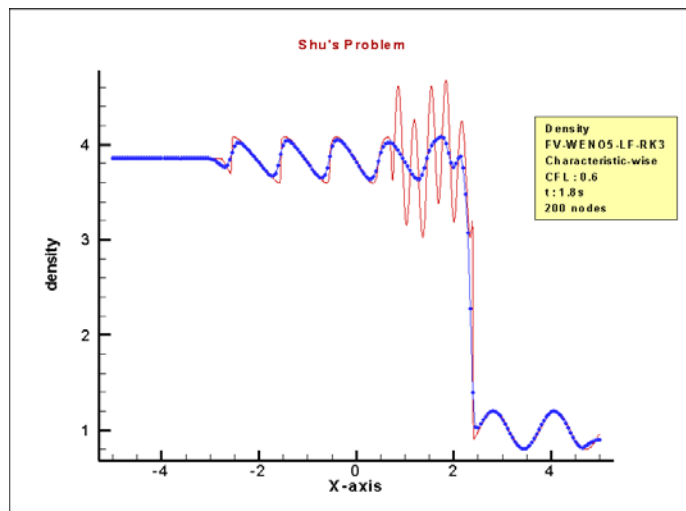


**Figure 9.44 :** Shu's Problem (FV - 200 cells)

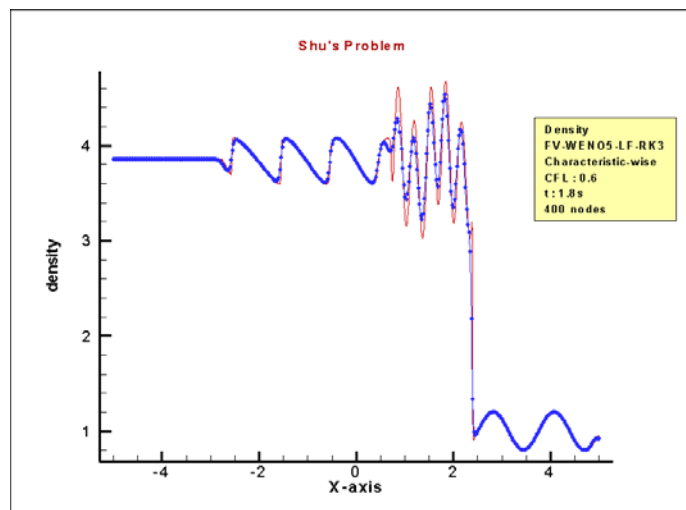


**Figure 9.45 :** Shu's Problem (FV - 400 cells)

From figure 9.42 to 9.45, the problem case is implemented to FD and FV decompositions with 200 and 400 cells. It is observed that, using 400 cells is sufficient to resolve the flow structure of shock/entropy wave interaction. In addition to that, using FD type decomposition gives only a slight improvement compared to FV type decomposition.

Figure 9.46, which is given below, is taken from the reference [13], this figure is for FD-WENO5 with characteristic-wise decomposition, using 200 nodes on the left, and using 400 nodes on the right. These solutions indicate that, the code generated for this thesis project works properly.

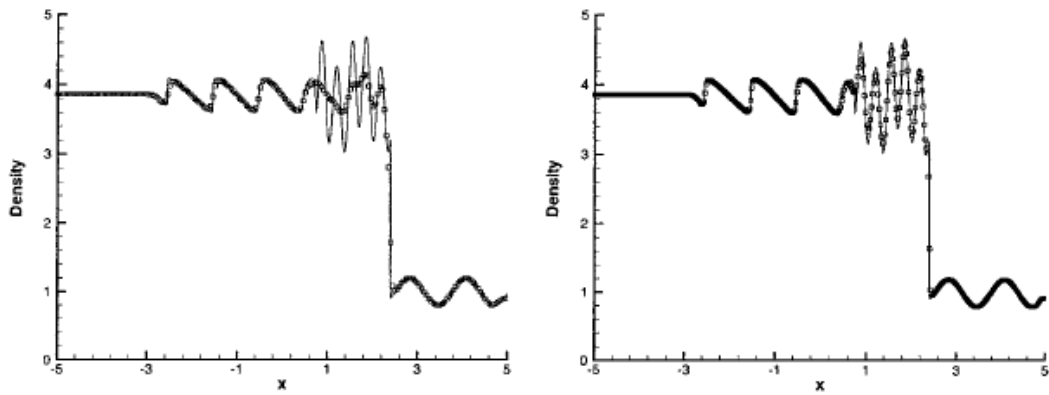


**Figure 9.46 :** Shu's Problem, taken from [13] (FD - 200 & 400 nodes)

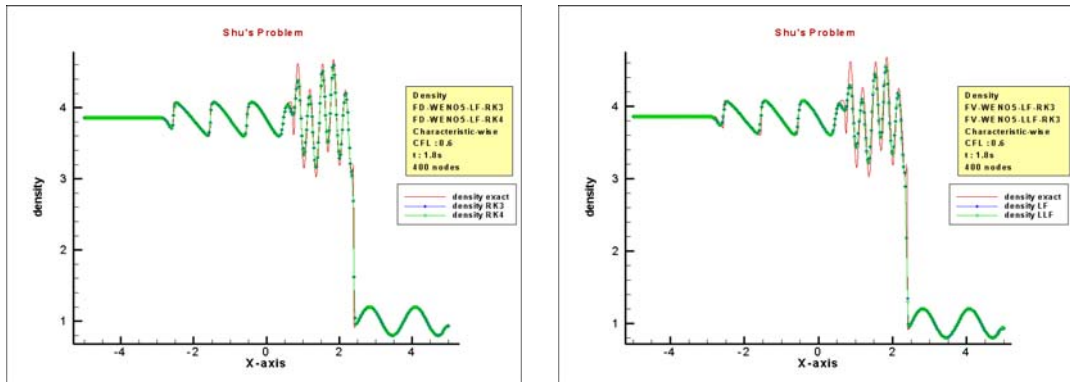


**Figure 9.47 :** Shu's Problem (FD - RK3 & RK4) and (FV - LF & LLF)

In figure 9.47 above, on the left, RK3 and RK4 is compared for FD decomposition, it turns out that there is not a significant difference between these two type time decompositions for this problem case. On the right, LF and LLF are compared for FV decomposition, a slight improvement is observed for LLF.

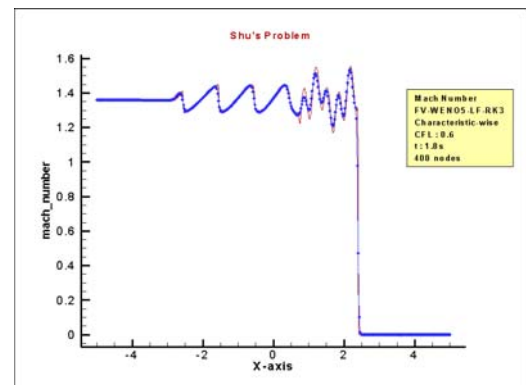
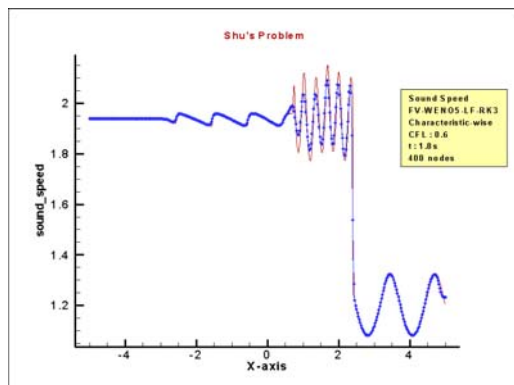
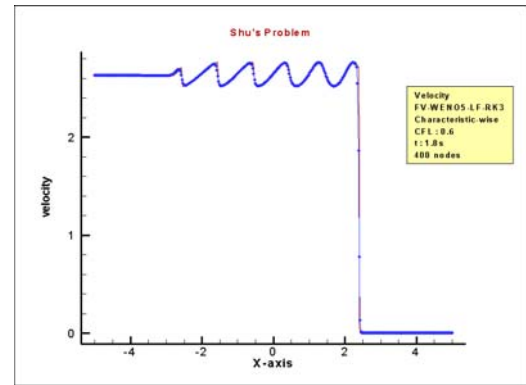
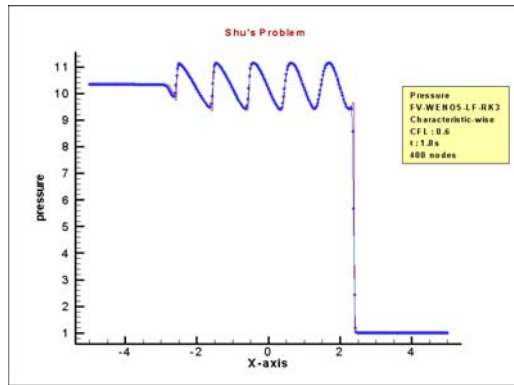
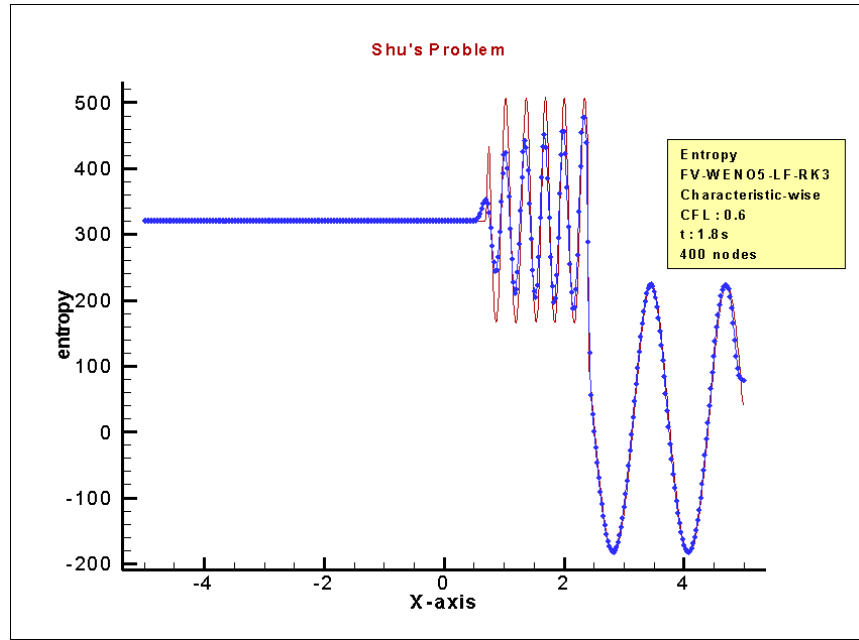


**Figure 9.48 : Shu's Problem, Other Variables (FV)**

Figure 9.48 presents other variables for FV type decomposition. In [29], second order TVD fluxes for WENO, instead of a first order monotone flux (LF), are compared with this kind of a problem case, which are FORCE, FLIC, HLLC, and WAF, and it is observed that WAF can capture entropy waves accurately.

### 9.2.4 Interaction of Two Blast Waves

This problem is suggested by Woodward and Colella as a test problem in [30]. The solution to this problem possesses drastic fluctuations under the impact of interactions, it is a good test case for stability of the scheme. The complex structure of the solution after the clash of the two blast waves demands a stable high-order method to capture the details of the solutions [31].

Initial conditions used for this test case are

$$\rho_L = 1.0, \quad u_L = 0.0, \quad p_L = 10^3 \quad \text{for} \quad 0 \leq x < 0.1 \quad (9.12a)$$

$$\rho_M = 1.0, \quad u_M = 0.0, \quad p_M = 10^{-2} \quad \text{for} \quad 0.1 \leq x < 0.9 \quad (9.12b)$$

$$\rho_R = 1.0, \quad u_R = 0.0, \quad p_R = 10^2 \quad \text{for} \quad 0.9 \leq x < 1.0 \quad (9.12c)$$

The domain is  $[0,1]$ , and wall boundary condition used at each end. Exact (reference) solutions found with a very fine grid, consists of 1600 cells.

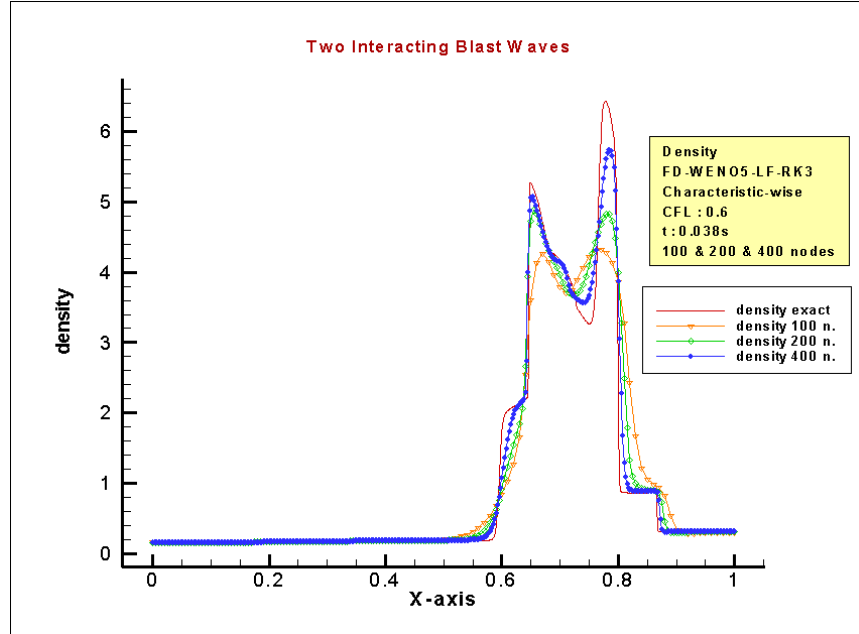


**Figure 9.49 :** Two Interacting Blast Waves Problem (FD - 100 & 200 & 400 cells)

In the figure 9.49, 100, 200, and 400 cells are used to solve the problem of two interacting blast waves for FD type decomposition. It can be seen that using 400 cells is sufficient for this problem case.

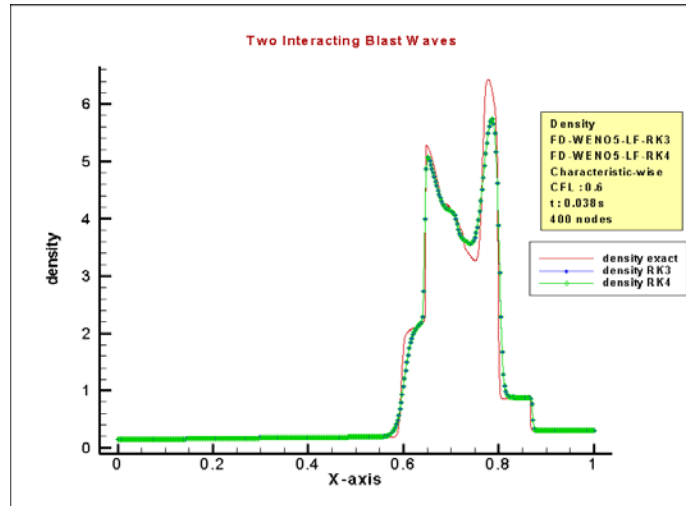


**Figure 9.50 :** Two Interacting Blast Waves Problem (FD - RK3 & RK4)

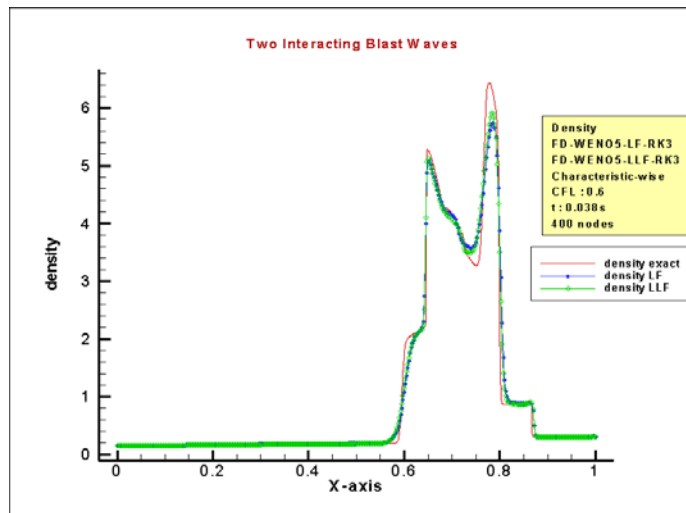


**Figure 9.51 :** Two Interacting Blast Waves Problem (FD - RK3 - LF & LLF)

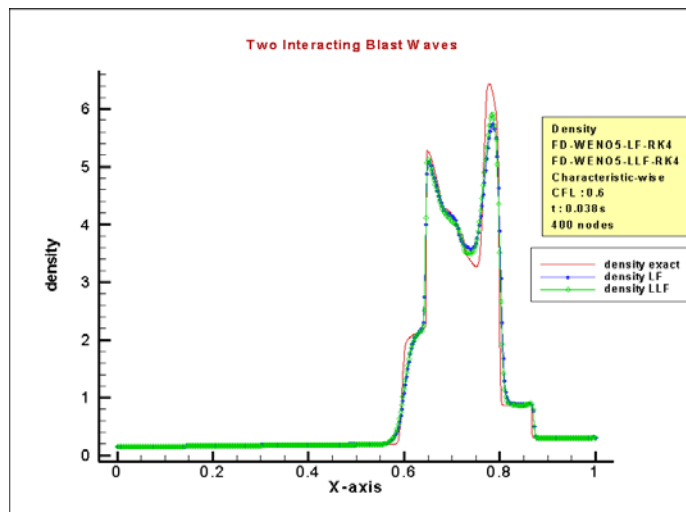


**Figure 9.52 :** Two Interacting Blast Waves Problem (FD - RK4 - LF & LLF)

In the figure 9.50, blast waves problem is implemented for RK3 and RK4, it is observed that using RK4 instead of RK3 gives only a slight improvement. In the figures 9.51 and 9.52, LF and LLF are implemented with RK3 and RK4, it is observed that using LLF instead of LF improves the strength of the scheme for this problem case.

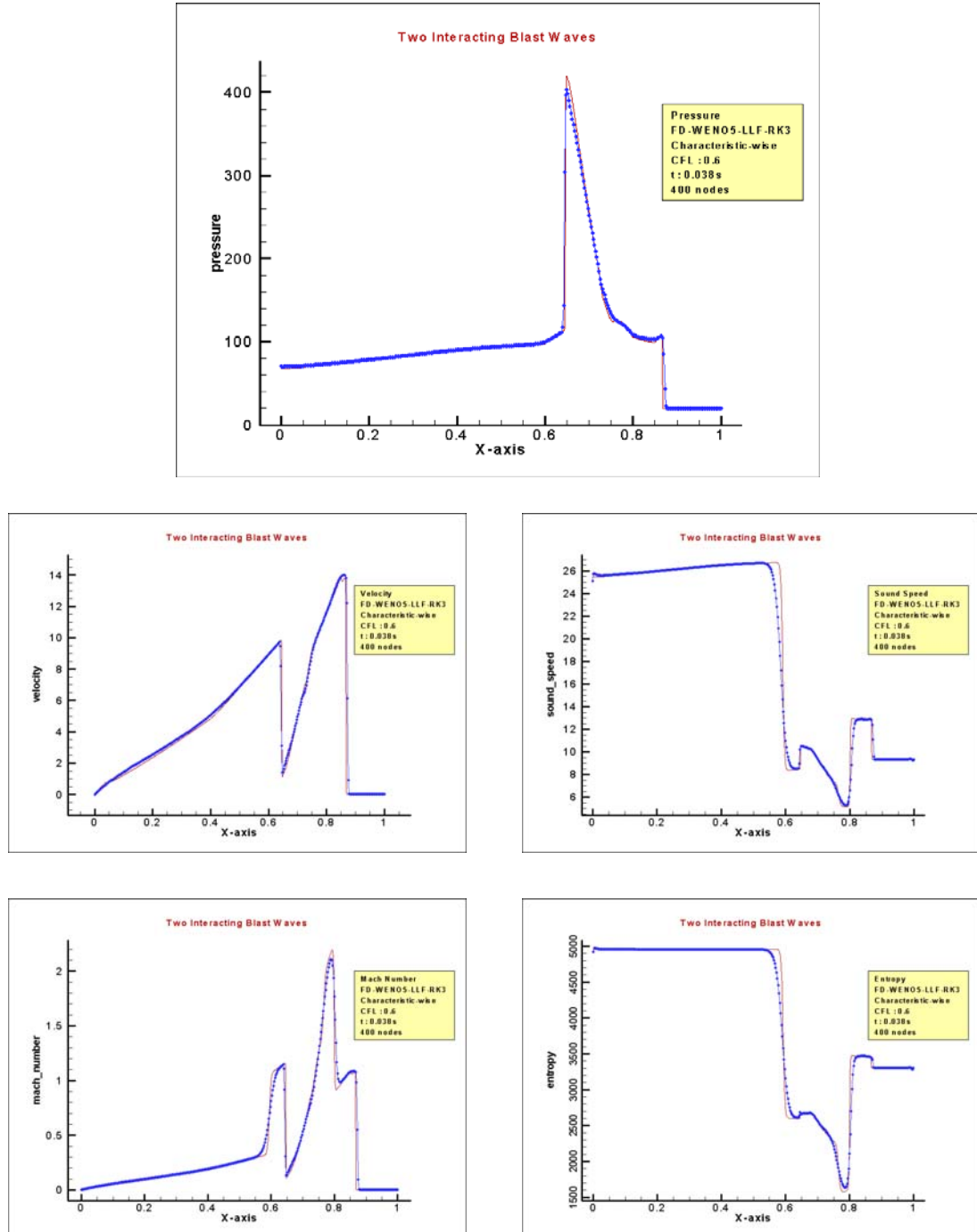


**Figure 9.53 :** Two Interacting Blast Waves Problem, Other Variables (FD - LLF)

In the figure 9.53, other variables are given for FD-WENO5-LLF-RK3 with 400 cells and characteristic-wise decomposition.

Implementation of FV decomposition for this problem case gives negative pressure error and explodes at  $\sim 0.026$  seconds. This is the time just before two waves interact with each other. This is because of the nature of high-order interpolations in such regions, where the values change drastically near a point of zero. In order to overcome this difficulty references [9,17] suggest locally reducing the order of accuracy at these points, i.e. to WENO3 and if problem persists to MUSCL. This technique is implemented for 2D FV version of the code, but not to 1D version. Order reduction technique is discussed in Appendix-B.

### 9.3 CPU Time Comparison for 1D Results

This subsection compares CPU times for one-dimensional results. The tables below give CPU times in seconds. To run the code a Laptop PC has been used, which has an AMD Athlon 64 3400+ (2200 MHZ) CPU. Code is compiled with Compaq Visual Fortran 6.6.C in Debug mode. Shu's problem is selected for the test case of CPU time comparisons. Also FD decomposition is used with alternative kind of reconstruction in addition to the classic one, which is discussed in Appendix-A, and it is denoted by Altr-FD. In table 9.9, characteristic-wise decomposition is used, CFL number is constant and 0.6, and the physical time of the problem is from 0.0 to 1.8 seconds. In table 9.10, component-wise decomposition is used with the same conditions.

**Table 9.9 : 1D CPU Time Comparison (Characteristic-wise)**

|                     |     | 400 cells | 1600 cells |          |
|---------------------|-----|-----------|------------|----------|
|                     |     | RK3       | RK3        | RK4      |
| Char. FD-WENO5      | LF  | 4,4375    | 71,6563    | 95,8438  |
| Char. Altr-FD-WENO5 |     | 3,4844    | 56,3281    | 75,4063  |
| Char. FV-WENO5      |     | 4,4844    | 72,5938    | 96,6719  |
| Char. FD-WENO5      | LLF | 4,6406    | 74,9375    | 100,5000 |
| Char. Altr-FD-WENO5 |     | 3,7031    | 59,7188    | 79,8906  |
| Char. FV-WENO5      |     | 4,6250    | 74,3125    | 98,8906  |

**Table 9.10 : 1D CPU Time Comparison (Component-wise)**

|                     |    |            |
|---------------------|----|------------|
|                     |    | 1600 cells |
|                     |    | RK3        |
| Comp. FD-WENO5      | LF | 47,3281    |
| Comp. Altr-FD-WENO5 |    | 36,2031    |
| Comp. FV-WENO5      |    | 47,2344    |

From the CPU time comparison given in this subsection, we observed that, using RK3 instead of RK4 gives approximately %25 CPU time advantage. Using LF instead of LLF is approximately %5 cheaper for FD, and %2,5 cheaper for FV. Using the alternative way of reconstruction for FD gives approximately %21 CPU time improvement. Finally, using a component-wise decomposition instead of a characteristic-wise decomposition gives approximately %51 CPU time advantage.

Throughout this section one-dimensional results for various scalar and vectorial test cases are discussed, it is observed that the code developed for one-dimensional WENO performs well and the results are identical with the results given in the literature. In the next section, we will consider two-dimensional scalar and vectorial test cases, and eventually the results of WENO around airfoil cross sections will be presented.

## 10. TWO DIMENSIONAL RESULTS AND DISCUSSION

This section gives two dimensional test case results for finite difference and finite volume WENO scheme. Only 5<sup>th</sup> order WENO is considered, with 3<sup>rd</sup> order TVD and 4<sup>th</sup> order non-TVD Runge-Kutta time discretizations, with Lax-Friedrichs flux-splitting for finite difference, and with Lax-Friedrichs numerical flux for finite volume cases. They are denoted by FD-WENO5-LF-RK# and FV-WENO5-LF-RK# respectively (local Lax-Friedrichs is denoted by LLF instead of LF). In addition to that, the Gaussian quadrature point rule, which discussed in section 5.2, is implemented for FV. In the next subsection, scalar problems are considered, which are linear advection equation and nonlinear Burger's equation, and in the subsequent subsection vectorial results are presented for Euler equations, which use component- and characteristic-wise reconstructions, and at the end of the section CPU time comparison is given. In addition, in the figures, number of nodes has to be treated as number of cells.

The code, which is generated for this thesis project, uses FD and FV decompositions for scalar problems and for two vectorial problems, which are vortex evolution and shock vortex interaction. However, the generation of FD code is stopped at that point, and further development has been done only for the FV case. Thus, the other vectorial problems are solved only for FV type WENO decomposition.

### 10.1 2D Results for Scalar Conservation Laws

First problem we solve is the following linear scalar problem, called the advection equation

$$u_t + au_x + bu_y = 0 \quad (10.1)$$

with  $a = 1$  and  $b = 1$ , which are the speed of the information (waves), and second one is the nonlinear scalar problem, called the Burger's equation, which is

$$u_t + f(u)_x + f(u)_y = 0 \quad (10.2)$$

where  $f(u)$  is a scalar function of  $u$ , and equal to  $\frac{1}{2}u^2$ .

### 10.1.1 Piecewise Smooth Problem with Advection and Burger's Equations

Here, the initial conditions are given as (10.3), the domain is  $x = y = [-2, 2]$ , and zero-order extrapolation boundary condition used at each end.

$$u(x, y, 0) = 1, \quad \text{for } x \leq 0 \quad (10.3a)$$

$$u(x, y, 0) = 0, \quad \text{for } x > 0 \quad (10.3b)$$

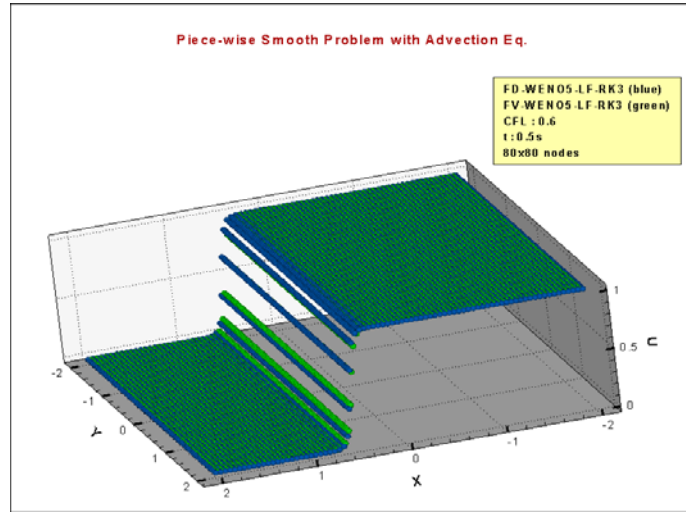


**Figure 10.1 :** Piece-wise Smooth Problem with Advection Equation (FD & FV)

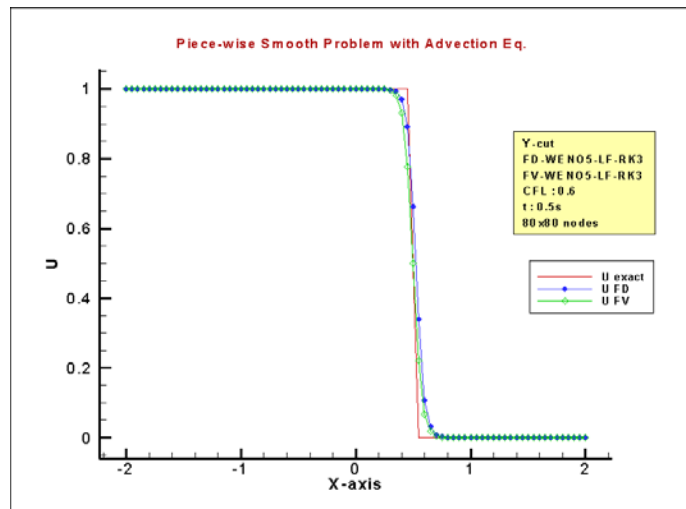


**Figure 10.2 :** Piece-wise Smooth Pr. with Advection Eq. (y-cut from fig. 10.1)

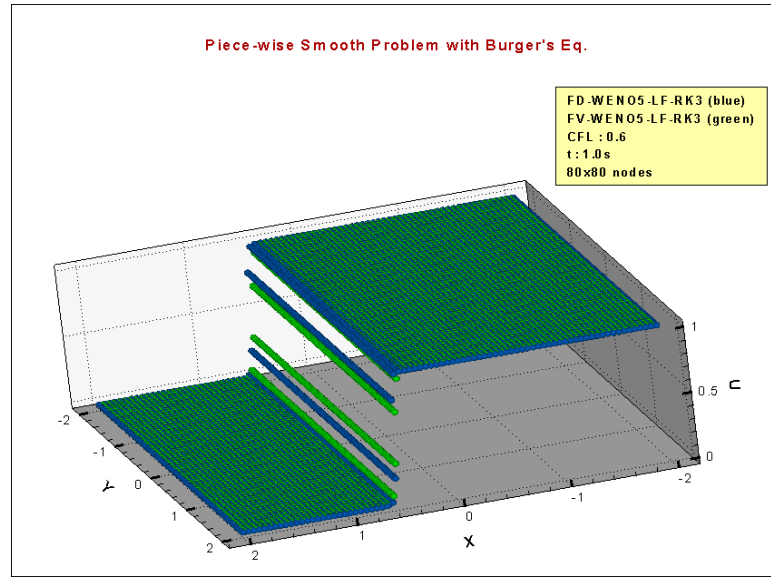


**Figure 10.3 :** Piece-wise Smooth Problem with Burger's Equation (FD & FV)

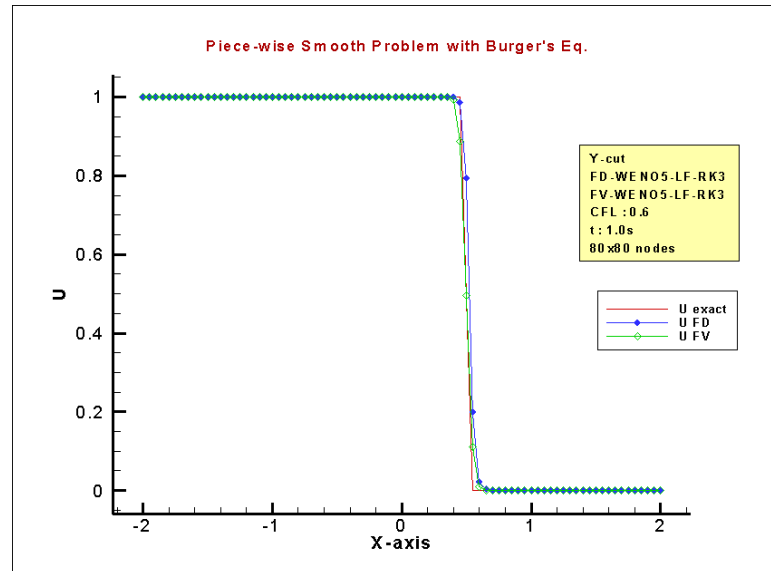


**Figure 10.4 :** Piece-wise Smooth Pr. with Burger's Eq. (y-cut from fig. 10.3)

Figures 10.1 to 10.4 show results for a piece-wise smooth problem for two-dimensional implementation of WENO5. Both FD and FV decompositions work appropriately like the one-dimensional case, which is discussed in previous section. For a deeper investigation, figures 10.2 and 10.4 give values at a cut from  $y$ -axis, where  $y = 0$ . It is observed here that, the results for two-dimensional case are similar to the one-dimensional case.

### 10.1.2 Sin-wave Problems with Advection and Burger's Equations

Two sinus waves are used, which have the initial  $\sin(\pi(x+y))$  for advection equation, and the initial  $0.3+0.7\sin\left(\frac{\pi}{2}(x+y)\right)$  for the Burger's equation. The domain is  $x = y = [-2,2]$ , and periodic boundary condition used at each end.

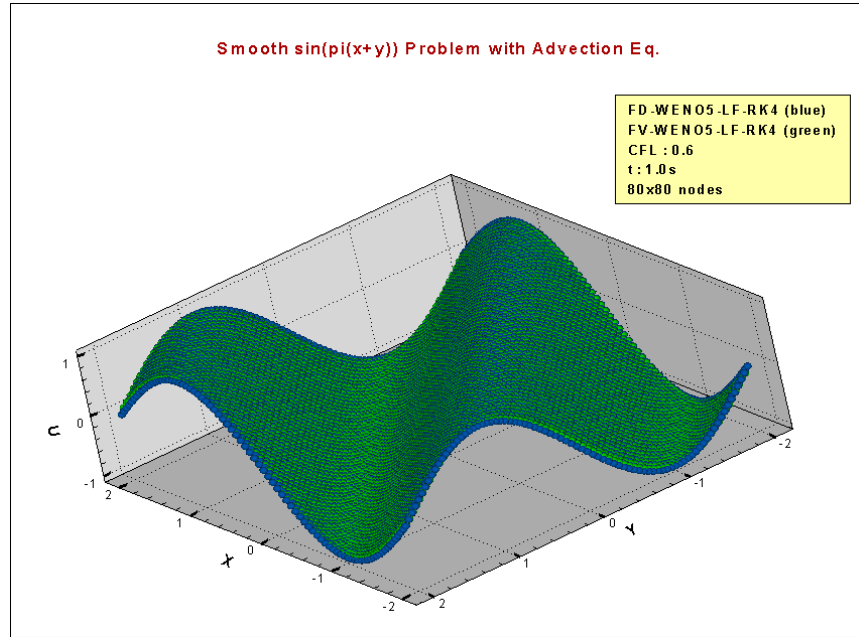


**Figure 10.5 :**  $\sin(\pi(x+y))$  Problem with Advection Equation (FD & FV)

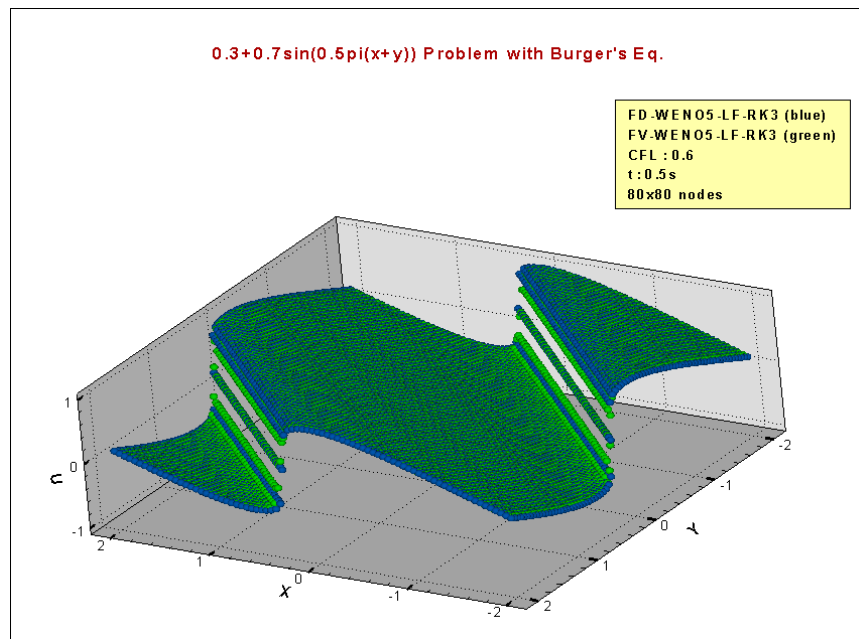


**Figure 10.6 :**  $0.3+0.7\sin\left(\frac{\pi}{2}(x+y)\right)$  Problem with Burger's Equation (FD & FV)

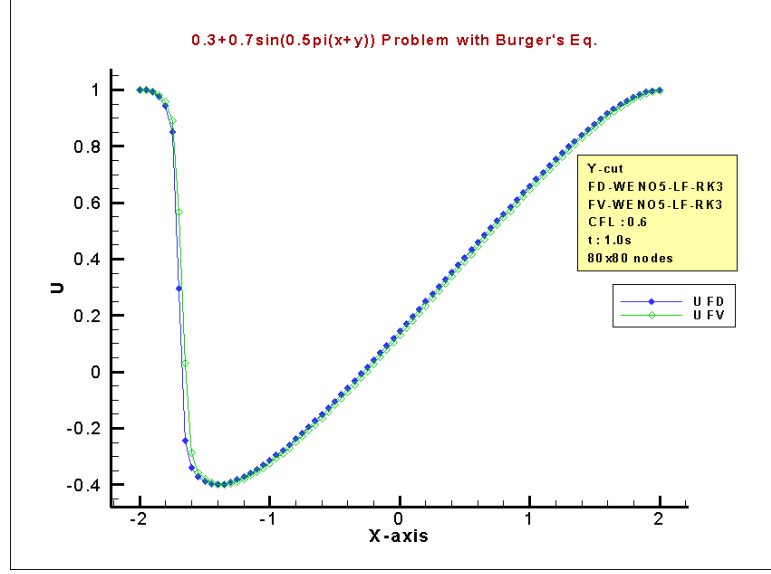


**Figure 10.7 :**  $0.3 + 0.7 \sin\left(\frac{\pi}{2}(x+y)\right)$  Pr. with Burger's Eq. ( y -cut from fig. 10.6)

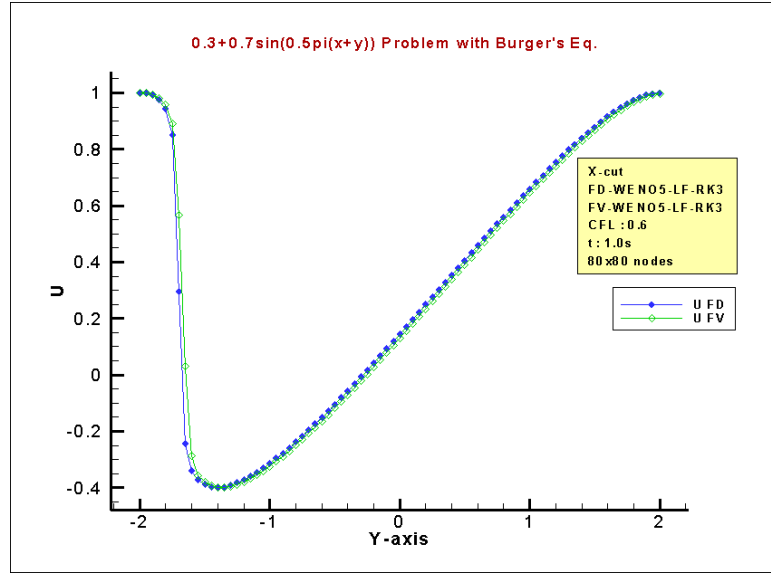


**Figure 10.8 :**  $0.3 + 0.7 \sin\left(\frac{\pi}{2}(x+y)\right)$  Pr. with Burger's Eq. ( x -cut from fig. 10.6)

Results of Sin-wave problems are presented from figures 10.5 to 10.8 above. It is observed that, both FV and FD type decompositions work properly. The Sin-wave used with Burger's equation forms a discontinuity near 0.5 seconds. In figures 10.7 and 10.8, y -cut for  $y = 0$ , and x -cut for  $x = 0$  are plotted. Exact solution for Burger's equation can be calculated by some techniques, but unfortunately it is not available in the scope of this thesis project.

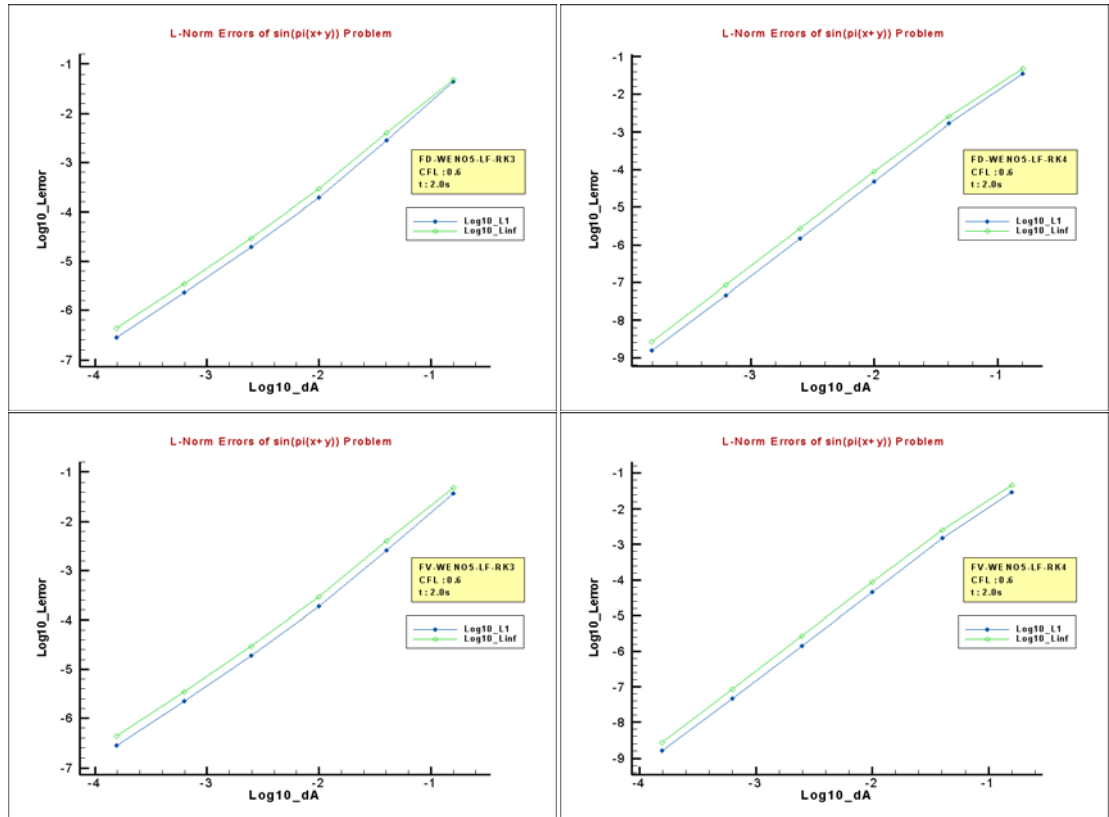


**Figure 10.9 :**  $L$  -Norm Err. of  $\sin(\pi(x + y))$  Pr. with Adv. Eq. (FD&FV-RK3&RK4)

**Table 10.1 :**  $L$  -Norm Errors of  $\sin(\pi(x + y))$  with Adv. Eq. (FD-LF-RK3)

| Nodes   | $L_1$ Error | $L_1$ Order | $L_\infty$ Error | $L_\infty$ Order |
|---------|-------------|-------------|------------------|------------------|
| 10x10   | 4,43E-02    |             | 4,91E-02         |                  |
| 20x20   | 2,78E-03    | 4,00        | 4,06E-03         | 3,60             |
| 40x40   | 1,93E-04    | 3,85        | 2,89E-04         | 3,82             |
| 80x80   | 1,93E-05    | 3,32        | 2,95E-05         | 3,29             |
| 160x160 | 2,25E-06    | 3,10        | 3,48E-06         | 3,08             |
| 320x320 | 2,75E-07    | 3,03        | 4,30E-07         | 3,02             |

**Table 10.2 :**  $L$  -Norm Errors of  $\sin(\pi(x + y))$  with Adv. Eq. (FD-LF-RK4)

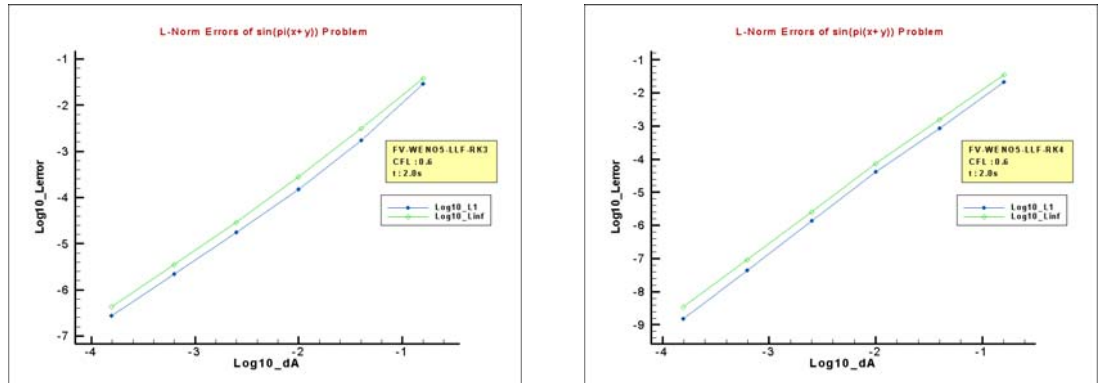
| Nodes   | $L_1$ Error | $L_1$ Order | $L_\infty$ Error | $L_\infty$ Order |
|---------|-------------|-------------|------------------|------------------|
| 10x10   | 3,54E-02    |             | 4,77E-02         |                  |
| 20x20   | 1,63E-03    | 4,44        | 2,52E-03         | 4,24             |
| 40x40   | 4,75E-05    | 5,10        | 8,81E-05         | 4,84             |
| 80x80   | 1,45E-06    | 5,03        | 2,76E-06         | 5,00             |
| 160x160 | 4,57E-08    | 4,99        | 8,59E-08         | 5,00             |
| 320x320 | 1,52E-09    | 4,91        | 2,64E-09         | 5,02             |

**Table 10.3 :**  $L$ -Norm Errors of  $\sin(\pi(x+y))$  with Adv. Eq. (FV-LF-RK3)

| Nodes   | $L_1$ Error | $L_1$ Order | $L_\infty$ Error | $L_\infty$ Order |
|---------|-------------|-------------|------------------|------------------|
| 10x10   | 3,69E-02    |             | 4,94E-02         |                  |
| 20x20   | 2,53E-03    | 3,87        | 4,05E-03         | 3,61             |
| 40x40   | 1,83E-04    | 3,78        | 2,88E-04         | 3,81             |
| 80x80   | 1,88E-05    | 3,28        | 2,94E-05         | 3,29             |
| 160x160 | 2,22E-06    | 3,08        | 3,48E-06         | 3,08             |
| 320x320 | 2,74E-07    | 3,02        | 4,29E-07         | 3,02             |

**Table 10.4 :**  $L$ -Norm Errors of  $\sin(\pi(x+y))$  with Adv. Eq. (FV-LF-RK4)

| Nodes   | $L_1$ Error | $L_1$ Order | $L_\infty$ Error | $L_\infty$ Order |
|---------|-------------|-------------|------------------|------------------|
| 10x10   | 2,95E-02    |             | 4,80E-02         |                  |
| 20x20   | 1,48E-03    | 4,32        | 2,51E-03         | 4,26             |
| 40x40   | 4,53E-05    | 5,03        | 8,77E-05         | 4,84             |
| 80x80   | 1,42E-06    | 4,99        | 2,77E-06         | 4,99             |
| 160x160 | 4,57E-08    | 4,96        | 8,63E-08         | 5,00             |
| 320x320 | 1,58E-09    | 4,86        | 2,66E-09         | 5,02             |

**Figure 10.10 :**  $L$ -Norm Errors  $\sin(\pi(x+y))$  with Adv. Eq. (FV-LLF-RK3&RK4)**Table 10.5 :**  $L$ -Norm Errors of  $\sin(\pi(x+y))$  with Adv. Eq. (FV-LLF-RK3)

| Nodes   | $L_1$ Error | $L_1$ Order | $L_\infty$ Error | $L_\infty$ Order |
|---------|-------------|-------------|------------------|------------------|
| 10x10   | 2,90E-02    |             | 3,85E-02         |                  |
| 20x20   | 1,70E-03    | 4,09        | 3,13E-03         | 3,62             |
| 40x40   | 1,49E-04    | 3,51        | 2,82E-04         | 3,47             |
| 80x80   | 1,76E-05    | 3,08        | 2,93E-05         | 3,27             |
| 160x160 | 2,18E-06    | 3,01        | 3,48E-06         | 3,08             |
| 320x320 | 2,72E-07    | 3,00        | 4,29E-07         | 3,02             |

**Table 10.6 :**  $L$ -Norm Errors of  $\sin(\pi(x+y))$  with Adv. Eq. (FV-LLF-RK4)

| <b>Nodes</b> | <b><math>L_1</math> Error</b> | <b><math>L_1</math> Order</b> | <b><math>L_\infty</math> Error</b> | <b><math>L_\infty</math> Order</b> |
|--------------|-------------------------------|-------------------------------|------------------------------------|------------------------------------|
| 10x10        | 2,16E-02                      |                               | 3,55E-02                           |                                    |
| 20x20        | 8,30E-04                      | 4,70                          | 1,57E-03                           | 4,50                               |
| 40x40        | 4,01E-05                      | 4,37                          | 7,24E-05                           | 4,44                               |
| 80x80        | 1,33E-06                      | 4,91                          | 2,59E-06                           | 4,80                               |
| 160x160      | 4,37E-08                      | 4,93                          | 9,01E-08                           | 4,84                               |
| 320x320      | 1,47E-09                      | 4,89                          | 3,44E-09                           | 4,71                               |

Accuracy test results are given from table 10.1 to 10.6 for the  $u_0(x,y) = \sin(\pi(x+y))$  problem. These results are obtained at 2 seconds, which is the half of the period for the domain. The convergence results are similar to one-dimensional case for FD and FV type decompositions, as expected. As already discussed before, FV type decomposition uses two point Gaussian integration rule for two dimensional case. Comparing table 10.3 to 9.3, and 10.4 to 9.4, indicates that the converged numbers are essentially similar. Eventually, it is observed that, for RK4, using LLF instead of LF results in  $L_\infty$  order sacrifice.

## 10.2 2D Results for Systems of Conservation Laws

In this subsection, results of two-dimensional conservation laws for Euler equations are presented. Vortex evolution, shock vortex interaction, double Mach reflection, forward facing step, flow past a cylinder, flow about NACA0012 airfoil cross-section, and planar shock past NACA0018 airfoil cross-section problems are considered.

### 10.2.1 Vortex Evolution Problem

This problem illustrates the power of high order methods clearly. The initial condition of the flow domain is given by primitive variables as (10.4), this condition creates a diagonal mean flow. Afterwards, we add to this mean flow, an isentropic vortex (10.5) (perturbations in velocities and the temperature  $T = \frac{p}{\rho}$ , no perturbation in the entropy  $S = \frac{p}{\rho^\gamma}$ ).

$$\rho = 1.0, \quad u = 1.0, \quad v = 1.0, \quad p = 1.0 \quad (10.4)$$

$$\delta u = -\bar{y} \frac{\varepsilon}{2\pi} e^{0.5(1-r^2)} \quad (10.5a)$$

$$\delta v = \bar{x} \frac{\varepsilon}{2\pi} e^{0.5(1-r^2)} \quad (10.5b)$$

$$\delta T = -\frac{(\gamma-1)\varepsilon^2}{8\gamma\pi^2} e^{1-r^2} \quad (10.5c)$$

$$\delta S = 0 \quad (10.5d)$$

where  $(\bar{x}, \bar{y}) = (x-5, y-5)$ ,  $r^2 = \bar{x}^2 + \bar{y}^2$ , and the vortex strength  $\varepsilon = 5.0$ .

The domain is  $x = y = [0, 10]$ , and periodic boundary condition used at each end to perform a long time simulation.

It is clear that the exact solution of the Euler equation with the above initial and boundary conditions is just the passive convection of the vortex with the mean velocity [1].

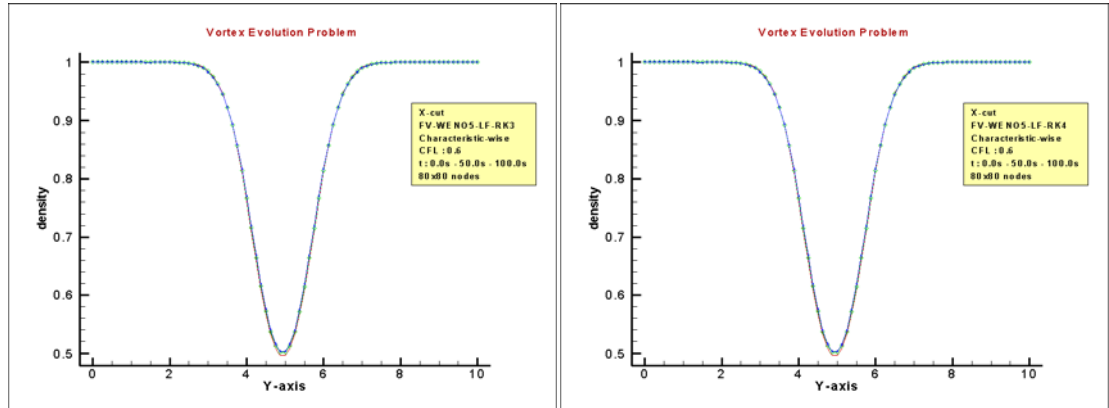


**Figure 10.11 :** Vortex Evolution Problem in Density Var. (FV-LF-RK3&RK4)

In figure 10.11 above, the results of vortex evolution problem compared for  $t = 50$  and  $t = 100$  seconds with characteristic-wise FV decomposition. The exact solution is the  $t = 0$  second that is because of periodicity and passive convection. Both RK3 and RK4 give very similar results, also the comparison can be seen from figure 10.12 in more detail. Solving the problem for  $t = 100$  seconds means 10 periods for this domain. This means a long time simulation, and the strength of a high order scheme comes out at this point, figure 10.13 compares TVD and WENO schemes for this problem case, which is taken from reference [2].

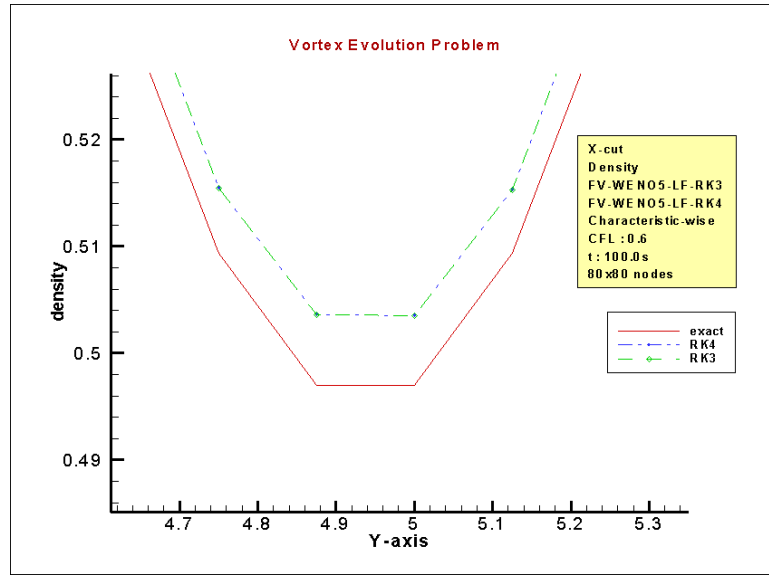


**Figure 10.12 :** Vortex Evolution Problem in Density Var. (FV-LF-RK3&RK4)

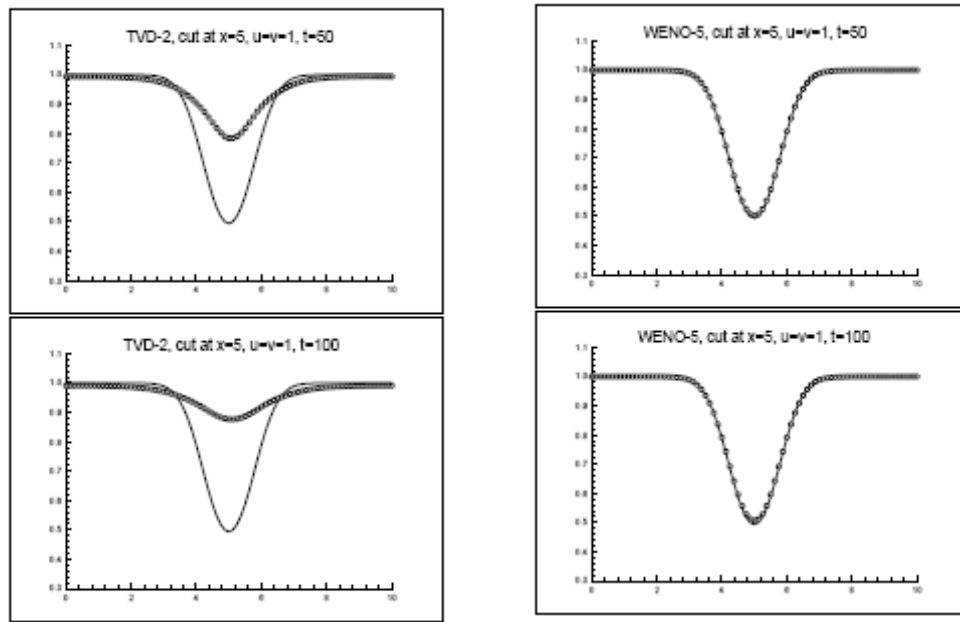


**Figure 10.13 :** Vortex Evolution Problem, taken from [2]

In figure 10.13 above, WENO scheme is compared to TVD scheme for the same problem discussed in this subsection. This implementation uses the same number of grid points and the domain with our implementation. One can observe that, using a high order scheme is much stronger than a low order one for long time simulations. In figure 10.14 below, FD type decomposition is implemented and results are given for one time period with  $dt = 1.0s$ .

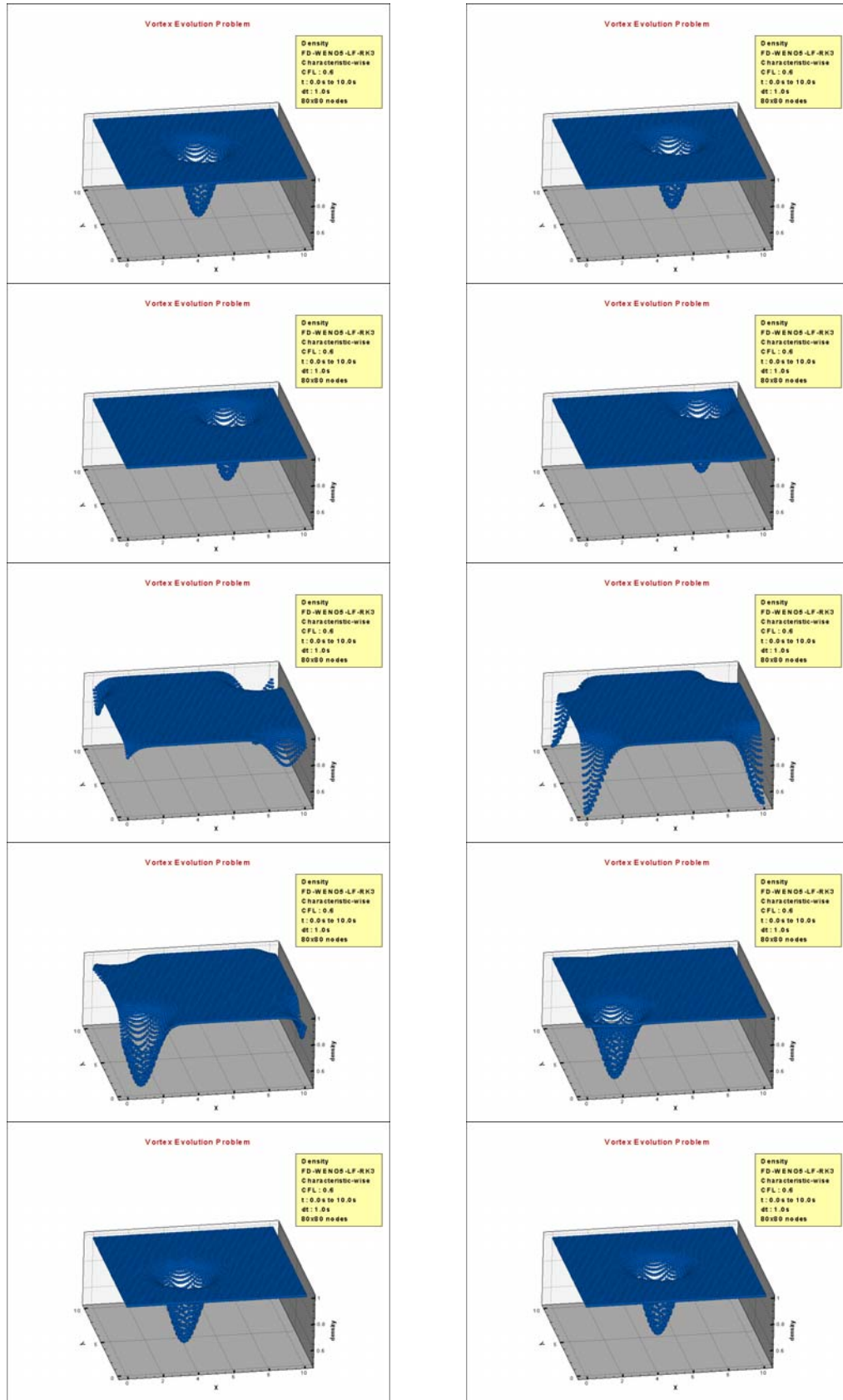


**Figure 10.14 :** Vortex Evolution Problem in Density Var.  $dt = 1.0$  s (FD-LF-RK3)

The results given for this test case is using uniform grid so far. But what happens if we use a non-uniform grid with the coefficients given in section 6 or using a coordinate transformation, which is given in section 8? These cases are investigated in this section and the results are presented below.

First of all, the non-uniform coefficients and the coordinate transformation are used for a uniform grid in order to prove that they are working correctly. The case is FV-WENO5-LF-RK3 with 80x80 grid domain.

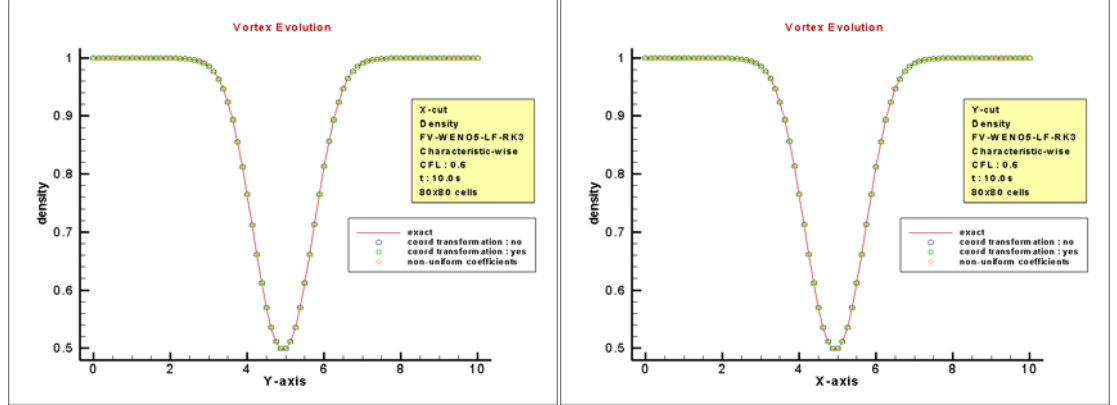


**Figure 10.15 :** Vortex Evolution Problem Uniform Grid Comparison

In figure 10.15 above,  $x$ - and  $y$ -cuts at the middle of the domain are given for one period of evolution. The problem case is solved for uniform grid with and without coordinate transformation and with non-uniform coefficients, they all give exactly the same results as expected.

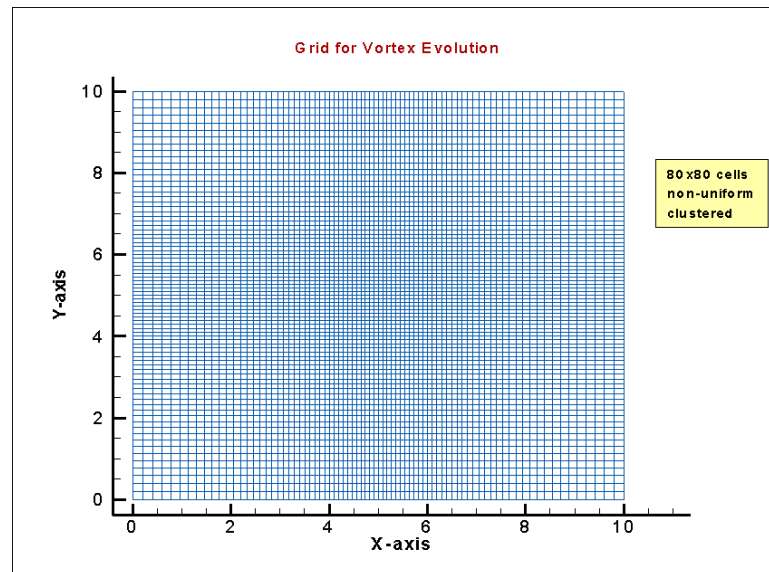


**Figure 10.16 :** Grid for Vortex Evolution Problem (Clustered Case-1)

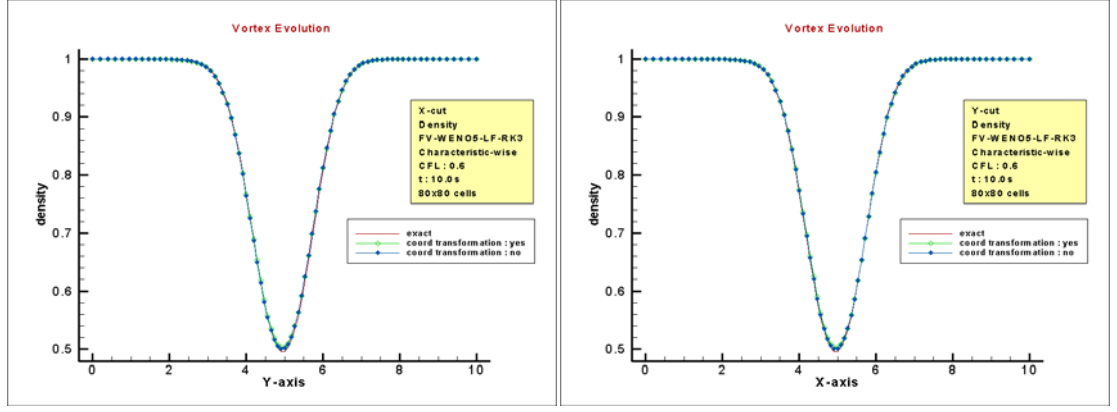


**Figure 10.17 :** Vortex Evolution Problem Non-Uniform Grid Comparison (Case-1)

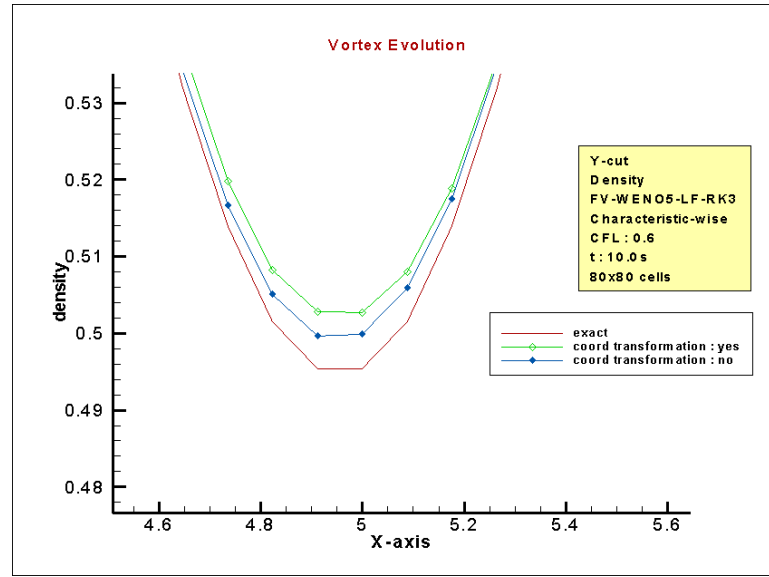


**Figure 10.18 :** Non-Uniform Grid Comparison (Figure 10.17 zoomed in)

In figure 10.16 the non-uniform clustered grid is presented and it is denoted by Case-1. For grid clustering the analytical expressions given in [23, page337] are used, and Case-1 uses a stretching parameter of 3. In figure 10.17  $x$ - and  $y$ -cuts at the middle of the domain are given for one period of evolution, and figure 10.18 presents a zoomed-in version for  $y$ -cut. The comparisons show that non-uniform grids introduce dissipation to the solution. Using a coordinate transformation introduces more dissipation compared to non-uniform coefficients. Comparing figure 10.18 with 10.12 shows that, dissipation for 1 period for this non-uniform grid is approximately equal to the dissipation for 10 periods for a uniform grid domain. However, the diameter of the vortex, grid spacing deviation, and direction of the vortex's motion have to be taken into account for that comparison.

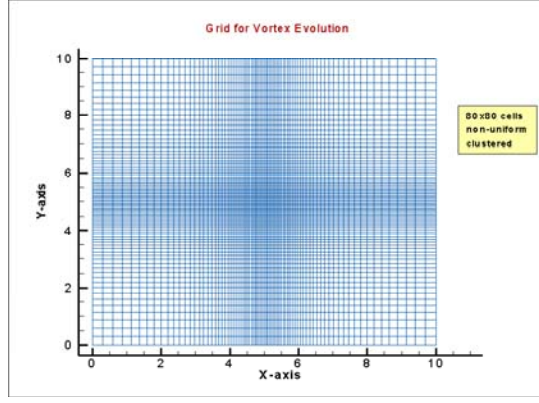


**Figure 10.19 :** Grid for Vortex Evolution Problem (Clustered Case-2)

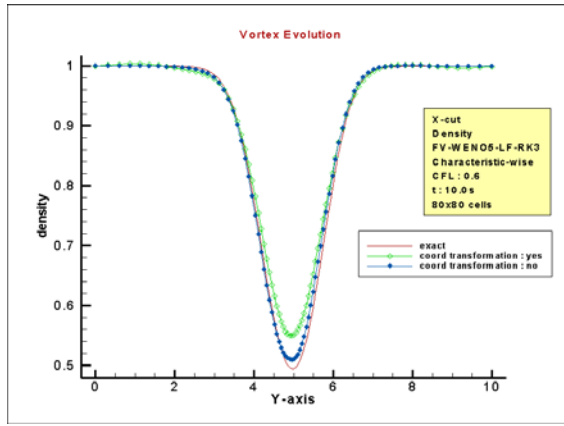


**Figure 10.20 :** Non-Uniform Grid Comparison ( $x$ -cut Case-2)

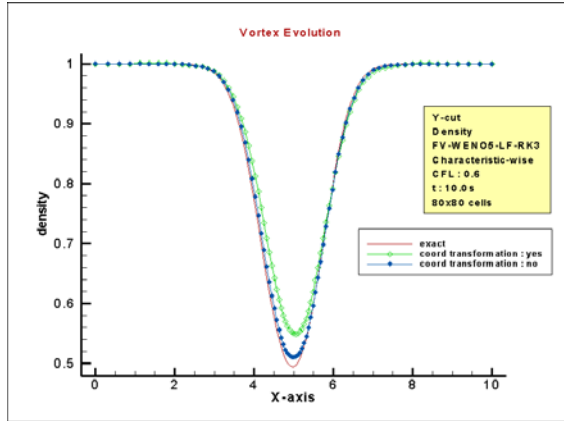


**Figure 10.21 :** Non-Uniform Grid Comparison ( $y$ -cut Case-2)

From figure 10.19 to 10.21 the grid and comparisons for this non-uniform grid case is given. Case-2 uses a stretching parameter of 5. Result comparisons are identical to Case-1. It can be seen that coordinate transformation introduces more dissipation than non-uniform coefficients, and that may be a consequence of using second order centered metric calculations, more investigations are needed here.

### 10.2.2 Shock Vortex Interaction Problem

High order methods resolve the vortex and the interaction of vortex with the shock wave better than low order methods and have some advantages for this problem case.

The domain is  $x = [0, 2]$ ,  $y = [0, 1]$ . Wall boundary condition is used for upper and lower boundaries, and zero-order extrapolation boundary condition is used for inlet and outlet.

This problem illustrates interaction between stationary shock and a vortex. A Mach 1.1 ( $M_s = 1.1$ ) shock is positioned at  $x = 0.5$  and normal to the  $x$ -axis. Initial left state variables are

$$\rho_L = 1.0, \quad u_L = \sqrt{\gamma}, \quad v_L = 0.0, \quad p_L = 1.0 \quad (10.6)$$

A small vortex superposed to the flow left to the shock and centers at  $(x_c, y_c) = (0.25, 0.5)$ . Afterwards, the vortex is described as perturbation to the velocities and the temperature  $T = \frac{p}{\rho}$ , and perturbation in the entropy  $S = \ln \frac{p}{\rho^\gamma}$  of the mean flow, and we denote it by

$$\delta u = \varepsilon \tau e^{\alpha(1-\tau^2)} \sin \theta \quad (10.7a)$$

$$\delta v = -\varepsilon \tau e^{\alpha(1-\tau^2)} \cos \theta \quad (10.7b)$$

$$\delta T = -\frac{(\gamma-1)\varepsilon^2 e^{2\alpha(1-\tau^2)}}{4\alpha\gamma} \quad (10.7c)$$

$$\delta S = 0 \quad (10.7d)$$

and pressure can be found from the following equation

$$p = \frac{T^{\gamma/(\gamma-1)}}{(e^S)^{1/(\gamma-1)}} \quad (10.7e)$$

where  $\tau = \frac{r}{r_c}$  and  $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ . Here  $\varepsilon$  indicates the strength of the vortex,  $\alpha$  controls the decay rate of the vortex, and  $r_c$  is the critical radius for which the vortex has the maximum strength. In this test case, these values are taken as;  $\varepsilon = 0.3$ ,  $\alpha = 0.204$ , and  $r_c = 0.05$ . The vortex defined here is a steady state solution to the 2D Euler equation [1].

The right state of the flow is found from normal shock relations, which are:

$$\rho_R = \rho_L \frac{(\gamma + 1)M_s^2}{2 + M_s^2(\gamma - 1)} \quad (10.8a)$$

$$u_R = u_L \frac{\rho_L}{\rho_R} \quad (10.8b)$$

$$p_R = p_L \frac{2\gamma M_s^2 - (\gamma - 1)}{\gamma + 1} \quad (10.8c)$$

The grid is chosen as 250x100 and uniform Cartesian for the first case, and non-uniform Cartesian clustered around shock location, for the second case.

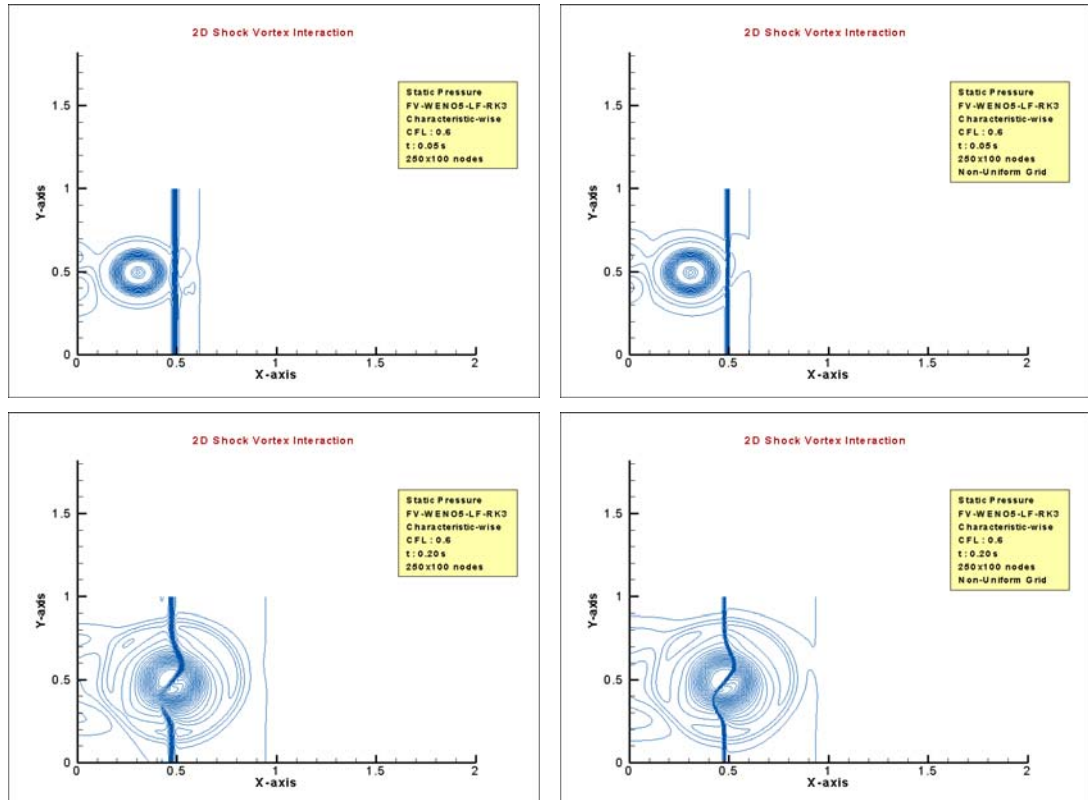


**Figure 10.22(a) : Shock Vortex Int. with Uni. and Non-Uniform Grids (LF-RK3)**

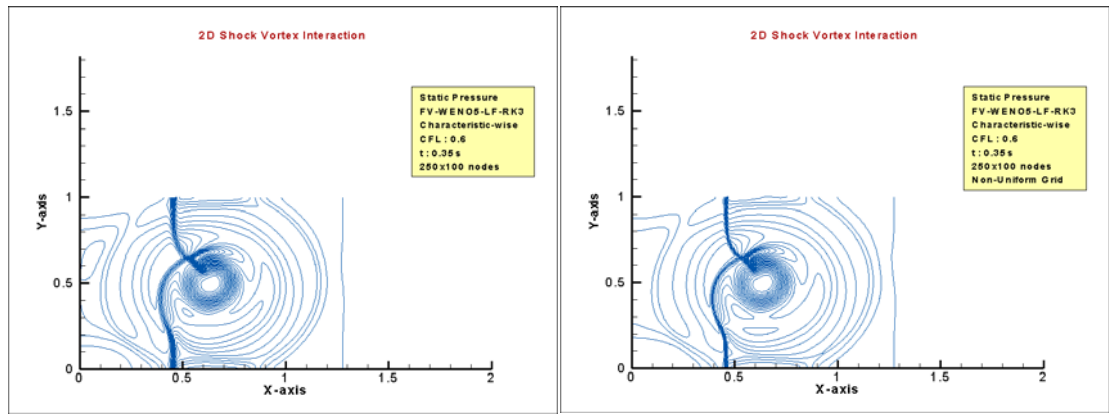


**Figure 10.22(b) : Shock Vortex Int. with Uni. and Non-Uniform Grids (LF-RK3)**

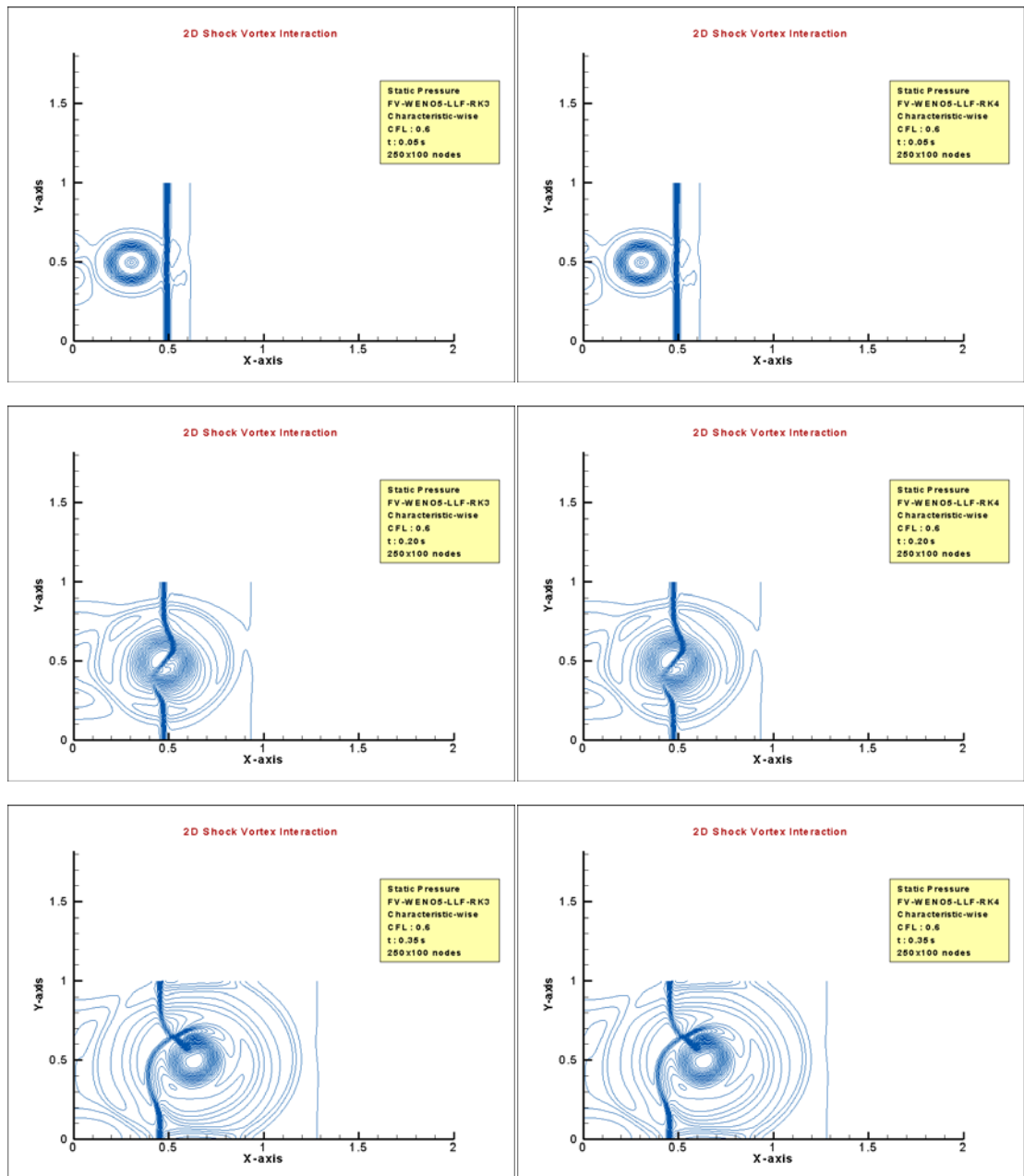


**Figure 10.23 : Shock Vortex Int. with Uniform Grid (LLF-RK3&RK4)**

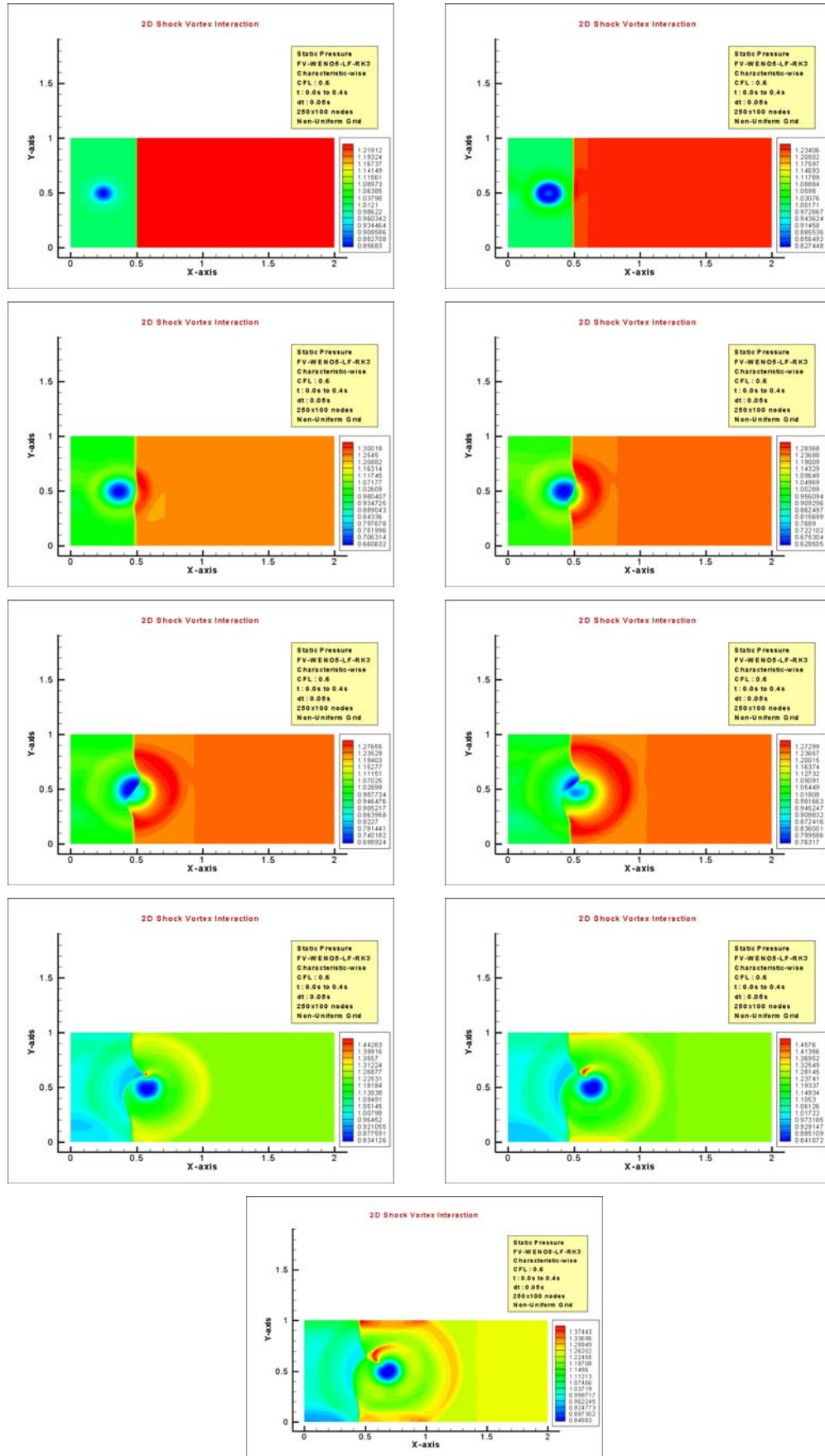


Figure 10.24 : Shock Vortex Int. with Non-Uniform Grid ( $dt = 0.05$  s. LF-RK3)

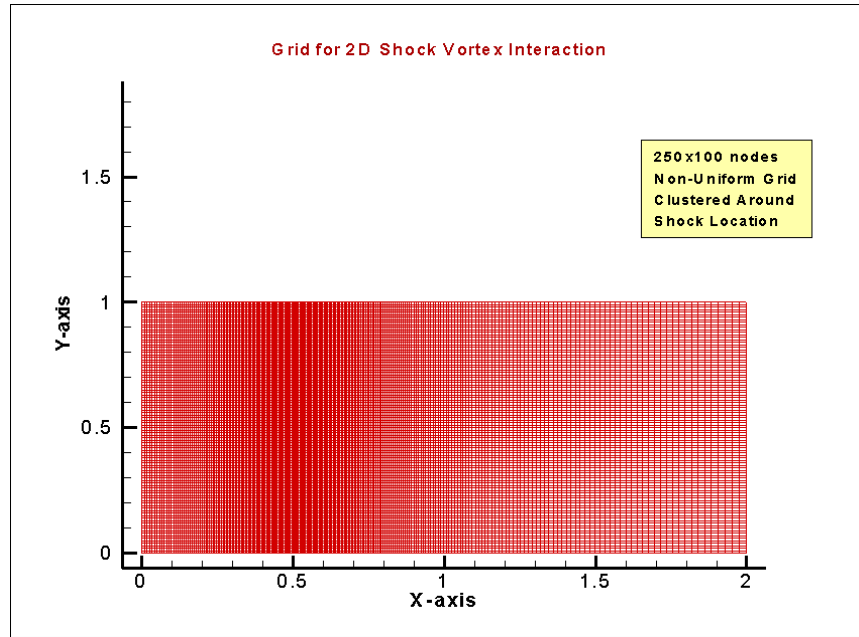


**Figure 10.25 :** Non-Uniform Grid for 2D Shock Vortex Interaction

In the figure 10.22, a uniform and a non-uniform grid (using non-uniform coefficients for solution from section 6), which is clustered around shock location shown in figure 10.25, are used with LF-RK3. The non-uniform grid has a better shock resolution as expected. Both in figures 10.22 and 10.23, static pressure contours are compared for 0.05, 0.20, and 0.35 seconds. Figure 10.23 compares LLF with RK3 and RK4 type time decomposition. It is observed that using LLF or RK4 results in no significant difference for the solution. In figure 10.24, flood contours of static pressure variable are given from 0.0 to 0.4 seconds with a time difference of 0.05 seconds.

### 10.2.3 Double Mach Reflection Problem

This problem is inspired by experimental and numerical studies of planar shocks in air from wedges. The flow can be set up experimentally by driving a shock down a tube which contains a wedge. When wall begins to slope complicated shock reflection occurs. A self-similar flow occurs here which can be parameterized, for a given gamma-law gas, by the Mach number of the incident shock and the angle with which it encounters the reflecting wall of the wedge [30]. In numerical implementation of this problem, despite using a wall which has a slope, tilting the incident shock and using a zero slope rectangular domain is a more feasible way.

The domain is  $x = [0,4]$ ,  $y = [0,1]$ . Reflecting wall boundary lies at the bottom from  $x = \frac{1}{6}$  to the end of the bottom boundary. Initially a right moving Mach 10 shock ( $M_s = 10$ ) is located at  $(x, y) = \left(\frac{1}{6}, 0\right)$  and makes  $60^\circ$  angle with respect to  $x$ -axis. The exact post shock condition is imposed from the beginning of the bottom boundary to  $x = \frac{1}{6}$ . At the top boundary, in the dummy cells, the exact motion of a right-moving 10 Mach shock, which has  $60^\circ$  angle slope, is simulated. The initial right-state variables are

$$\rho_R = 1.4, \quad u_R = 0.0, \quad v_R = 0.0, \quad p_R = 1.0 \quad (10.9)$$

With help of equations (10.8) and (10.10) below [32], the post shock (left-state) variables can be found,

$$V_s = M_s \sqrt{\gamma \frac{p_R}{\rho_R}} \quad (10.10a)$$

$$\rho_R V_s = \rho_L (V_s - V_L) \quad (10.10b)$$

where  $V_s$  is the speed of the shock, and  $V_L$  is the post shock speed of the flow vector, which is making a right angle with the shock. Then, the initial left-state variables are found to be (10.11)

$$\rho_L = 8.0, \quad u_L = 7.1447, \quad v_L = -4.125, \quad p_L = 116.5 \quad (10.11)$$

The initial shock location can be found from the equation (10.12) below

$$x_{s(init)} = \frac{1}{6} + \frac{1}{\tan(60)}(y - y_b) \quad (10.12)$$

where  $x_{s(init)}$  is the initial shock location in  $x$ -coordinate,  $y_b$  is the bottom in  $y$ -coordinate. The updated location of shock is needed for the implementation of the boundary condition at the top dummy cells. The location can be found using (10.13) below

$$x_{s(dum)} = x_{s(init)} + \frac{V_s}{\sin(60)} \Delta t \quad (10.13)$$

where  $x_{s(dum)}$  is the updated shock location in dummy cells.

For the inlet, inflow boundary condition is implemented (subsonic and supersonic boundaries determined automatically by the code), and for the outlet non-reflecting boundary condition is used.

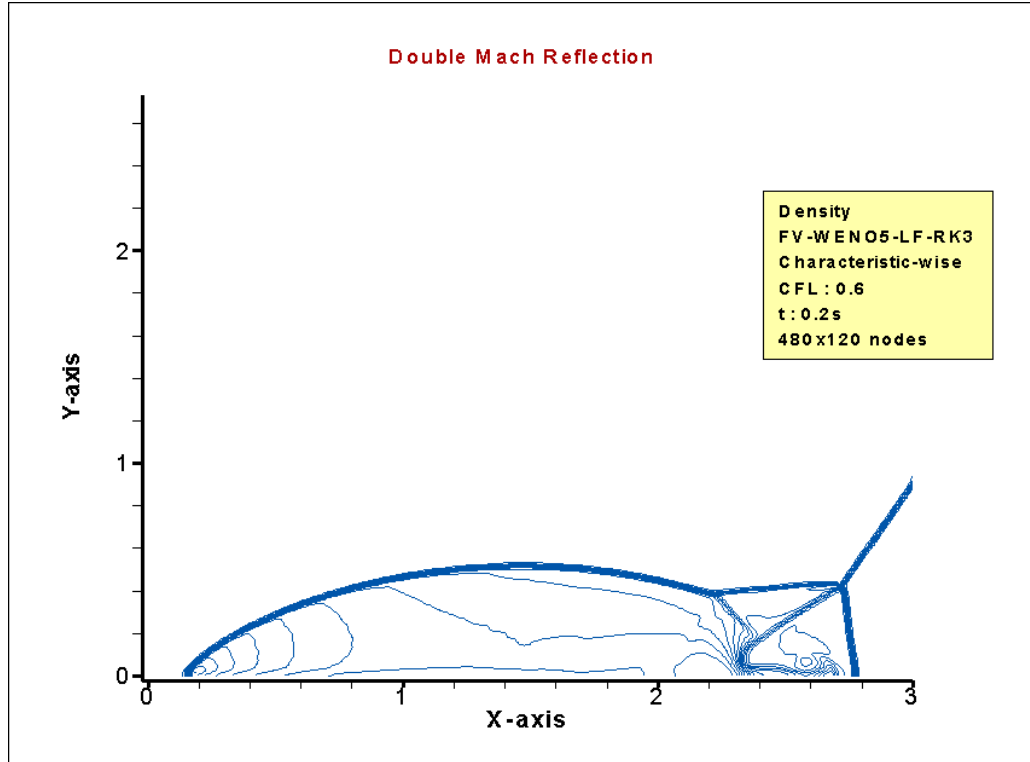


**Figure 10.26 :** Double Mach Reflection 30 Contours from 1.731 to 20.92

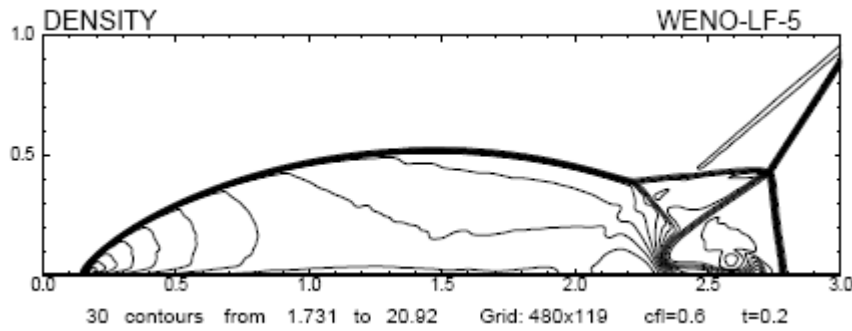


**Figure 10.27 :** Double Mach Reflection, taken from [2]

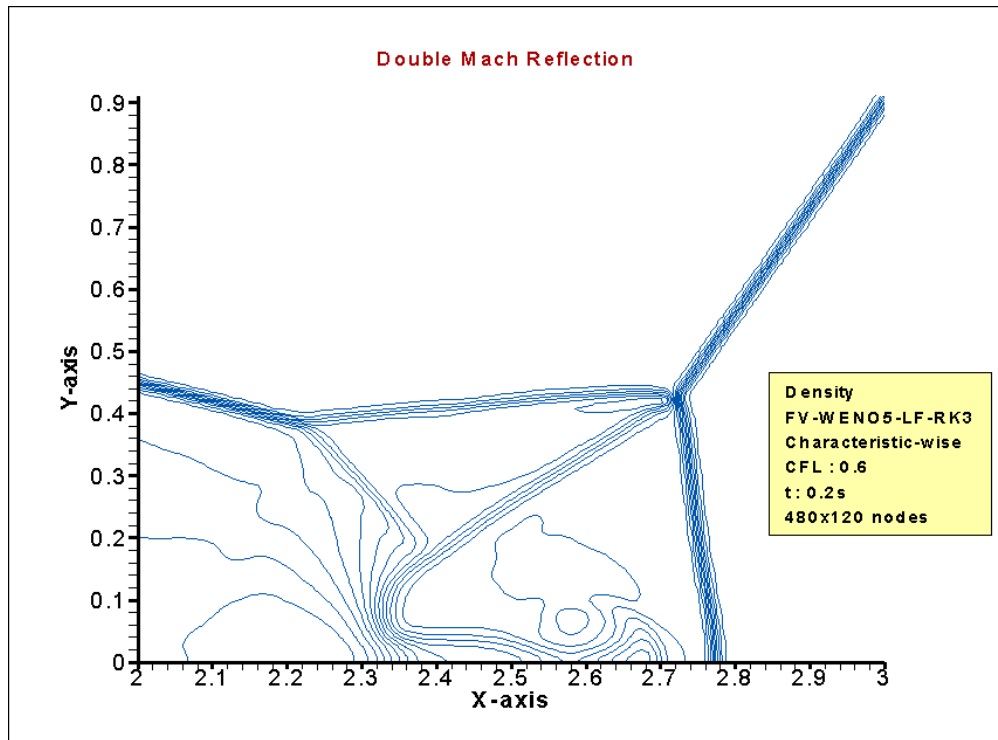


**Figure 10.28 :** Double Mach Reflection (Figure 10.26 zoomed-in)

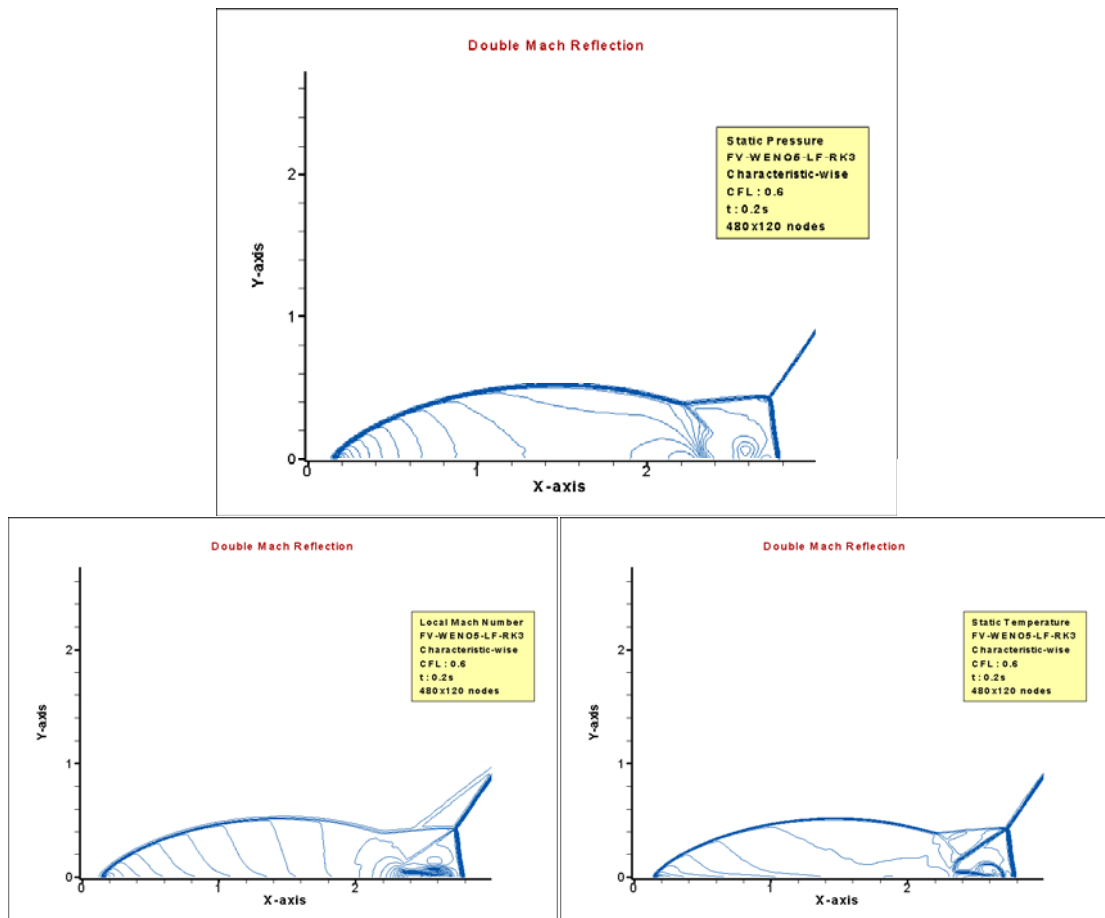


**Figure 10.29(a) :** Double Mach Reflection, Other Variables

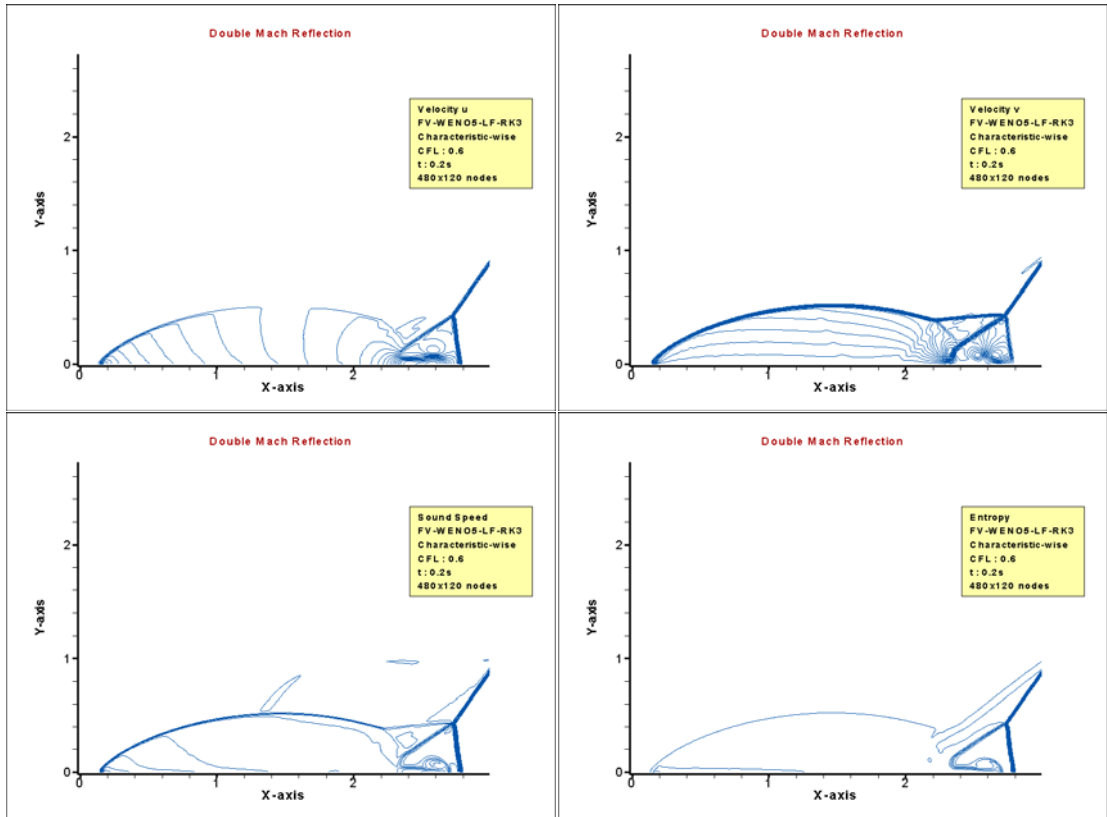


Figure 10.29(b) : Double Mach Reflection, Other Variables

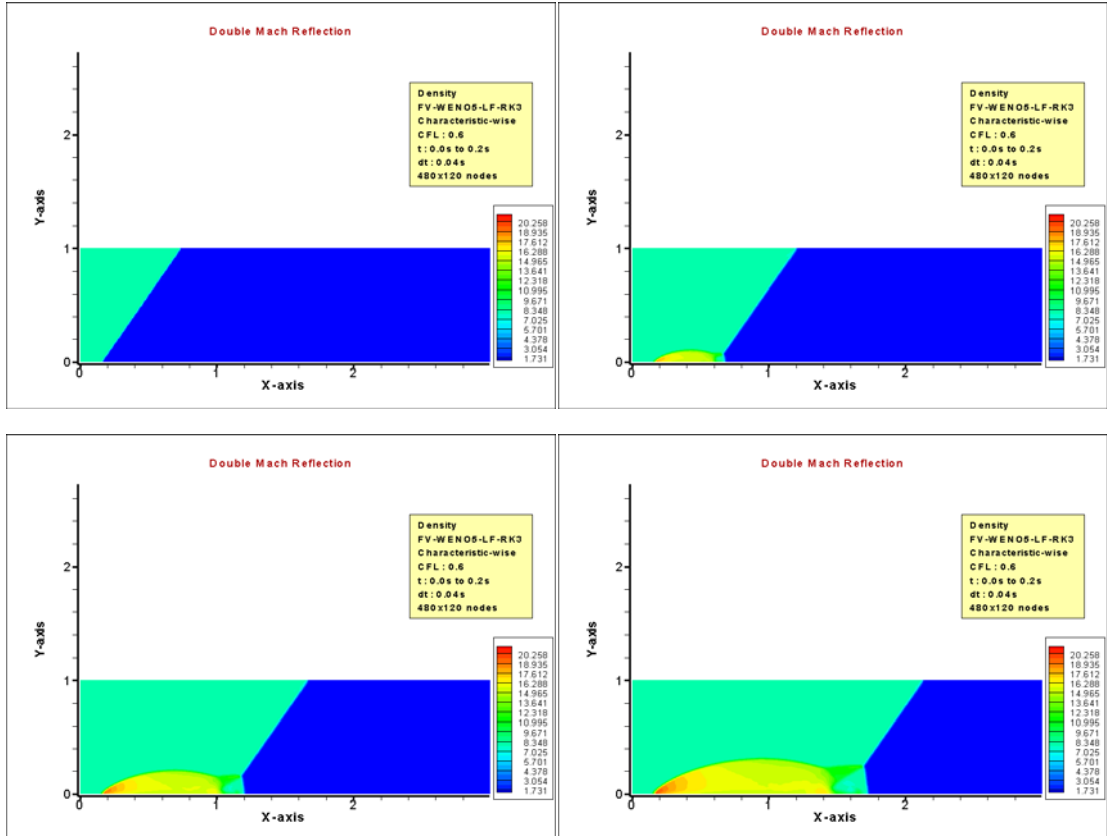


Figure 10.30(a) : Double Mach Reflection ( $dt = 0.04s$ )

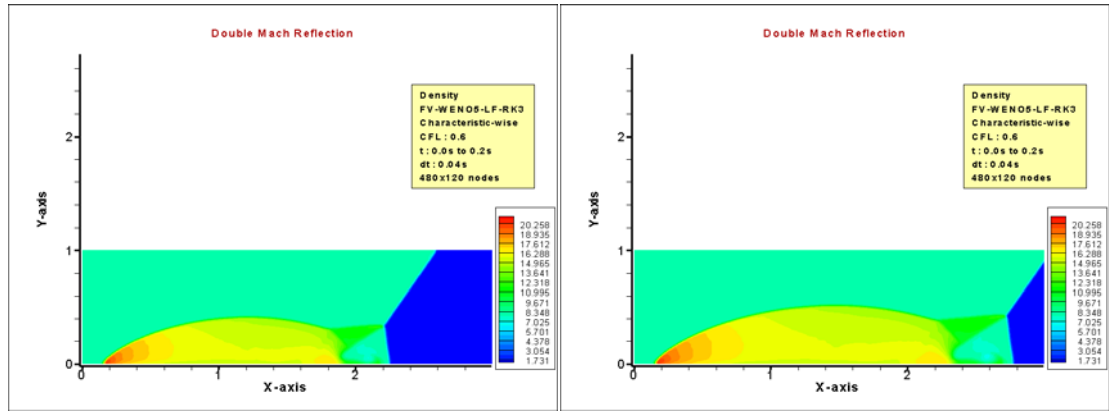


**Figure 10.30(b) :** Double Mach Reflection ( $dt = 0.04s$ )

This test case is implemented using LF and time decomposition type of RK3. Density results, which are given in figure 10.26, show good agreement with the figure 10.27, which is taken from literature. In figure 10.28, the blow-up region near the Mach stem of figure 10.26 is zoomed-in. In figure 10.29, other variables; static pressure, local Mach number, static temperature, u velocity, v velocity, sound speed, and entropy are presented. In figure 10.30, density variable is given from the beginning to 0.2 seconds in flood contour mode. The contours, which are given in line mode, use 30 contour lines. In figure 10.28, although the complicated flow structure can be seen, more mesh refinement is necessary to capture this flow structure more precisely. Figures 10.31 and 10.32 use a finer grid, which has 720x180 cells, complicated flow structure can be seen more clearly.

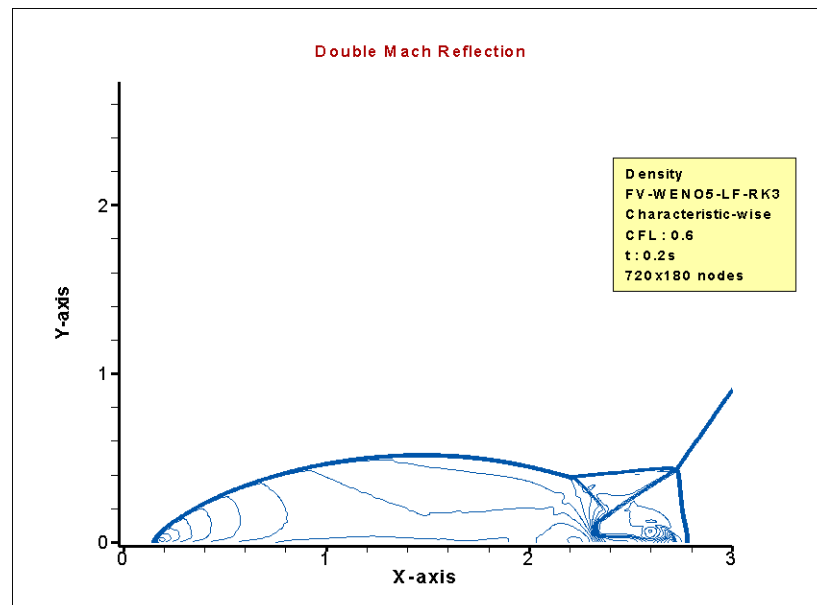


**Figure 10.31 :** Double Mach Ref. 30 Contours from 1.731 to 20.92 (720x180 cells)

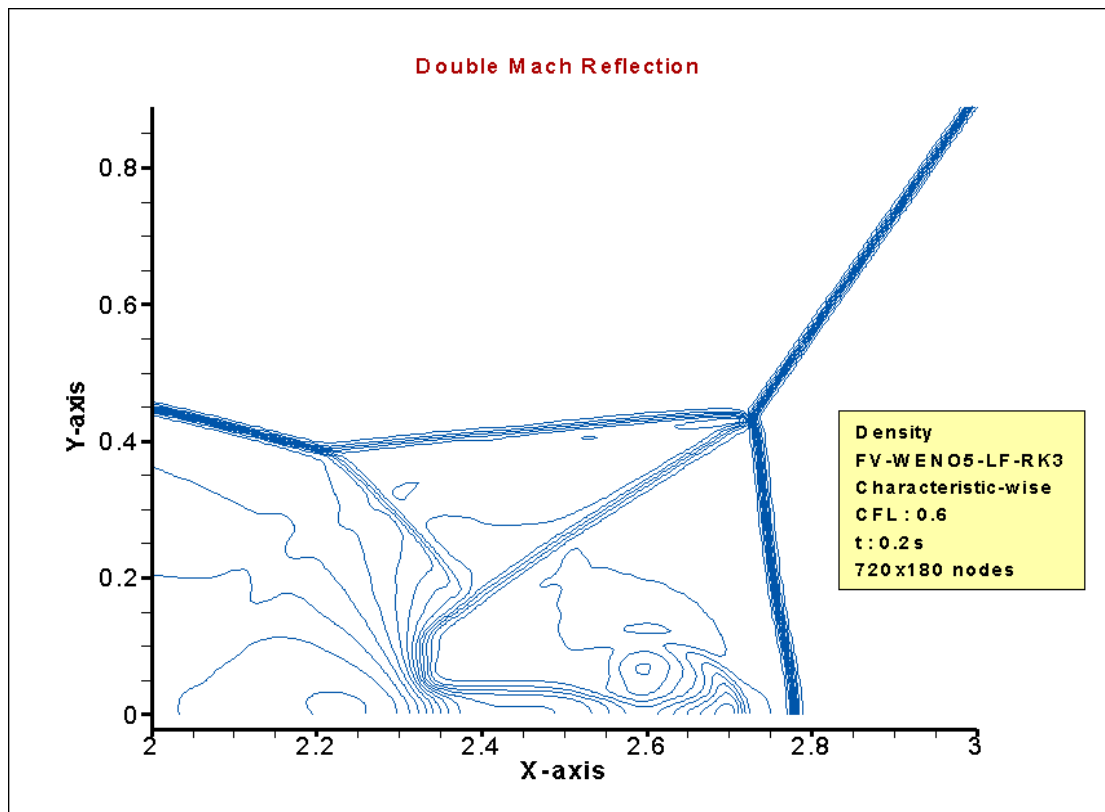


**Figure 10.32 :** Double Mach Reflection (720x180 cells, figure 10.31 zoomed-in)

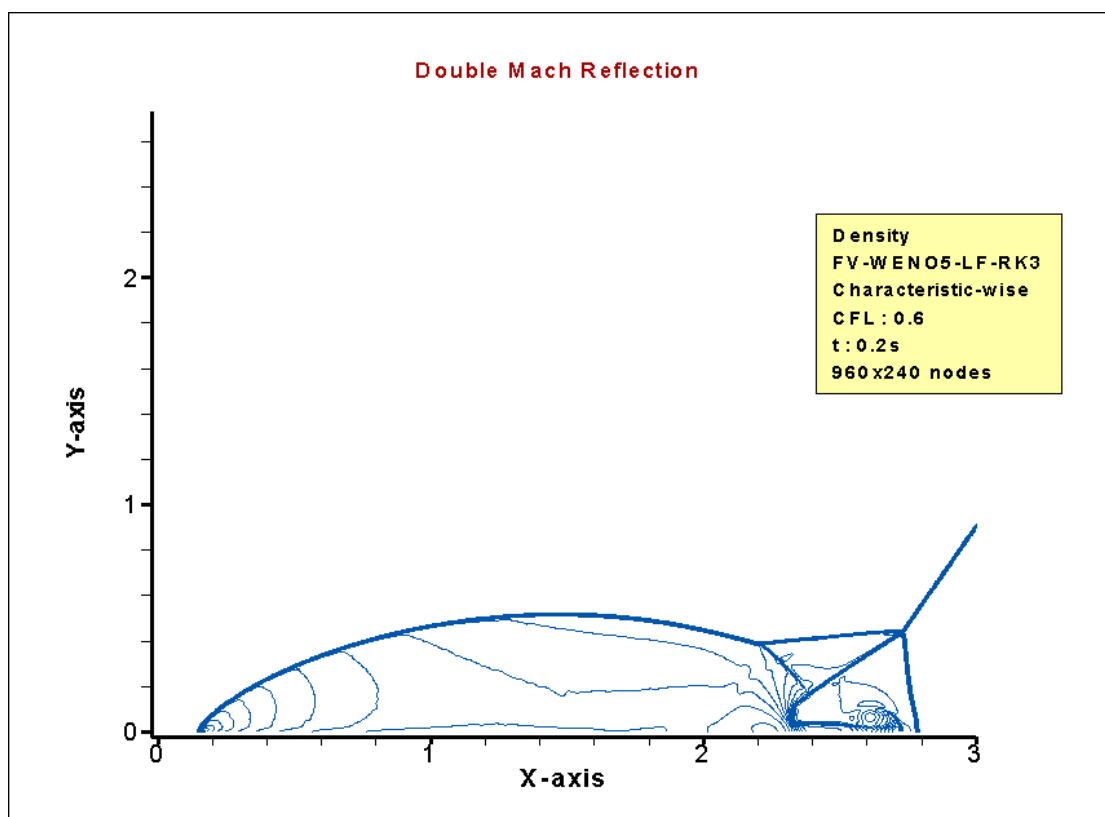


**Figure 10.33 :** Double Mach Ref. 30 Contours from 1.731 to 20.92 (960x240 cells)

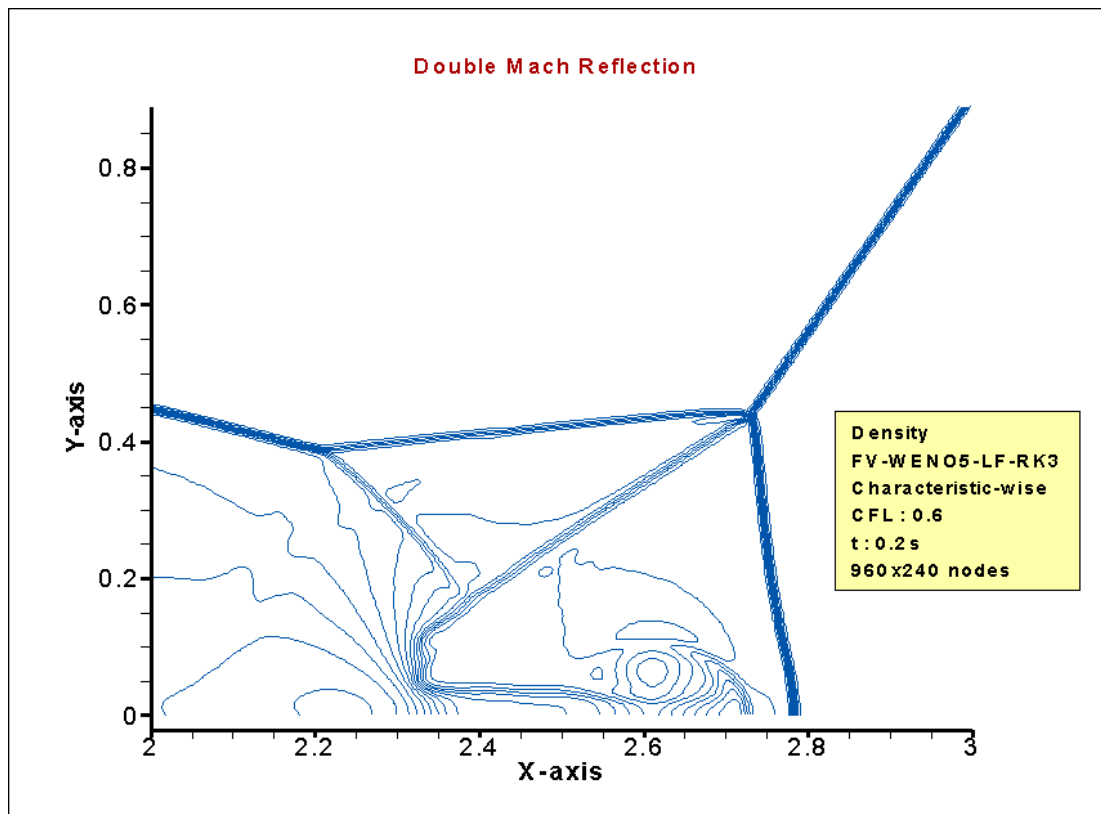


**Figure 10.34 :** Double Mach Reflection (960x240 cells, figure 10.33 zoomed-in)

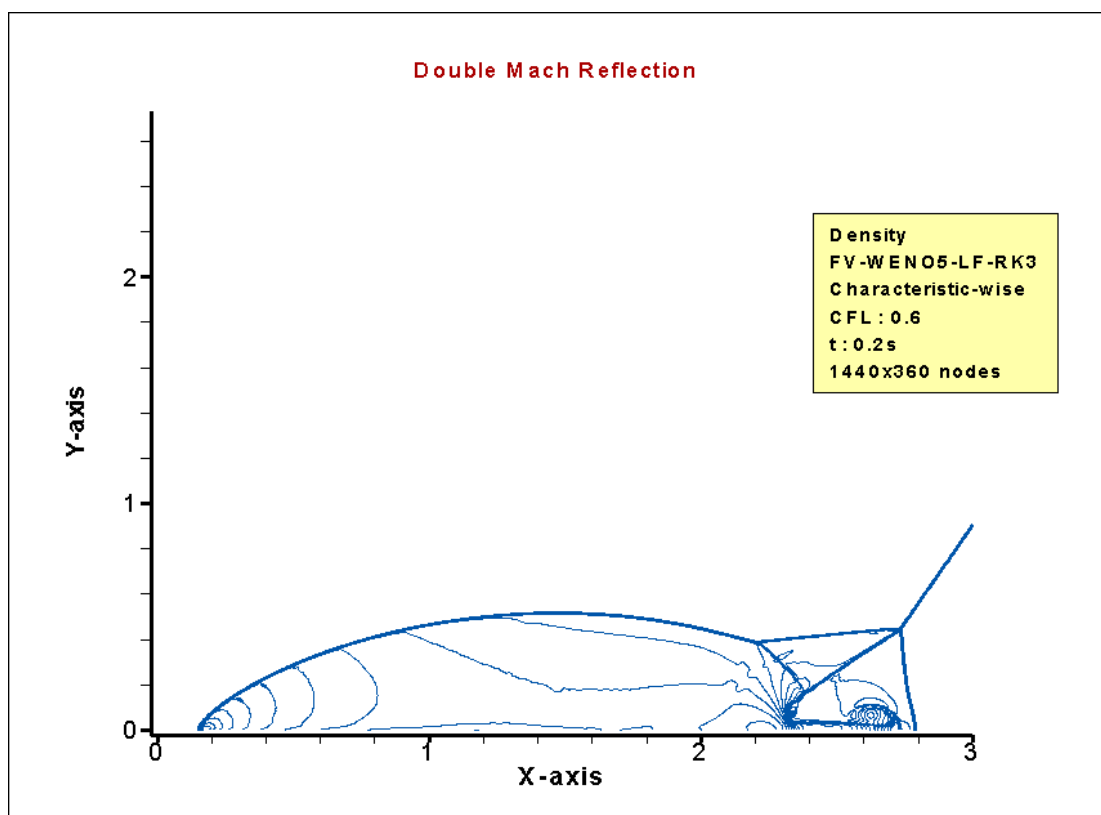


**Figure 10.35 :** Double Mach Ref. 30 Contours from 1.731 to 20.92 (1440x360 cells)

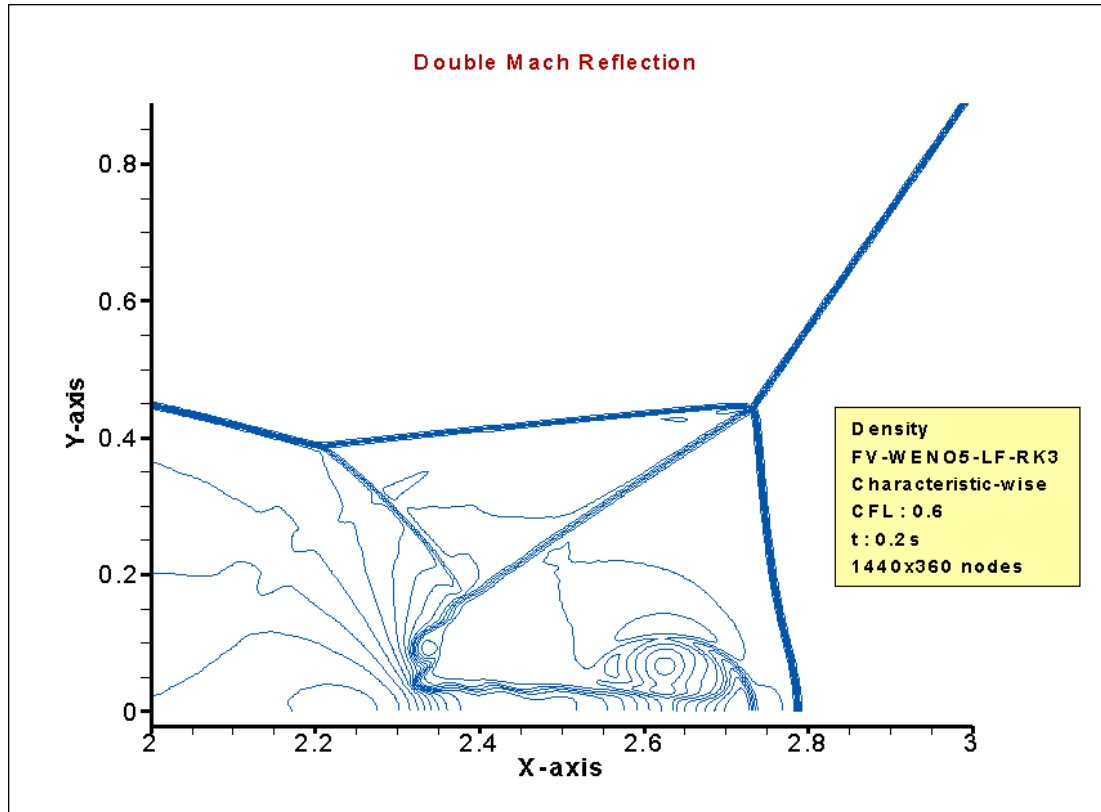


**Figure 10.36 :** Double Mach Reflection (1440x360 cells, figure 10.35 zoomed-in)

Figures 10.31 to 10.36 present results for finer grids. From figure 10.36 above, the complicated flow structure can be seen more clearly.

#### 10.2.4 Forward Facing Step Problem

This problem starts with an initial right-going Mach 3 flow in a wind tunnel containing a step. The wind tunnel is 1 length unit wide and 3 length units long. The step is 0.2 length units high and is located 0.6 length units from the left-hand end of the tunnel. Wall boundary conditions are applied inside the wind tunnel. Inflow and outflow boundary conditions are implemented to inlet and outlet respectively. This problem is solved using two grid blocks, thus a multi block interface, which is discussed in section 7.7, is implemented between two blocks. In addition to these, the corner of the step is the center of a rarefaction fan and hence is a singular point of the flow. This singularity is treated in this implementation similar to the one which is explained in [30], by resetting density variable to its initial value in the first four cells in the first row beginning from the corner, and resetting magnitudes of velocities to their initials not their directions in the first two cells, which are located in the second row just above the step. Resetting values to their initials should not be the right case,

as far as I understand from reference [30], however it works and prevents from the problems, which arise because of this singular point, but introduces additional right moving flow near the wall boundary just after the step.

In order to create a Mach 3 flow, the following initial condition used.

$$\rho = 1.4, \quad u = 3.0, \quad v = 0.0, \quad p = 1.0 \quad (10.14)$$

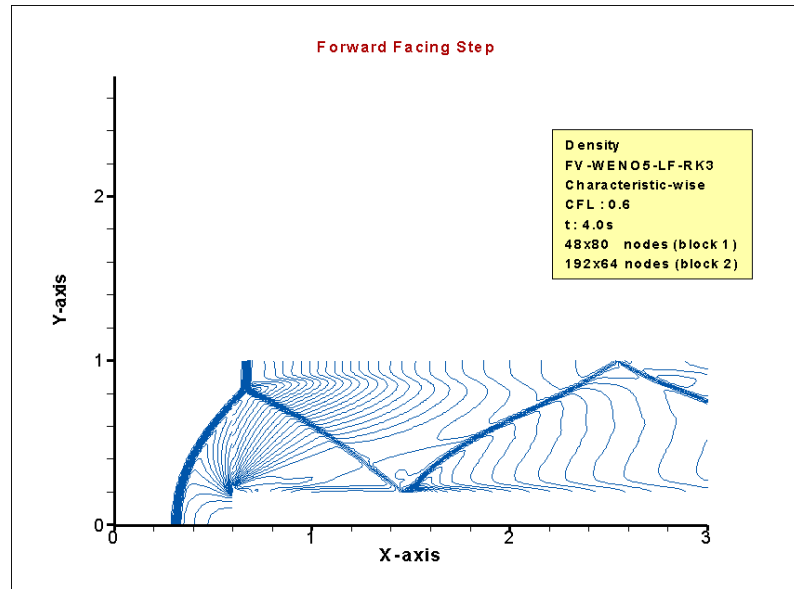


**Figure 10.37 :** Forward Facing Step Line Contours at  $t = 4.0$  s

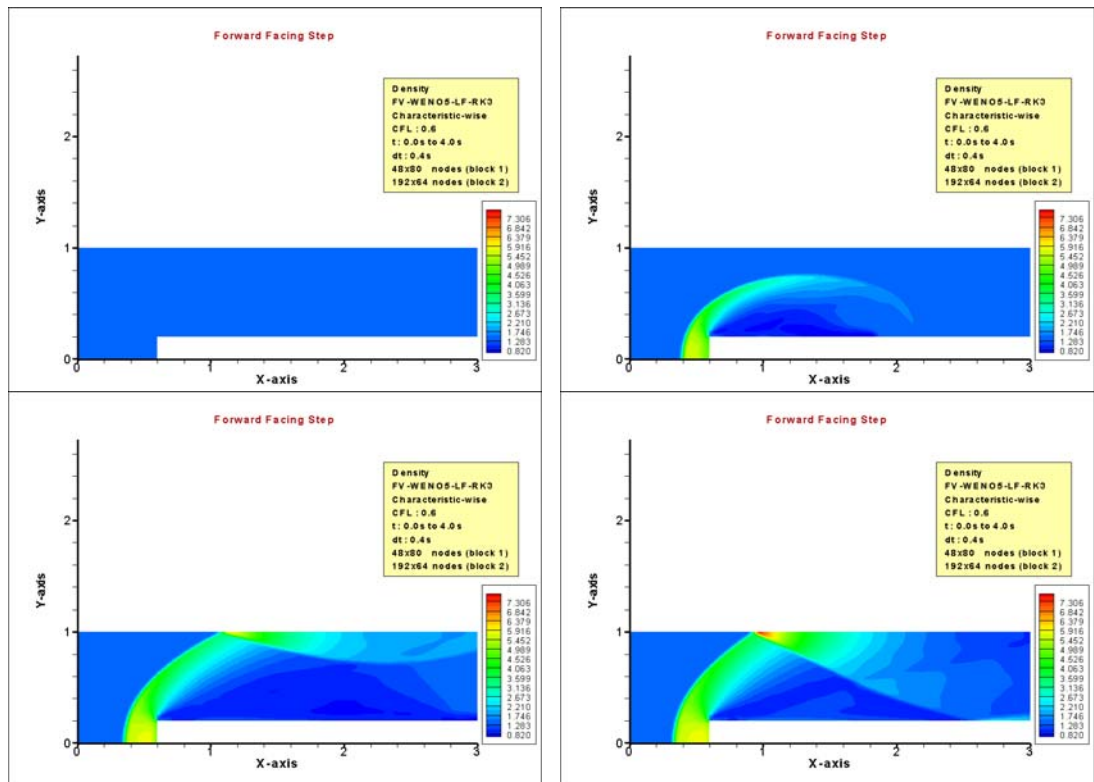


**Figure 10.38(a) :** Forward Facing Step ( $dt = 0.4$  s)

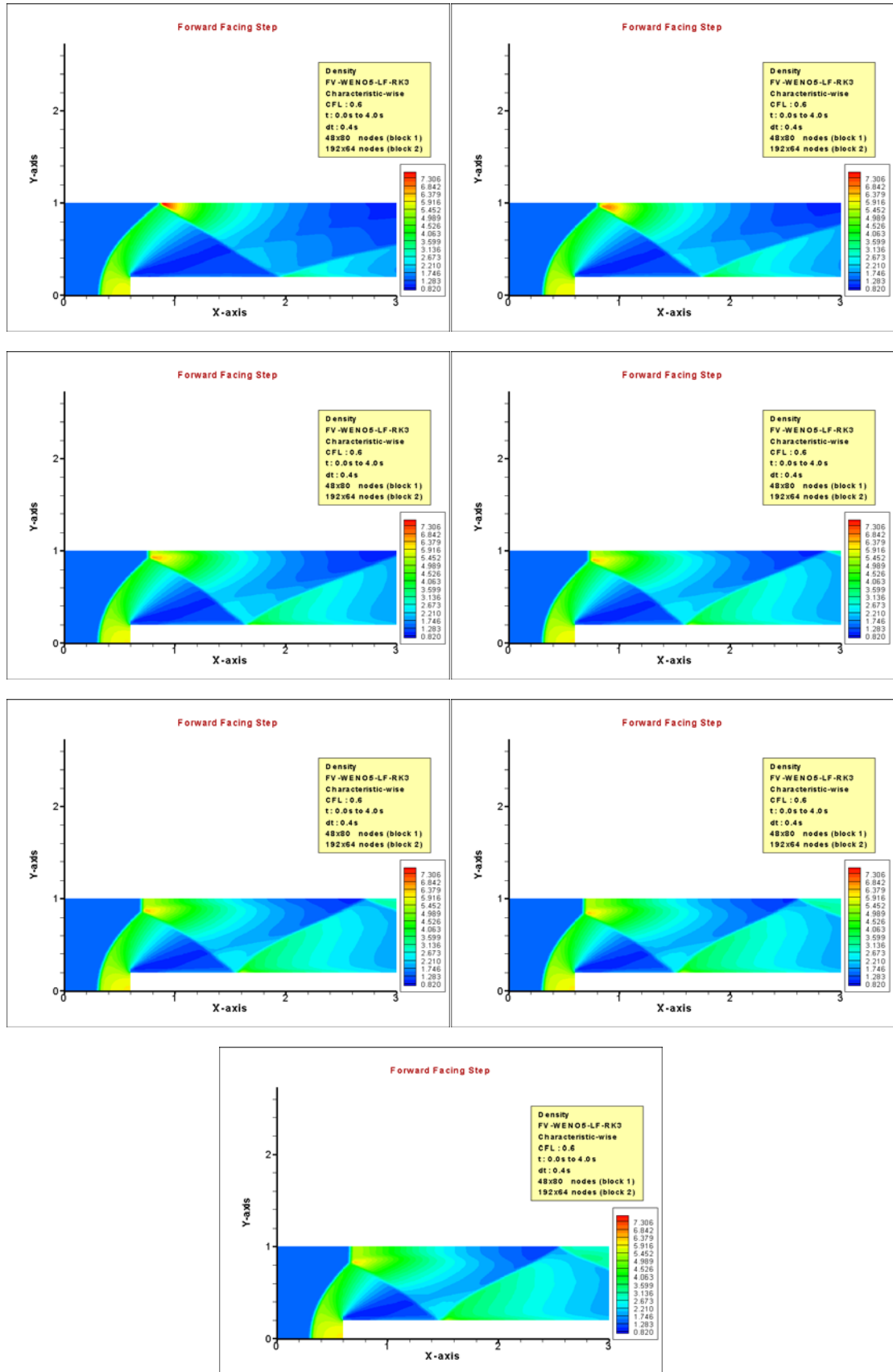


Figure 10.38(b) : Forward Facing Step ( $dt = 0.4s$ )

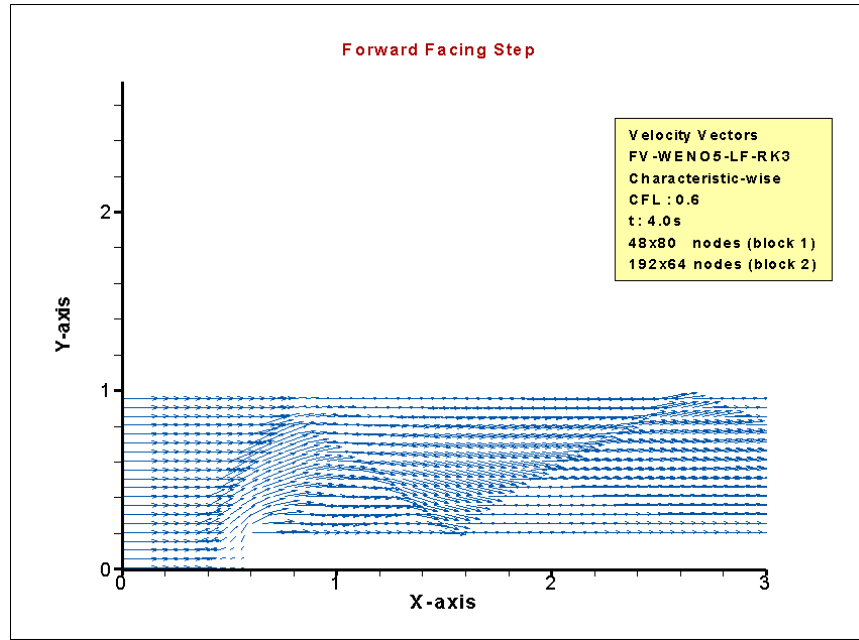


**Figure 10.39 :** Forward Facing Step Velocity Vectors

In figure 10.37, the results of forward facing step test case is given for density variable in 30 contour lines at 4.0 seconds. In figure 10.38 results are given with a time difference of 0.4 seconds. The time evolution; development from beginning, reflection from walls and to the last phase at 4.0 seconds can be seen clearly. In figure 10.39 velocity vectors at 4.0 seconds are presented. In figure 10.40 below, solution at time 4.0 seconds is given, which is taken from literature.

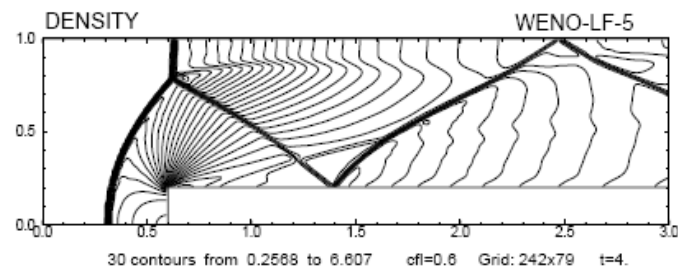


**Figure 10.40 :** Forward Facing Step, taken from [2]

### 10.2.5 Supersonic Flow Past A Cylinder

This test case considers a supersonic flow past a cylinder. The domain contains a portion of the space around the cylinder in the front.

In the physical space, a cylinder of unit radius is positioned at the origin on a  $x-y$  plane. The computational domain is chosen as  $\xi = [0,1]$ ,  $\eta = [0,1]$  on  $\xi-\eta$

plane. The mapping between the computational plane and the physical domain is as the following equation (10.15).

$$x = (R_x - (R_x - 1)\xi)\cos(\theta(2\eta - 1)) \quad (10.15a)$$

$$y = (R_y - (R_y - 1)\xi)\sin(\theta(2\eta - 1)) \quad (10.15b)$$

where the parameters are taken to be  $R_x = 3$ ,  $R_y = 6$ , and  $\theta = \frac{5\pi}{12}$  [1].

The problem is initialized by a Mach 3 flow in the domain (in addition, this is not the same case used in [1,2], they are using a Mach 3 shock moving toward the cylinder). Wall boundary condition is used at the surface of the cylinder ( $\xi = 1$ ), inflow boundary condition is used at ( $\xi = 0$ ), and outflow boundary condition is used at ( $\eta = 0,1$ ).

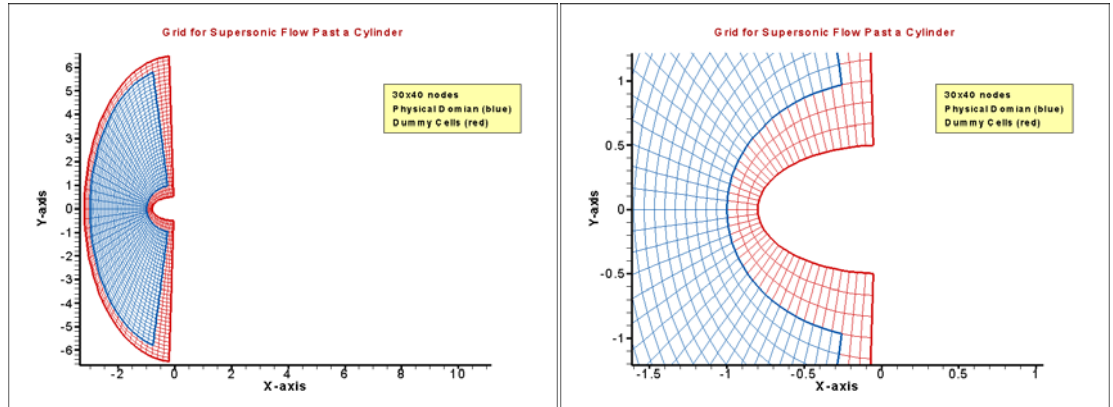


**Figure 10.41 :** Physical Grid for Flow Past a Cylinder (Red:Ghost Cells)

In figure 10.41, the physical grid used for this test case problem is presented. Blue cells denote the physical space, and the extended cells that are given in red color denote the ghost cells. A zoomed-in version around the cylinder boundary is given at the right side of the figure. Because we implemented the boundary conditions in physical space in the code, these ghost cells must be used in the physical space -not in the computational space-. Generation of the ghost cells had already discussed in section 7.1.

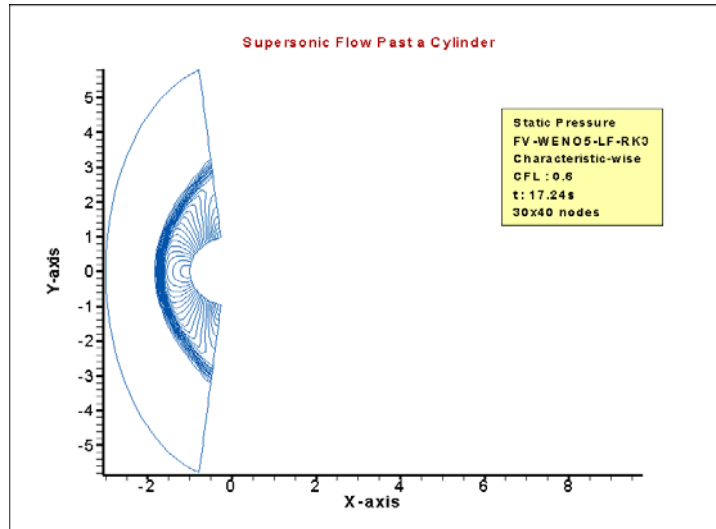


**Figure 10.42 :** Line Contours of Flow Past a Cylinder in Pressure Var. (LF-RK3)

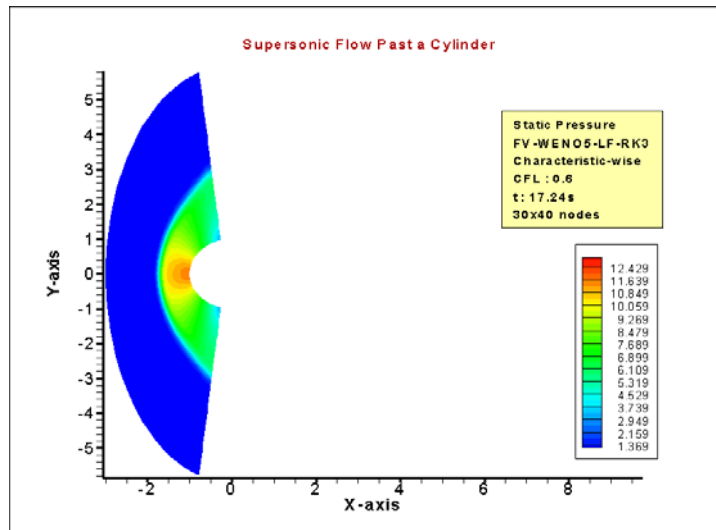


**Figure 10.43 :** Flood Contours of Flow Past a Cylinder in Pressure Var. (LF-RK3)

In figures 10.42 and 10.43 above, the results for static pressure are given in line contour mode (30 lines) and in flood contour mode.

In particular, implementing convergence criteria for an unsteady case is not as easy as a steady state case, however, a convergence criteria is embedded in the code. It uses  $L_2$ -Norms to calculate the density difference between consequent time steps, the discussion about this issue is given in Appendix-B. It is observed that around 17 seconds the difference in density variable drops below  $10^{-3}$ . However, before 17 seconds, like around 10 seconds the flow approximately reaches its final shape. In figure 10.44 below, the time evolution in static pressure is given.

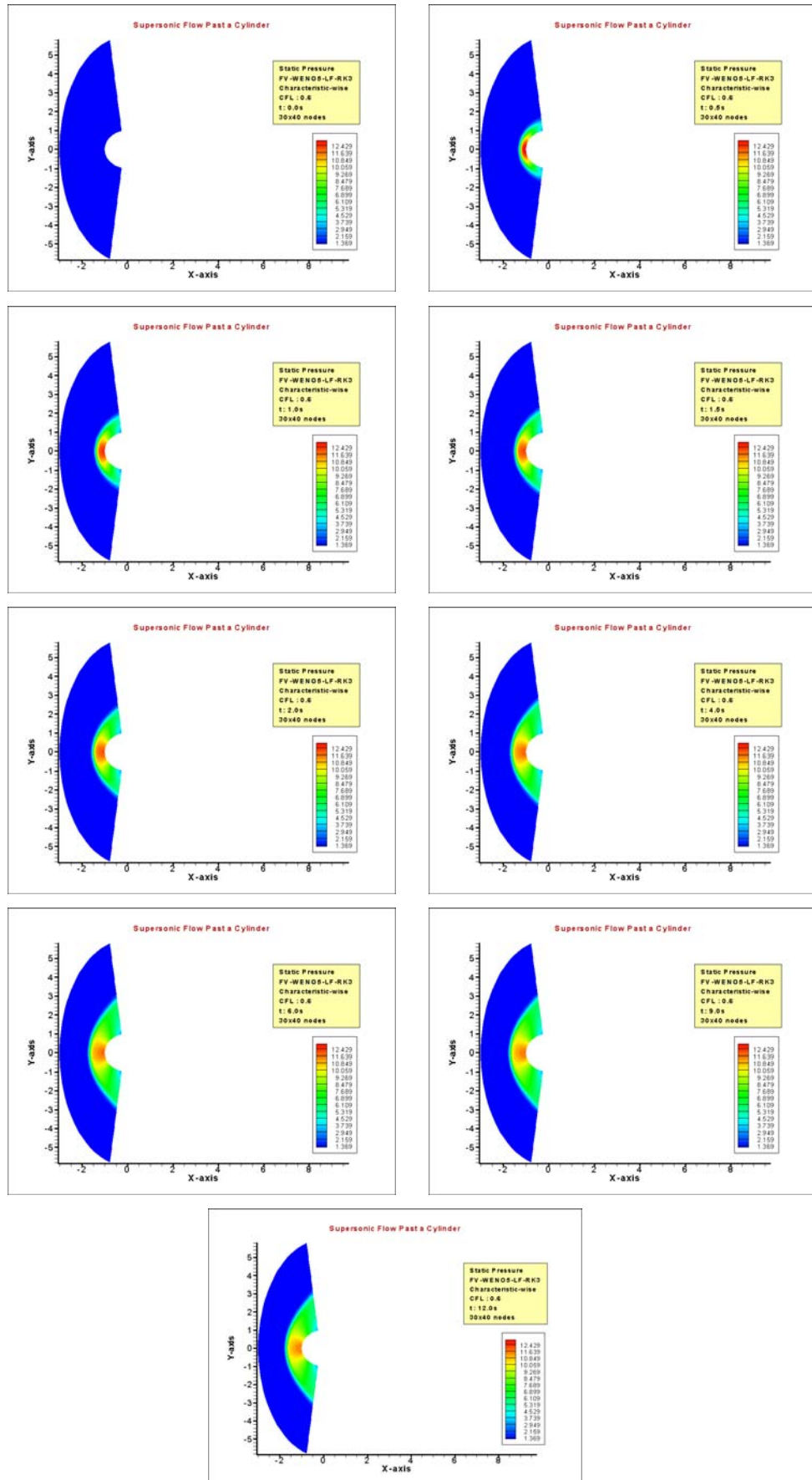


**Figure 10.44 :** Flow Past a Cylinder in Pressure Var.  $t = 0.0s$  to  $12.0s$  (LF-RK3)

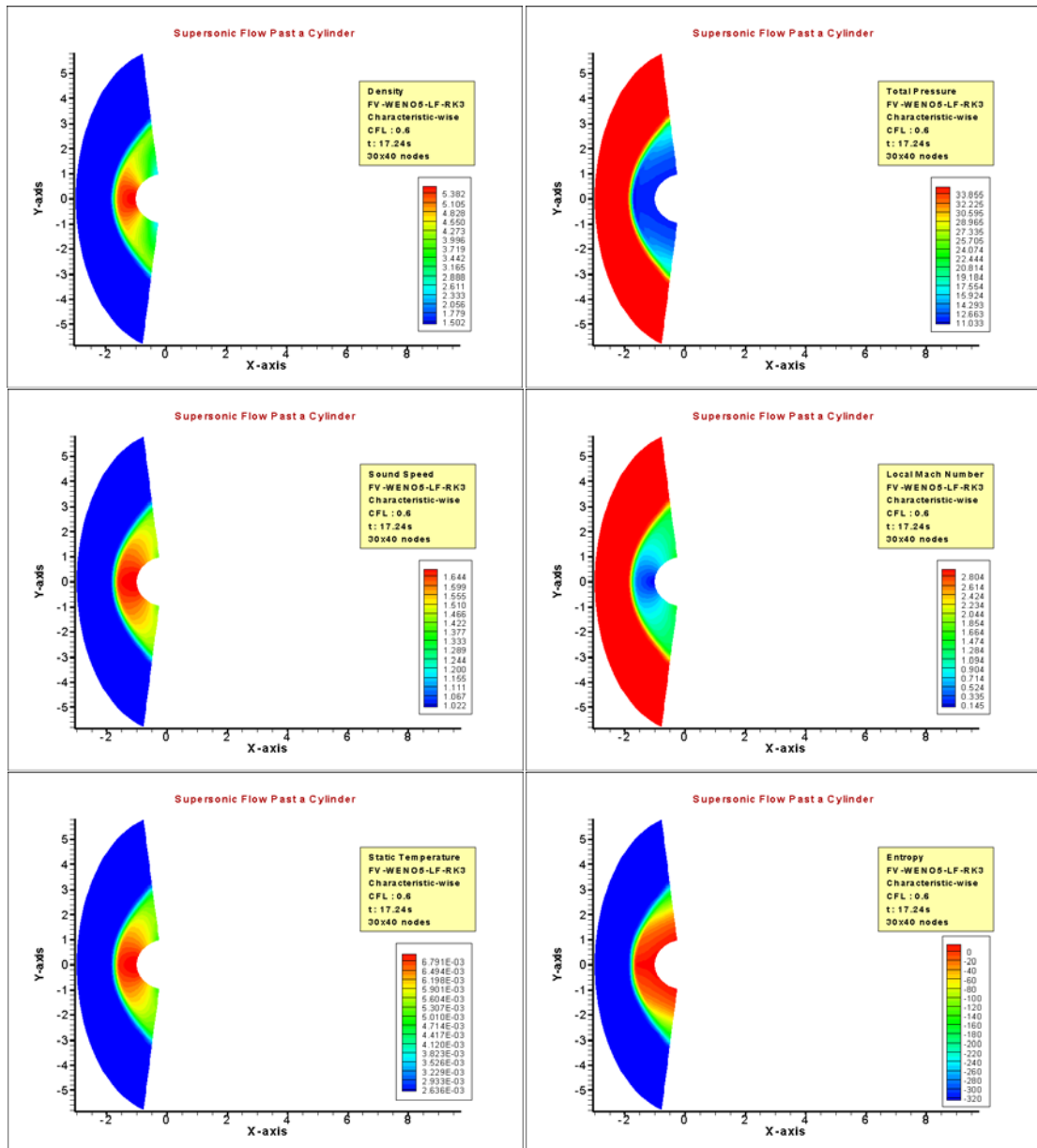


**Figure 10.45 :** Flow Past a Cylinder, Other Variables (LF-RK3)

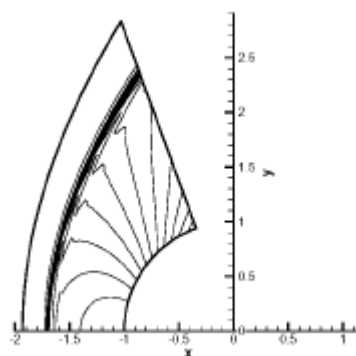


**Figure 10.46 :** Flow Past a Cylinder, taken from [33]

This problem includes a curvilinear grid domain. As already discussed in the previous sections, the structured WENO is constructed for rectangular domains. So, in order to handle curvilinear domains a coordinate transformation has done, which is discussed in section 8.

In figure 10.45, other variables computed for this test case; density, total pressure, sound speed, local Mach number, static temperature, and entropy are given. The pressure contours presented in figure 10.46 is taken from [33], it indicates that the location of the shock and the flow domain is captured well with the coordinate transformation implementation for the code. In [33] they used a fourth-order upwind biased linear finite difference scheme based on Lax-Friedrichs flux splitting, solving Mach 3 flow with Euler equations, and used a uniform 65x65 grid.

### 10.2.6 Flow About NACA0012 Airfoil Cross Section

In this subsection transonic and supersonic flows about NACA0012 cross section will be presented. 180x49 C-Grid, which has a distance of 10 chords to outer boundary is used for the calculations, shown in figure 10.47 and a zoomed-in version in the right of it below. GRIDGEN program is used for grid generation. Results are presented for 0.05 seconds, this is approximately according to 7500 time steps for Case-2 using 180x49 grid.

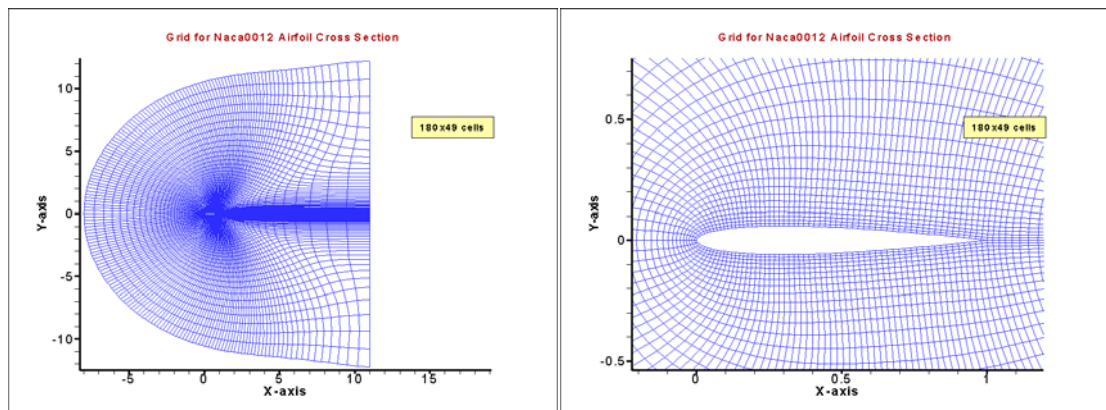


**Figure 10.47 :** Grid for NACA0012 Airfoil Cross Section (180x49)

#### 10.2.6.1 NACA0012 Airfoil Cross Section (Case-1 - $M = 0.8$ , $\alpha = 0^\circ$ )

This is the first test case used with NACA0012 airfoil cross section. Mach 0.8 flow for the domain and zero angle of attack for the airfoil are considered. This flow

type shows transonic flow characteristics when it passes across the airfoil, which is symmetric about the upper and lower part.

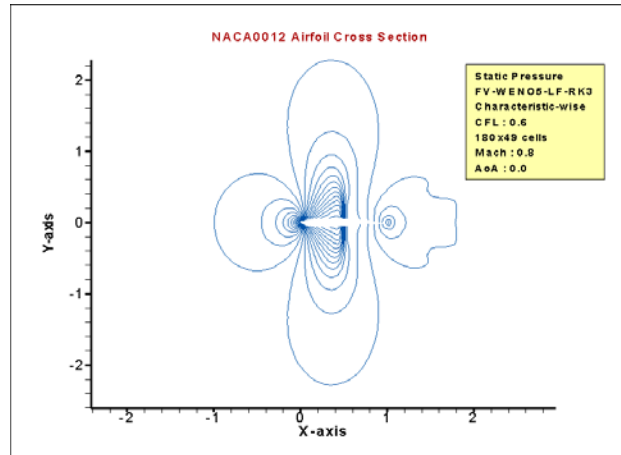


**Figure 10.48 :** Line Contours of NACA0012 (Case-1 Static Pressure)

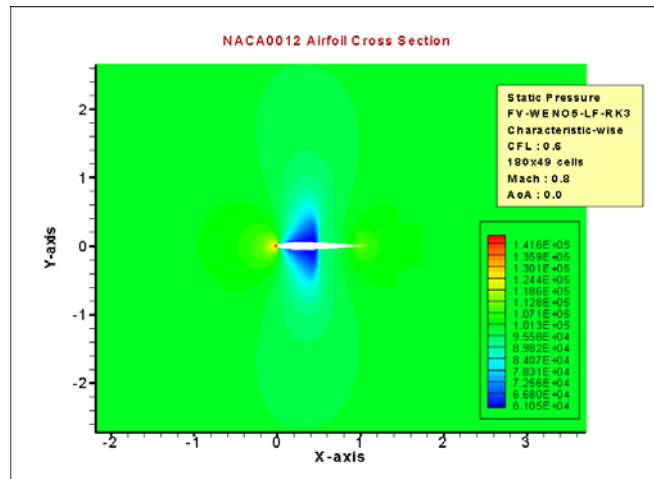


**Figure 10.49 :** Flood Contours of NACA0012 (Case-1 Static Pressure)

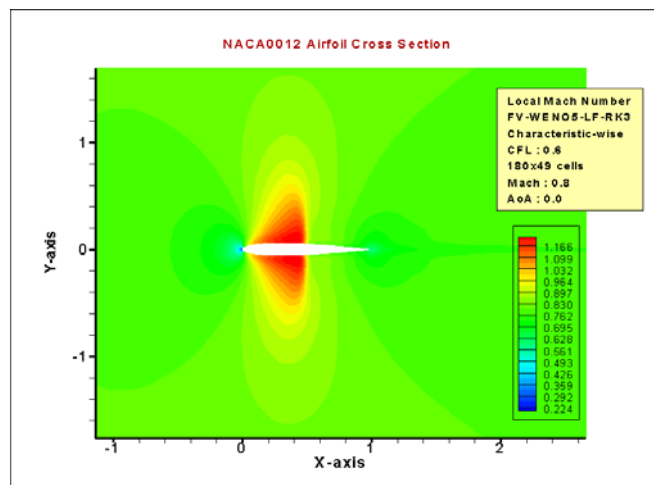


**Figure 10.50 :** Flood Contours of NACA0012 (Case-1 Local Mach Number)

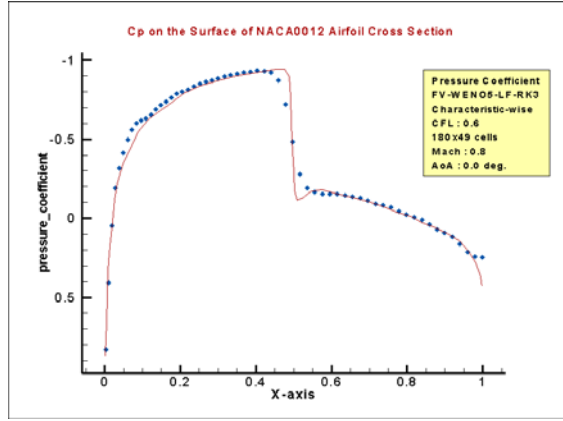


**Figure 10.51 :**  $C_p$  Distribution of NACA0012 (Case-1)

From figure 10.48 to 10.50 pressure and Mach contours are given. It is observed that flow symmetry is well captured. The smoothness of the line curves indicates the strength of this high order scheme. However, at the trailing edge the curve is not a smooth ellipse. This effect is believed to arise from the shape of the grid. WENO is a high order scheme and its dependence on the shape of the grid is much more than a low order scheme, thus, some negative effects arise in the non-smooth grid regions. The  $C_p$  distribution is given in figure 10.51. The distribution is well captured except the trailing edge (grid clustering isn't used for the grid for the trailing edge), and a non-oscillatory shock capturing is achieved. The solid line is from MSES flow solver and taken from [34].

#### 10.2.6.2 NACA0012 Airfoil Cross Section (Case-2 - $M = 0.8$ , $\alpha = 1.25^\circ$ )

This test case uses transonic and non-symmetric flow. Mach number is 0.8 and the airfoil makes 1.25 degrees angle of attack.

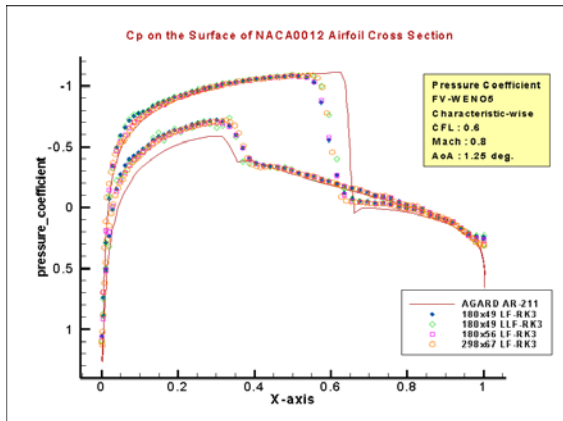


**Figure 10.52 :**  $C_p$  Distribution of NACA0012 (Case-2)

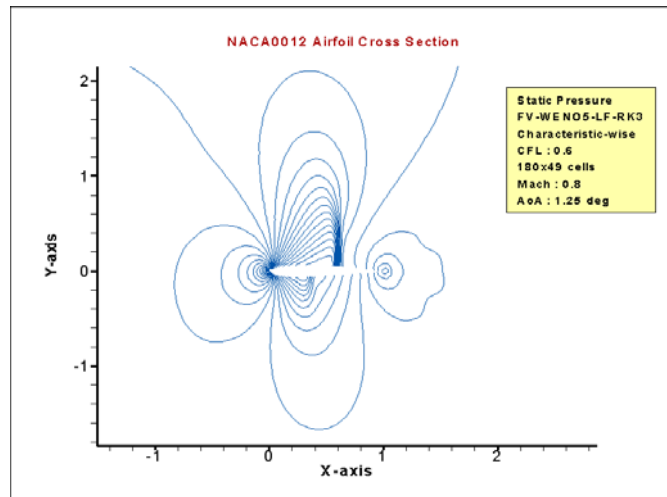


**Figure 10.53 :** Line Contours of NACA0012 (Case-2 Static Pressure)

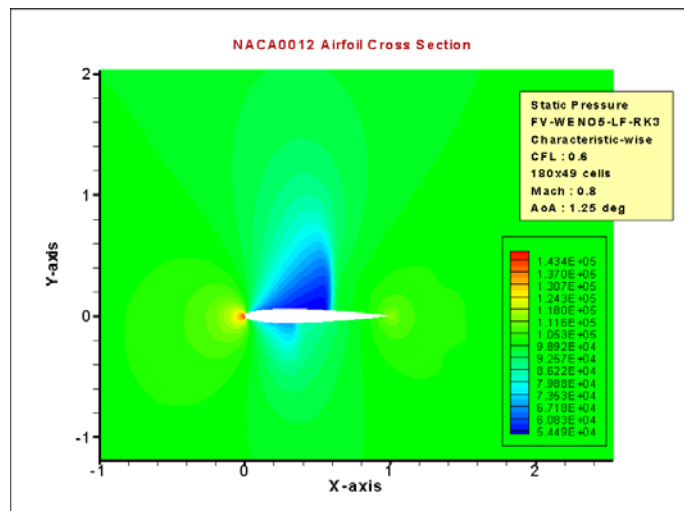


**Figure 10.54 :** Flood Contours of NACA0012 (Case-2 Static Pressure)

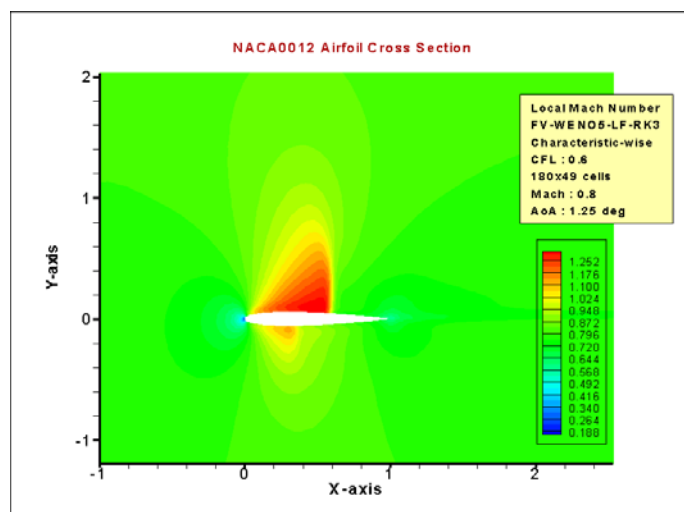


**Figure 10.55 :** Flood Contours of NACA0012 (Case-2 Local Mach Number)

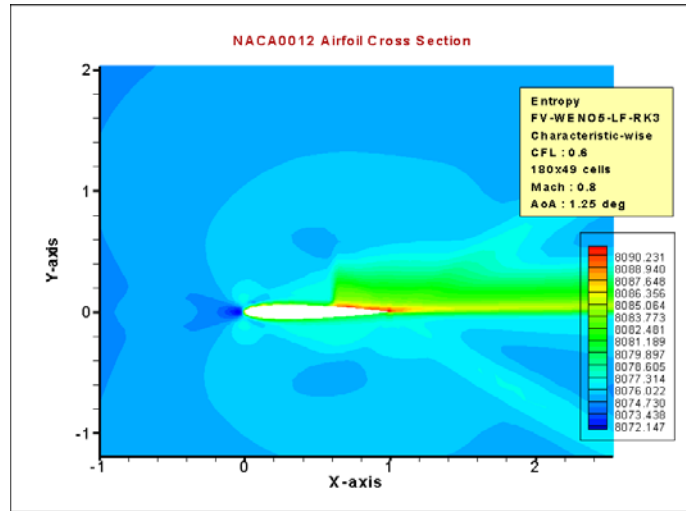


**Figure 10.56 : Flood Contours of NACA0012 (Case-2 Entropy)**

In figure 10.52, pressure coefficient on the surface of the airfoil is given with the comparison of 180x49 LF-RK3, 180x49 LLF-RK3, 180x56 LF-RK3, and 298x67 LF-RK3 cases. The solid line is taken from [35], which is the solution from 320x64 grid and uses 25 chords for the distance to outer boundary. It is observed that the distribution on the lower surface has an offset. As discussed in the previous subsection, these negative effects are believed to arise from the grid (non)-smoothness, in another words, the smoothness that can't achieve the (high) order as the scheme calls for. The smearing of the shock is more than expected. LLF type numerical flux has less smearing than LF, but it has some oscillations on the lower surface before the shock. 180x56 grid eliminates the overshoots between the  $x$ -axis 0.05 to 0.1 on the upper surface. 298x67 grid gives more refined solution. From computational cost and accuracy point of view 180x56 LF is the best of all, although we used 180x49 for the contours. From figures 10.53 to 10.56 static pressure, local Mach number and entropy contours are given.

#### 10.2.6.3 NACA0012 Airfoil Cross Section (Case-3 - $M = 0.85$ , $\alpha = 1.0^\circ$ )

This test case uses transonic and non-symmetric flow. Mach number is 0.85 and the airfoil makes 1.0 degrees angle of attack.

The pressure coefficient distribution on the surface of airfoil is given in figure 10.57 below. The offset of values is again observed on the lower surface. 298x67 grid domain gives an improvement compared to 180x49, but this is not a considerable amount when the additional computational cost taken into account.

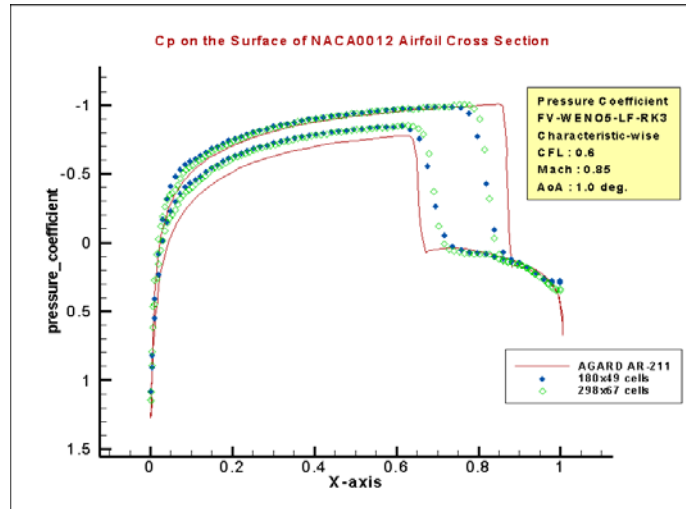


**Figure 10.57 :**  $C_p$  Distribution of NACA0012 (Case-3)

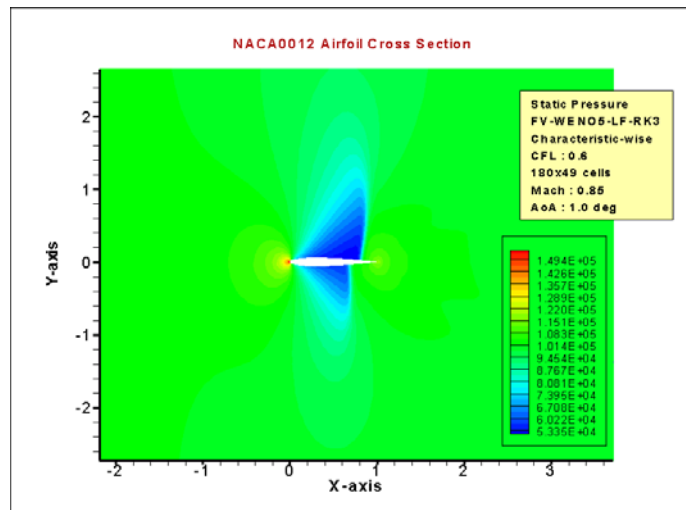


**Figure 10.58 :** Flood Contours of NACA0012 (Case-3 Static Pressure)

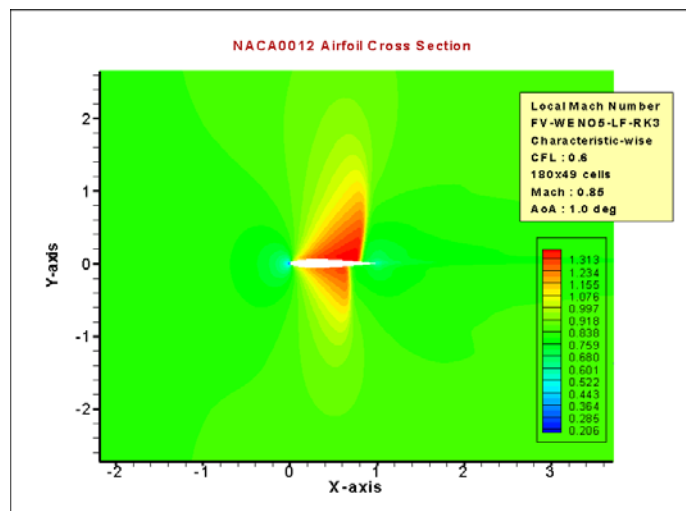
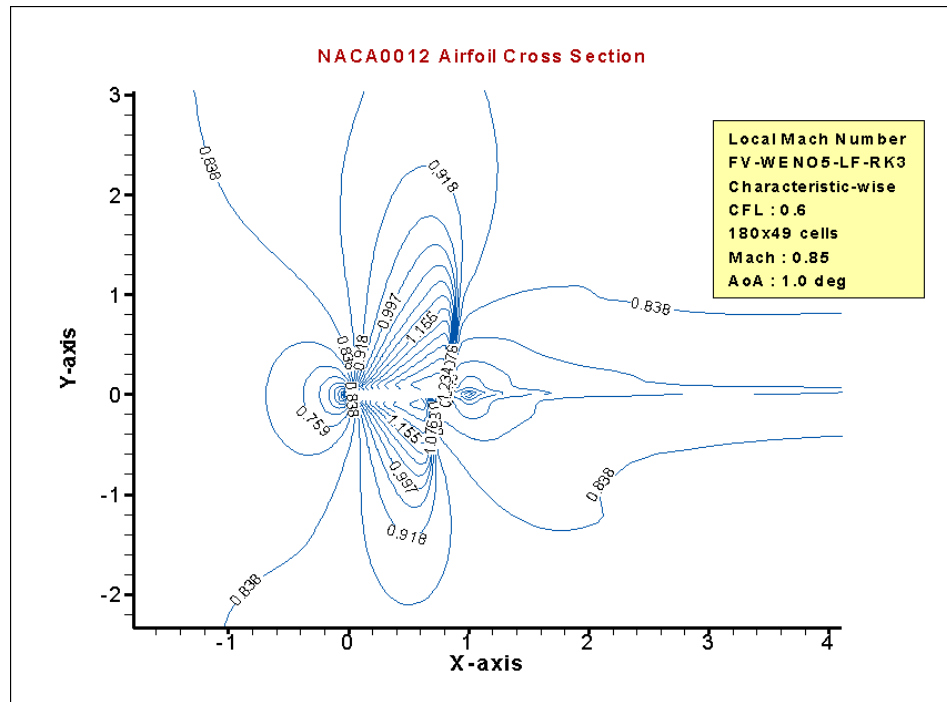


**Figure 10.59 :** Flood Contours of NACA0012 (Case-3 Local Mach Number)



#### 10.2.6.4 NACA0012 Airfoil Cross Section (Case-4 - $M = 0.95$ , $\alpha = 0^\circ$ )

This test case uses transonic and symmetric flow. Mach number for the flow is 0.95 and the airfoil makes zero degrees angle of attack.

The pressure coefficient distribution on the surface of airfoil is given in figure 10.62 below. The results show good agreement with the AGARD case. 298x67 grid domain gives more accurate results compared to 180x49 grid domain, but again this not a considerable amount when the computational cost is taken into account.

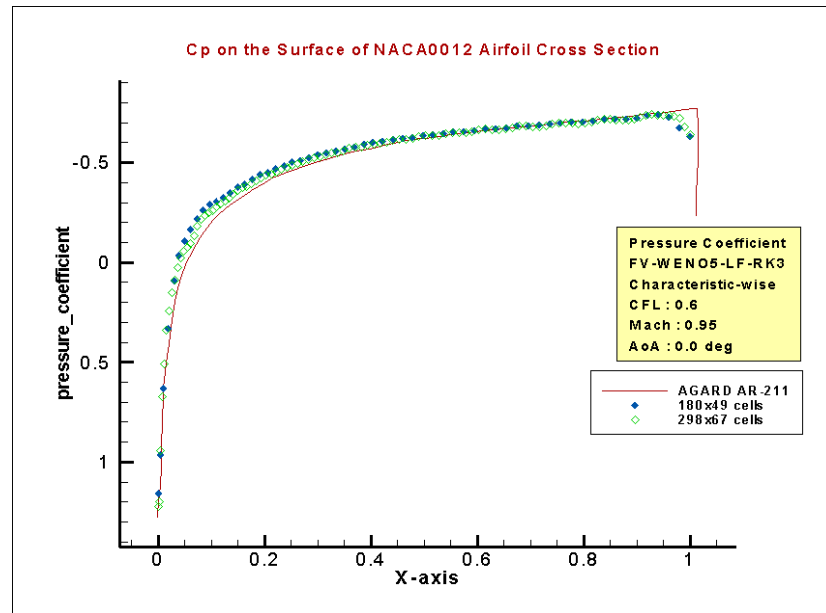


Figure 10.62 :  $C_p$  Distribution of NACA0012 (Case-4)

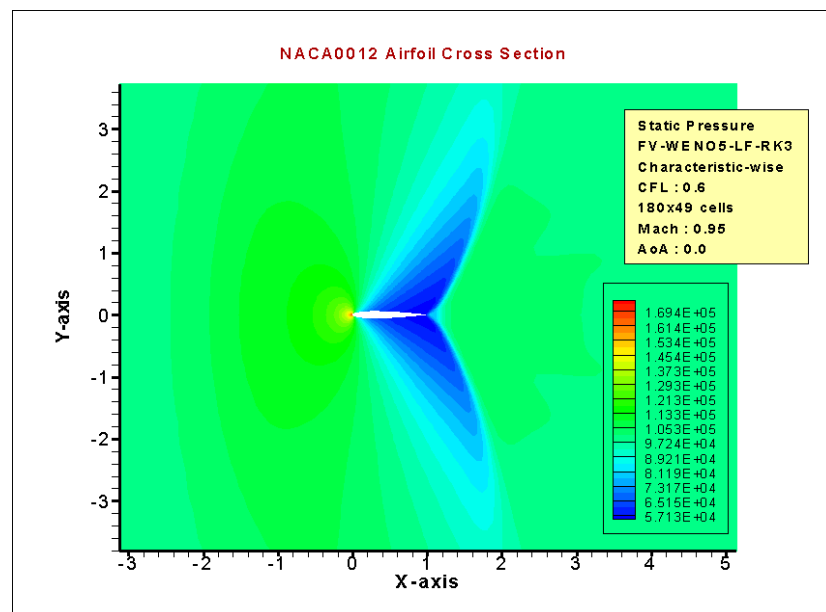


Figure 10.63 : Flood Contours of NACA0012 (Case-4 Static Pressure)

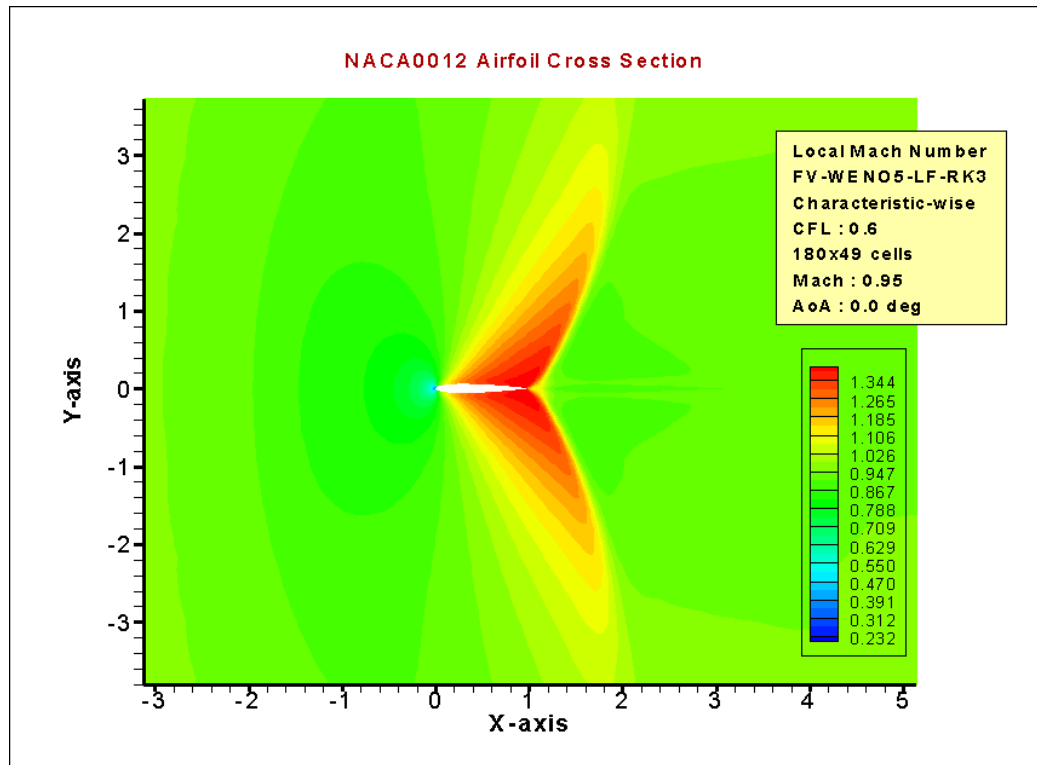


**Figure 10.64 : Flood Contours of NACA0012 (Case-4 Local Mach Number)**

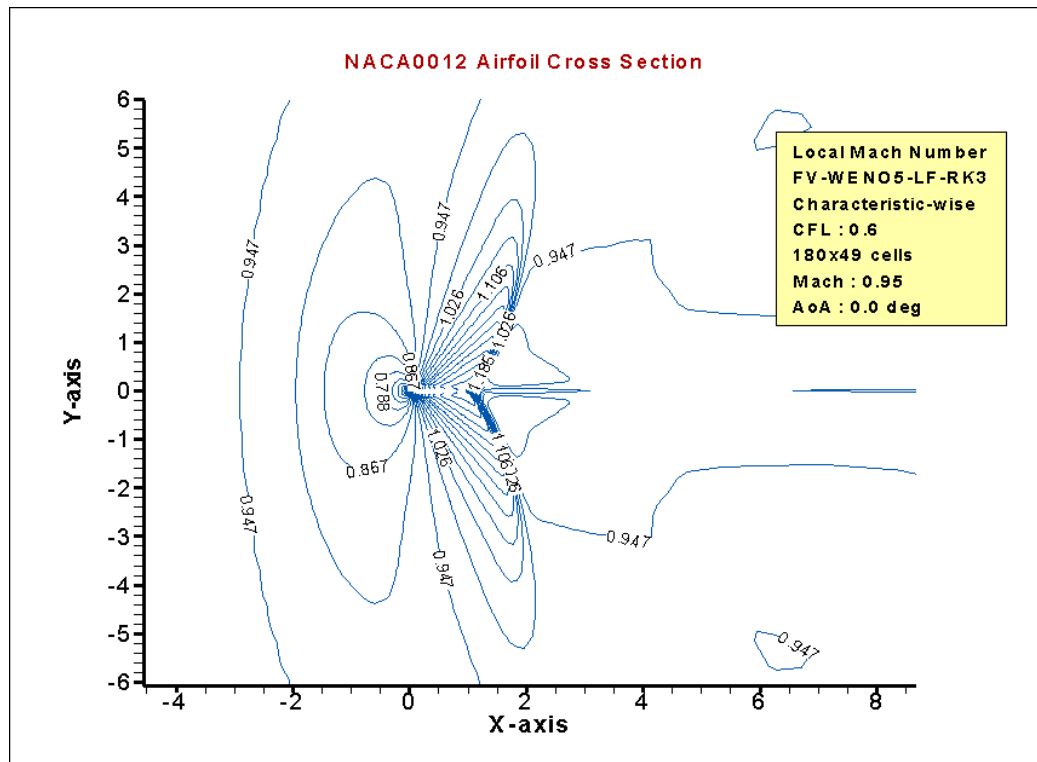


**Figure 10.65 : Line Contours of NACA0012 (Case-4 Local Mach Number)**

From figures 10.63 to 10.65 static pressure, local Mach number contours are given for this test case.

#### 10.2.6.5 NACA0012 Airfoil Cross Section (Case-5 - $M = 1.2$ , $\alpha = 0^\circ$ )

This test case uses supersonic and symmetric flow. Mach number for the flow is 1.2 and the airfoil makes zero degrees angle of attack.

The pressure coefficient distribution on the surface of airfoil is given in figure 10.66 below. Results comparison is identical to Case-4.

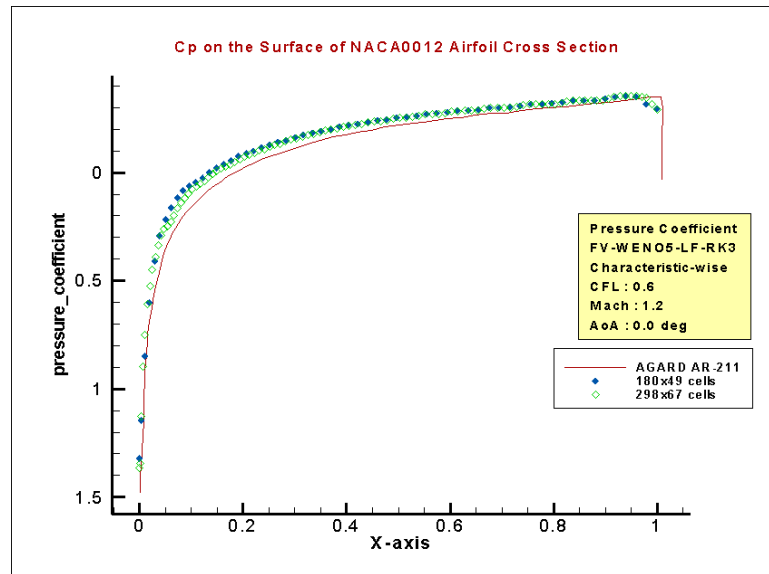


Figure 10.66 :  $C_p$  Distribution of NACA0012 (Case-5)

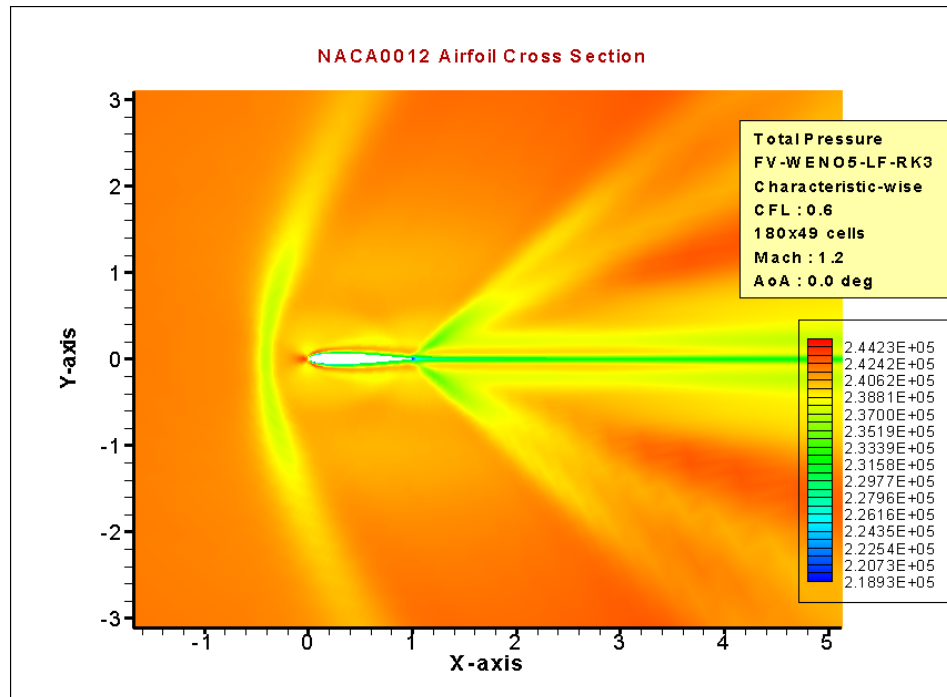


Figure 10.67 : Continuous Flood Contours of NACA0012 (Case-5 Total Pressure)

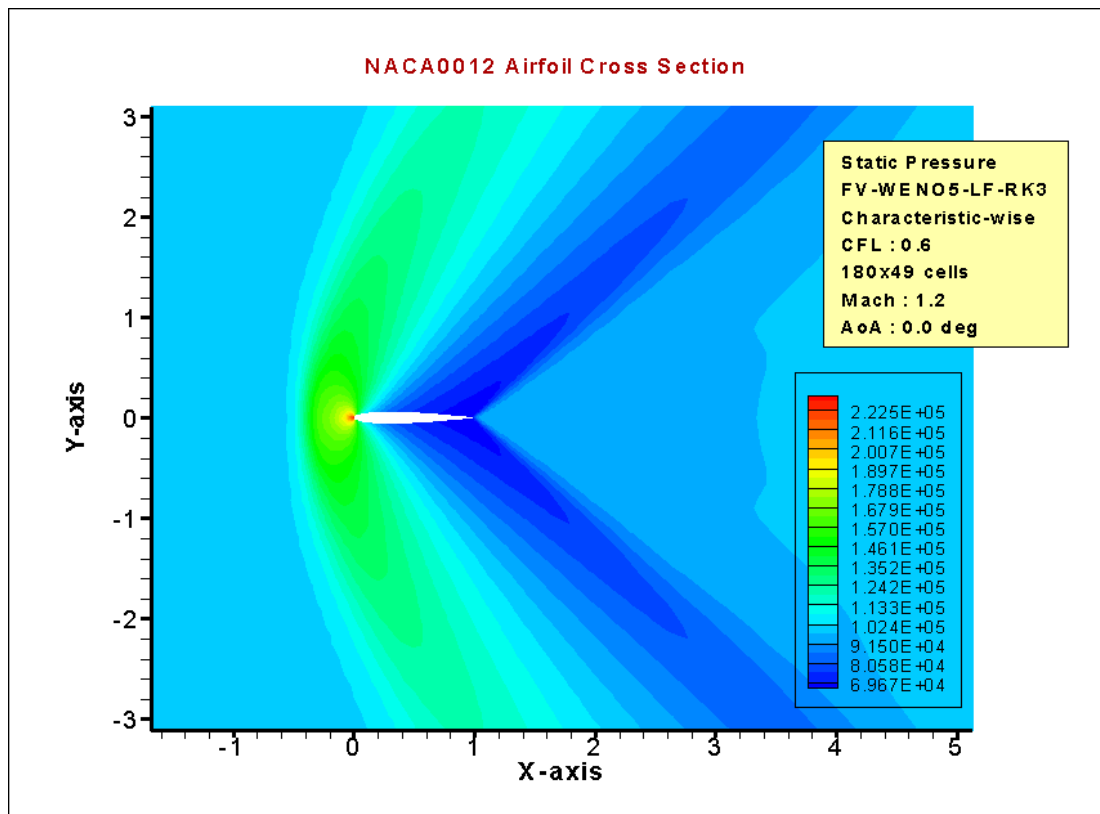


**Figure 10.68** : Flood Contours of NACA0012 (Case-5 Static Pressure)

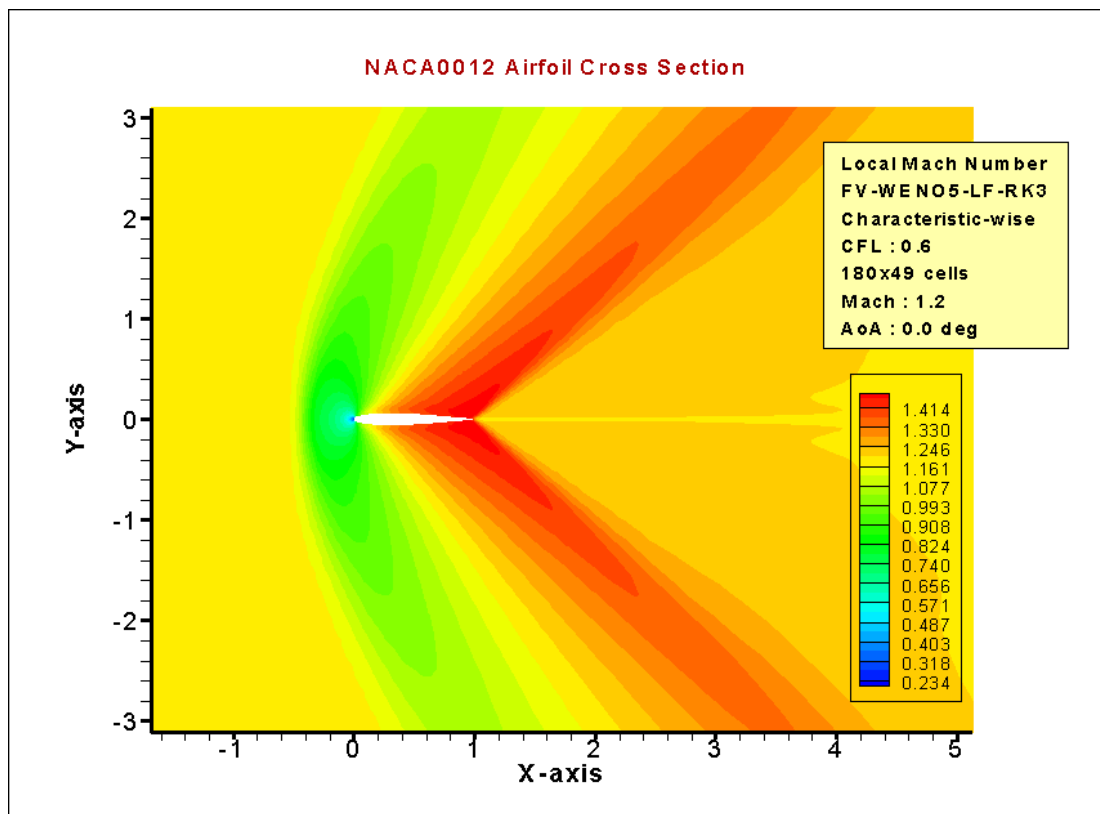


**Figure 10.69** : Flood Contours of NACA0012 (Case-5 Local Mach Number)

From figures 10.67 to 10.69; total pressure, static pressure, and local Mach number contours are given. In the figures 10.70 and 10.71 below, local Mach number iso-contours are presented. Figure 10.71 is taken from [35]. Good agreement between Mach contours is observed.

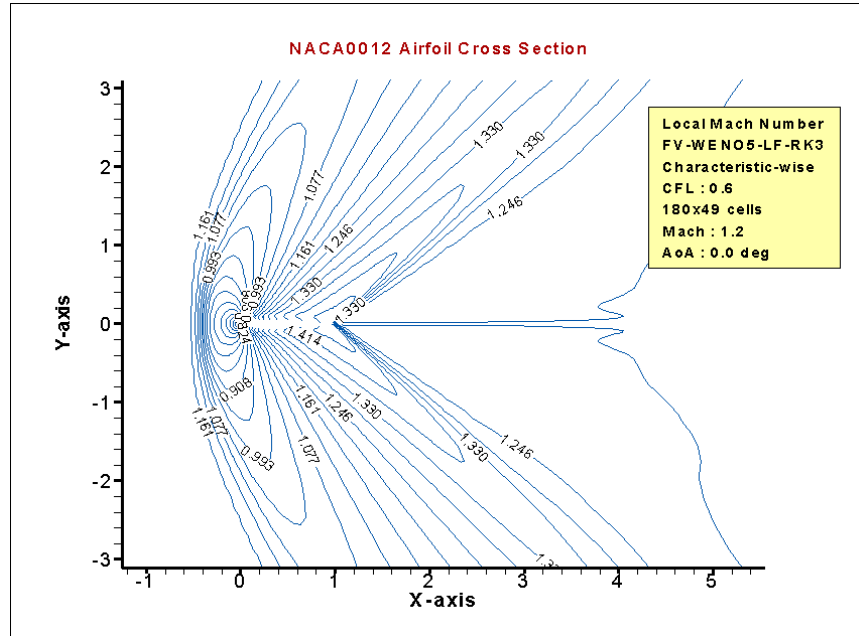


**Figure 10.70 :** Line Contours of NACA0012 (Case-5 Local Mach Number)

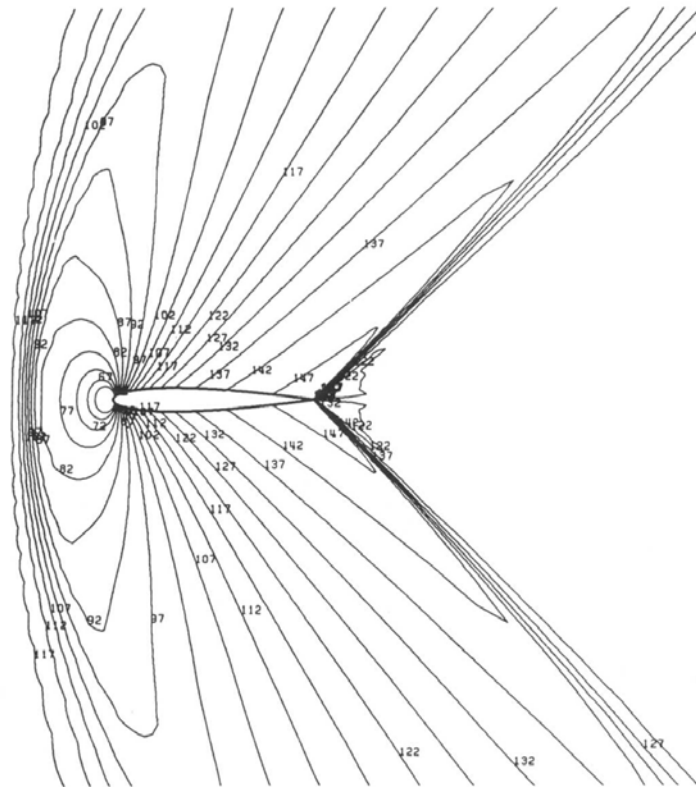


**Figure 10.71 :** Line Contours (Case-5 Local Mach Number), taken from [35]

### 10.2.6.6 NACA0012 Airfoil Cross Section (Case-6 - $M = 1.2$ , $\alpha = 7^\circ$ )

This test case uses supersonic and non-symmetric flow. Mach number for the flow is 1.2 and the airfoil makes 7 degrees angle of attack.

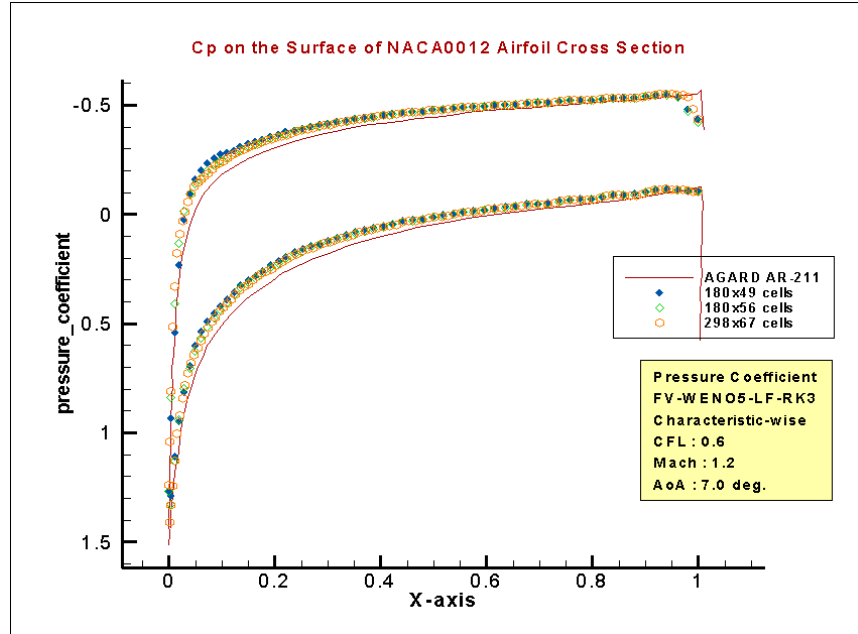


**Figure 10.72 :**  $C_p$  Distribution of NACA0012 (Case-6)

The pressure coefficient distribution on the surface of airfoil is given in figure 10.72 above. Comparison of 180x49 and 180x56 grids shows that, there is an improvement by using 180x56 instead of 180x49 grid. In addition, 298x67 and 180x56 grids give nearly the same results.

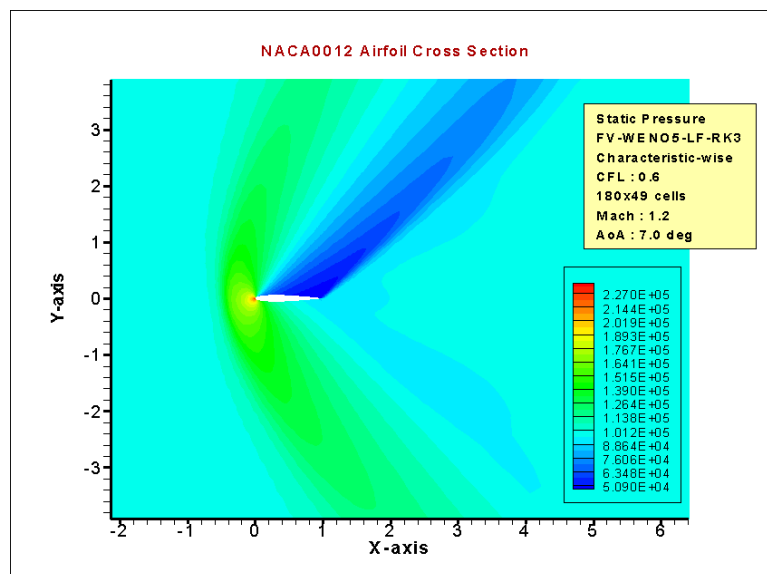


**Figure 10.73 :** Flood Contours of NACA0012 (Case-6 Static Pressure)

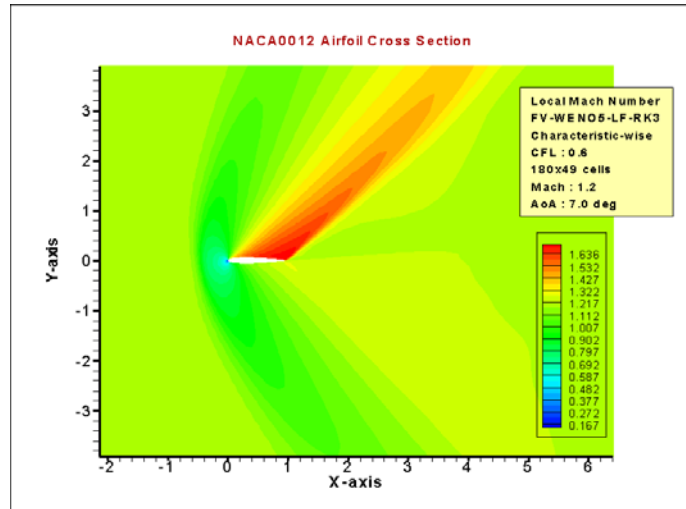


**Figure 10.74 :** Flood Contours of NACA0012 (Case-6 Local Mach Number)

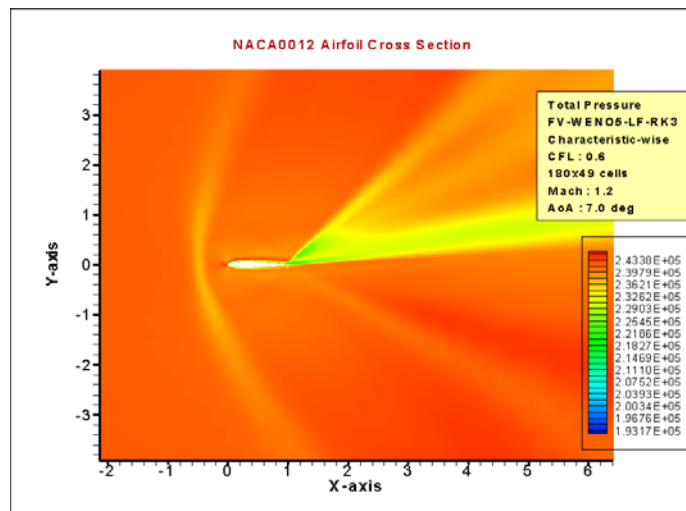


**Figure 10.75 :** Continuous Flood Contours of NACA0012 (Case-6 Total Pressure)

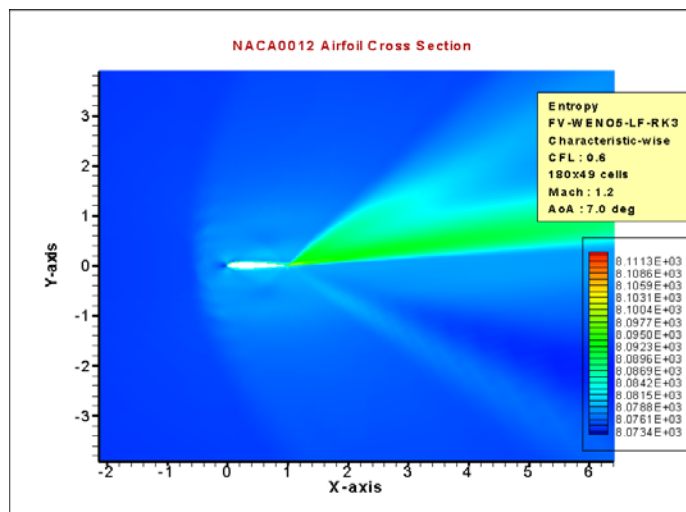


**Figure 10.76 :** Continuous Flood Contours of NACA0012 (Case-6 Entropy)

165

**Table 10.7 :  $C_L$  and  $C_D$  Values Comparison for Test Cases**

|                 |        | Case-1    |          | Case-2    |        | Case-3 |        |
|-----------------|--------|-----------|----------|-----------|--------|--------|--------|
|                 |        | $C_L$     | $C_D$    | $C_L$     | $C_D$  | $C_L$  | $C_D$  |
| AGARD-AR-211    | N9     |           |          | 0,3632    | 0,0230 | 0,3584 | 0,0580 |
| Current Code    | 180x49 | 1,67E-11  | 8,47E-03 | 0,2663    | 0,0184 | 0,2272 | 0,0533 |
| FV-WENO5-LF-RK3 | 180x56 |           |          | 0,2656    | 0,0185 |        |        |
|                 | 298x67 |           |          | 0,2669    | 0,0176 | 0,2286 | 0,0532 |
|                 |        | Case-4    |          | Case-5    |        | Case-6 |        |
|                 |        | $C_L$     | $C_D$    | $C_L$     | $C_D$  | $C_L$  | $C_D$  |
| AGARD-AR-211    | N9     |           | 0,1084   |           | 0,0960 | 0,5138 | 0,1538 |
| Current Code    | 180x49 | 8,67E-13  | 0,1077   | 3,42E-13  | 0,0957 | 0,5028 | 0,1547 |
| FV-WENO5-LF-RK3 | 180x56 |           |          |           |        | 0,5045 | 0,1536 |
|                 | 298x67 | -6,75E-15 | 0,1074   | -1,86E-15 | 0,0955 | 0,5076 | 0,1534 |

In table 10.7 above, the computed lift and drag coefficients for the test cases are compared with the computed values in [35]. These results are obtained for 0.05 seconds like the results given before for NACA0012 airfoil cross section. For example for Case-6, this instance corresponds to; 8382 time steps for 180x49, 12036 time steps for 180x56, and 16283 time steps for 298x67 grid domain. It is observed that, there isn't any significant increase in the values for finer grids (except for Case-6), and also sometimes decreases observed which may due to extra time steps, which gives extra dissipation to the converged solution.

These results show that, the zero lift coefficient values for symmetric flows are well achieved. For Case-2 and Case-3 the results shows a significant deviation from the reference values, however, for Case-4, Case-5, and Case-6 the results shows good agreement with the reference values. Case-2 and Case-3 correspond to a relatively low Mach number compared to the subsequent cases. It is interesting to see that, when the Mach number increases to a supersonic regime results are getting better. Eventually, we can claim here two things, first one is; subsonic boundary condition, which is working well for low order methods, works poor for high order, and the second one is; the structured WENO5 is vulnerable to the smoothness of the grid in all flow regimes, and its vulnerability decreases for the supersonic regime.

For transonic cases one can implement multiblock approach and observe the differences in the results. This kind of approach is implemented but didn't give better results for this test case. This may be due to some bugs in the implementation, but unfortunately further investigations isn't possible for these cases in this thesis project due to time restrictions for deadline.

### 10.2.7 Planar Shock Interaction with NACA0018 Airfoil Cross Section

This test case simulates two-dimensional planar shock traveling towards a NACA0018 airfoil cross section which has a high angle attack of 30 degrees. This case is conducted experimentally by Madella [37] and Mandella and Bershader [38]. In the experiment a planar shock is generated in a shock tube, this shock travels down the tube and exits to a broader space. Airfoil cross section is positioned at a distance of  $3\frac{1}{3}$  times the shock tube height away from the tube exit. The shock starts to diffract as soon as it leaves the shock tube, and forms a curved shock wave, which finally impinges on the airfoil. This curved shock starts to lose its energy as soon as it exits from the shock tube and at the instant of impingement the Mach number decreases approximately to 1.5.

Due to positive angle of attack of the airfoil, interesting shock reflection features are seen with this case. There is a transition from a short moment of compression to expansion on the upper surface, and compression on the lower surface. Along upper surface, shock reflection retains the regular reflection up to compressive region and then makes a rapid transition from regular to Mach reflection. A single Mach reflection forms a triple point from which the Mach stem, contact surface, and reflected shock emanate. Also a vortex starts to form and grows very slowly at the upper nose due to sudden but mild expansion. Meanwhile, the transition process develops gradually but rather strongly from a regular to a Mach reflection on the compressive lower surface. Eventually, the Mach stem developed on the lower side wraps around the trailing edge of the airfoil and a vortex is generated due to sudden strong expansion and the sharp trailing edge [36].

For the numerical implementation of this problem case, the curved shock moving towards the airfoil is modeled as a moving planar shock with constant velocity for simplicity. The outer boundary is chosen to be 10 chords away from the airfoil. C-Grid, which has 298x79 cells, is used and given in figure 10.79. An incident shock is placed half chords away in front of the leading edge. Non-reflecting boundary condition is used for the trailing boundaries, cut boundary condition is used for the C-Grid cut, wall boundary condition is used for the surface of the airfoil, and a problem specific boundary condition, which simulates right moving shock, is used for the C-shaped outer boundary. The Mach number of the shock is 1.5.

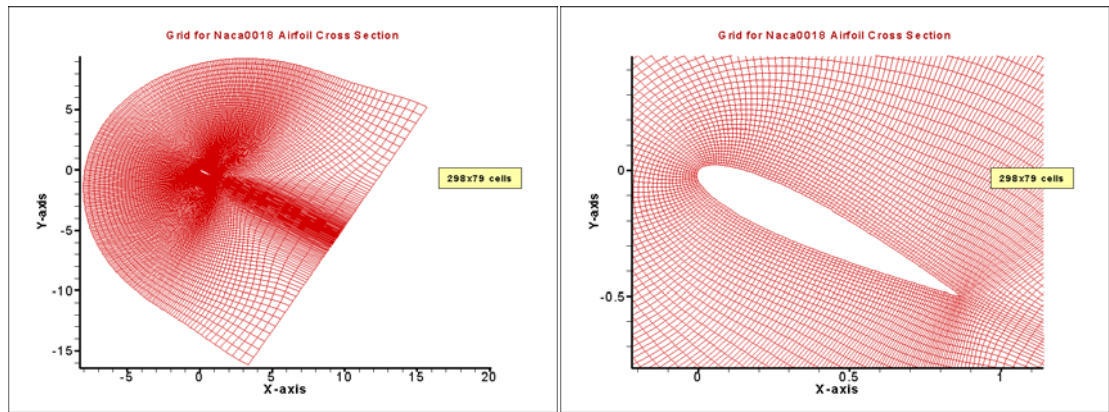


**Figure 10.79 :** Grid for NACA0018 Airfoil Cross Section (298x79)

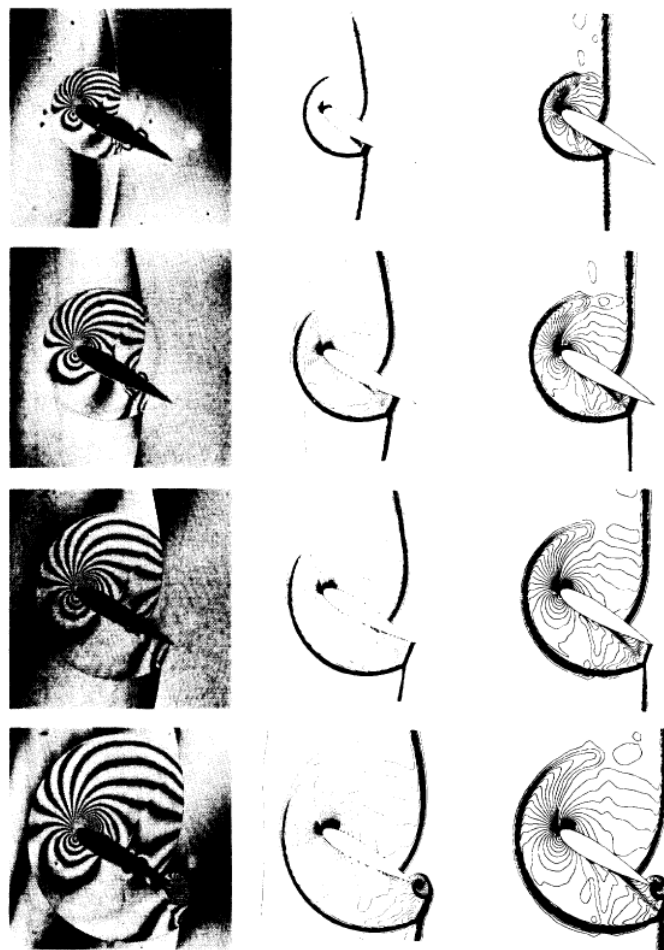


**Figure 10.80 :** Density Contours at Some Time Instances, taken from [36]

Figure 10.80 is taken from [36], these density contours are computed with a time-split symmetric TVD scheme using a curved shock in the middle, and a planar shock on the right. And the interferograms are given on the left at some time instances.

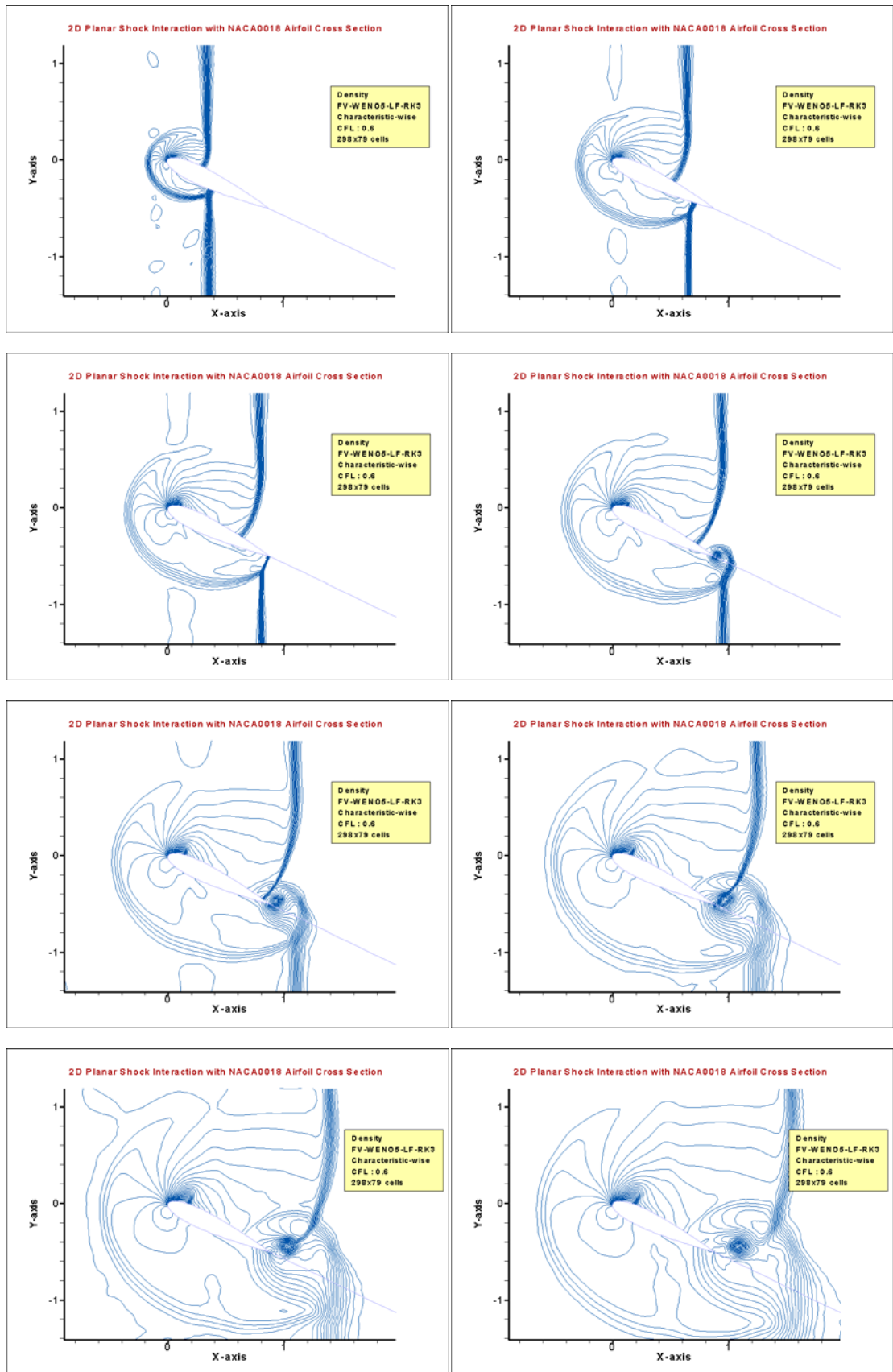


**Figure 10.81 : Line Contours (Density) at Some Time Instances**

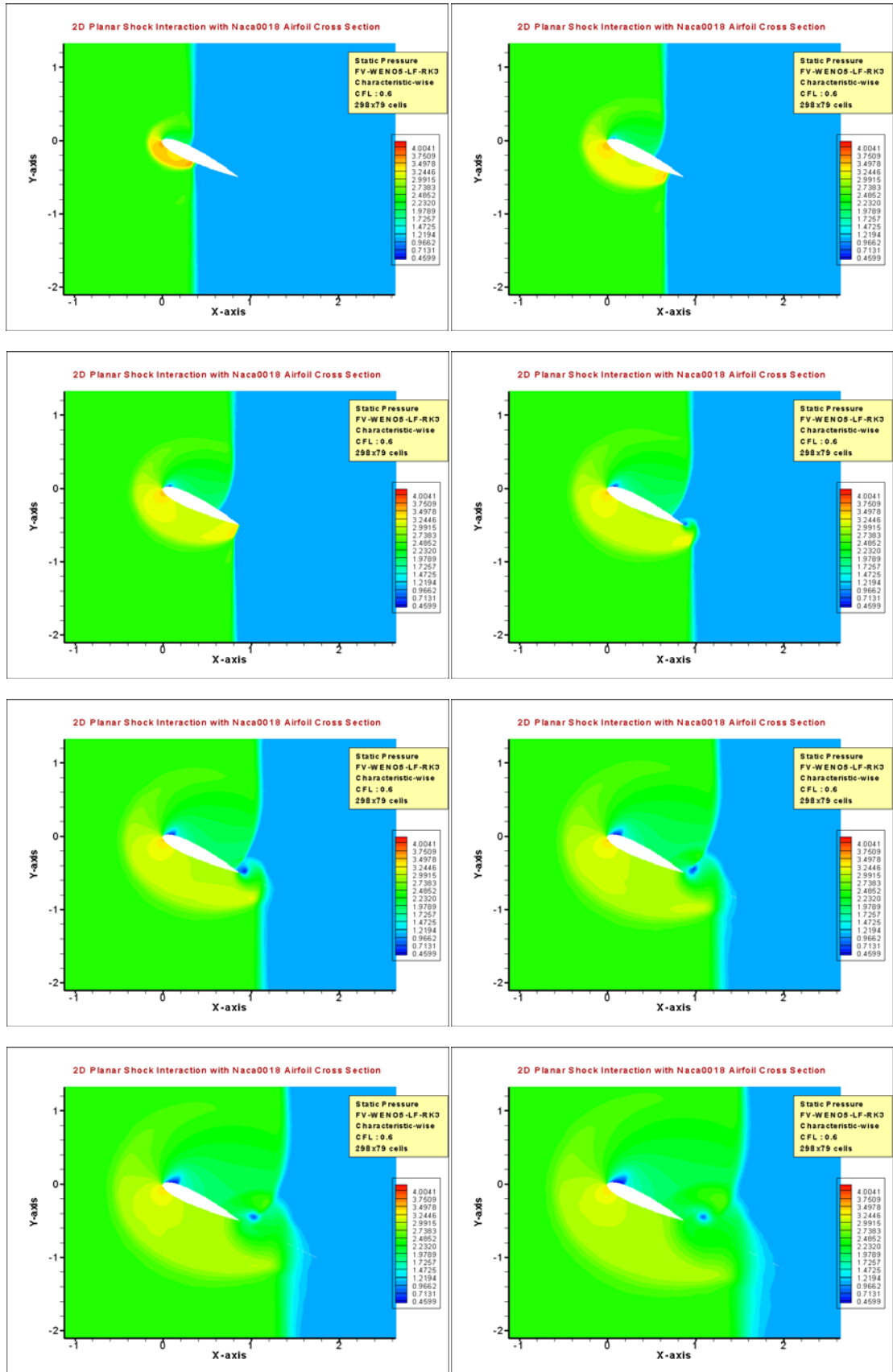


**Figure 10.82 : Flood Contours (Static Pressure) at Some Time Instances**

In figure 10.81, density contours are given at some time instances after the impingement of the shock with the airfoil cross section. It can be seen that results are identical with figure 10.80. The vortex which is formed at the trailing edge and which is starting to develop at the nose can be seen. In figure 10.82, flood contours of static pressure at the same time instances, which are used in figure 10.81, are given.

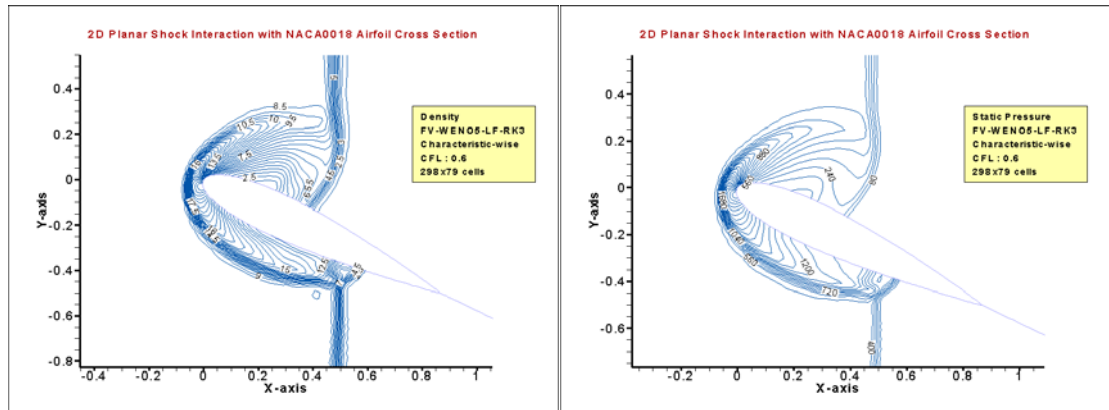


**Figure 10.83 :** Line Contours (Left:Density, Right:Static Pressure)

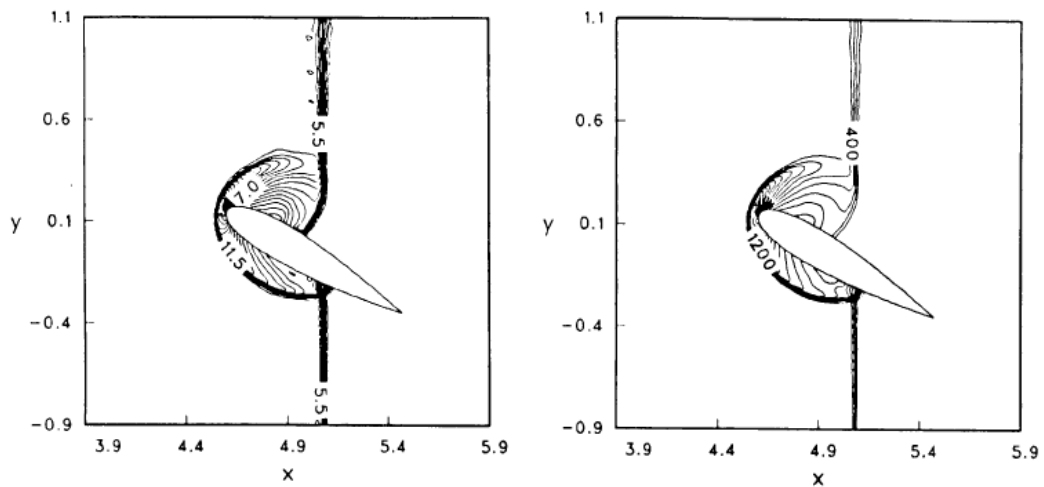


**Figure 10.84 :** Line Contours (Left:Density, Right:Static Pressure), taken from [36]

Figure 10.83 presents density and static pressure contours at some time instance. This case involves a Mach 20 shock interaction with NACA0018 airfoil cross section at the same angle of attack with the previous case, which is 30 degrees. The grid used for this case is the same grid used for the previous 1.5 Mach case. The results show good agreement with the results from [36], which is given in figure 10.84 above. In addition, local order reduction has to be used for Mach 20 case.

### 10.3 CPU Time Comparison for 2D Results

This subsection compares CPU times for two-dimensional results. The tables below give CPU times in seconds. To run the code a Laptop PC has been used, which has an AMD Athlon 64 3400+ (2200 MHZ) CPU. Code is compiled with Compaq Visual Fortran 6.6.C in Release mode. Vortex evolution problem is selected for the test case of CPU time comparisons, which uses 80x80 cells, characteristic-wise decomposition and a CFL number of 0.6 by default. In addition to that, table 10.8 uses a different code from the rest, which can handle FD and FV type decompositions in one code.

**Table 10.8 :** 2D CPU Time Comparison (Case-1)

|              |    | Char-wise | Comp-wise |
|--------------|----|-----------|-----------|
| WENO5-LF-RK3 | FD | 35,16     | 26,20     |
|              | FV | 122,22    | 96,98     |

**Table 10.9 :** 2D CPU Time Comparison (Case-2)

|          |     | RK3    | RK4    |
|----------|-----|--------|--------|
| FV-WENO5 | LF  | 163,05 | 216,17 |
|          | LLF | 168,98 | 223,78 |

**Table 10.10 :** 2D CPU Time Comparison (Case-3)

| with Order Reduction on | RK3    | RK4    |
|-------------------------|--------|--------|
| FV-WENO5-LF             | 168,06 | 223,86 |

**Table 10.11 :** 2D CPU Time Comparison (Case-4)

|                 | 80x80  | 160x160 |
|-----------------|--------|---------|
| FV-WENO5-LF-RK3 | 163,05 | 1362,92 |

**Table 10.12 :** 2D CPU Time Comparison (Case-5)

|                 | with Coord.Trn. | with Non-Uni.Coeff. |
|-----------------|-----------------|---------------------|
| FV-WENO5-LF-RK3 | 195,16          | 442,16              |

In table 10.8 finite difference and finite volume decompositions are compared with component- and characteristic-wise decompositions. It can be seen that, FV has approximately 3.5 times more cost than FD because of the obligation to use Gaussian point integration, and component-wise decomposition is %20 to %25 cheaper than characteristic-wise decomposition. In table 10.9 LF and LLF fluxes, which uses

either RK3 or RK4 are compared. It is observed that LF is approximately %3.5 cheaper than LLF, and RK3 is approximately %24.5 cheaper than RK4. Table 10.10 compares CPU times with order reduction switch on. In this mode, code has to compare the reconstructed values to the cell center values and decides if there is a need to make a local order reduction. It is observed that having this switch on (without a need for reduction, only comparison of reconstructed values) is %3 to %3.5 more costly than having this switch off. In table 10.11 the domain is enlarged 4 times and the calculated CPU times presented. It can be seen that the CPU time for 160x160 cells is approximately 8.36 times bigger than 80x80 cells. In table 10.12 a non-uniform grid, which is given in figure 10.16, is used with coordinate transformation and non-uniform coefficients. It is observed that using non-uniform coefficients has approximately 2.26 times more cost than a coordinate transformation.

Throughout this section two-dimensional results and the discussion about them are given. From these results we can claim that, WENO gives very good results for Cartesian uniform grid and for curvilinear grids, which provide the grid smoothness restrictions, like flow past a cylinder problem. However, like NACA0012 airfoil for Case-2 and Case-3 the pressure, lift and drag coefficient results are not satisfactory. It is believed to be a result of the smoothness of the grid, however, it can be a consequence of the second order centered difference metric calculations, or the implementation of a subsonic boundary condition that is not specially designed for a high order method, or some combination of them (more investigations have to be done for it).

Eventually, we can suggest that, for a high order method like WENO, a multiblock structured approach or an unstructured decomposition should be studied for this kind of highly skewed curvilinear domains.

## 11. CONCLUSION

The aim of this thesis project is to make computations around airfoil cross sections by using WENO, which is a high order accurate scheme. WENO is used at a 5<sup>th</sup> order of accuracy level in this thesis project. Both finite difference and finite volume decompositions are implemented to the scheme, Lax-Friedrichs flux splitting and Lax-Friedrichs numerical flux are used for flux calculations. For time discretization; 3<sup>rd</sup> order TVD Runge-Kutta and 4<sup>th</sup> order non-TVD Runge-Kutta algorithms are used.

First of all, a comprehensive literature survey is done, then, after understanding the mathematical background of WENO scheme, code generation process started with scalar one-dimensional problems. And in the end, we are able to calculate and present flow field around airfoil cross sections using WENO.

Throughout this thesis report, all the information and calculation results are presented in the following order. First of all, the mathematical infrastructure of this work is discussed, which are; Euler equations for the inviscid fluid flow and polynomial approximations for making the reconstruction work. After that, these general discussions are specialized for ENO and WENO algorithms. ENO is discussed because it is the main background of WENO algorithm. After the discussions about one-dimensional space, these discussions are extended to second dimension. At this stage, it is observed that, finite difference and finite volume cases are separating from each other to some extent for more than one-dimensional space. Thus, Gaussian quadrature integration point rule is discussed for finite volume case.

In section 6, piece-wise linear and piece-wise parabolic WENO schemes are generalized for non-uniform, structured, rectangular, Cartesian domains. In addition, with the help of these expressions, more than two point Gaussian integration rule can be used for a uniform or non-uniform domain. It has to be stated here that, we haven't encountered with such a generalization in the literature so far.

Some number of boundary conditions, ghost cell approach, and after that, coordinate transformation from curvilinear to rectangular domain is discussed in the text. At the end of these discussions, results are given for one-dimensional and two-dimensional cases in section 9 and 10 respectively.

Some of these test cases are smooth problems, some of them containing discontinuities, and some of them are more demanding problems; like shock interaction with complex flow structures, or long time simulations, yet another one is the demanding problem of Mach 20 shock interacting with airfoil. In all these problems we observed that WENO give good and accurate results.

These implementations in this thesis project yields good results, however, it is observed that there is room for improvements. First of all, the Lax-Friedrichs flux is one of the most diffusive from all the candidate numerical flux algorithms, yet the simplest of all. In the literature other numerical flux algorithms have been investigated so far, some of them include; HLL, HLLC, MUSTA, FORCE, FLIC, and WAF. Some of these fluxes can be implemented for improvement. In addition to that, some other versions of WENO reconstruction, which were given in literature so far, can be used, like Central WENO, Weighted Power ENO ... etc.

High order methods depend on the smoothness of the grid far more than low order methods. Therefore, in curvilinear coordinates, using a single block structured grid has some risks for WENO. In order to reduce those risks, multiblock approach and unstructured decomposition have to be investigated a lot more. And order reduction can be implemented to the boundary regions in order to reduce ghost cell numbers. In addition to these, for a recent work; the simulation of flow field around an aircraft with a high order method, which is WENO, we refer to the reference [39].

Eventually, computational cost is yet another issue for high order methods. The restriction to achieve the high order accuracy for the numerical flux at the cell boundaries, turns finite volume decomposition for a piece-wise parabolic WENO to be a demanding algorithm for a computer. Thus, we can say that, a high order scheme needs a high order effort. This is a certain drawback for the implementation, but the CPU powers are increasing day-by-day, therefore, this drawback will not be an important issue in the near future.

## REFERENCES

- [1] **Shu, C.W.**, 1999. High Order ENO-WENO Schemes for Computational Fluid Dynamics in *High-Order Methods for Computational Physics*, Springer-Verlag Berlin Heidelberg, 439-582.
- [2] **Shu, C.W.**, 1997. Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws, *ICASE REPORT*, **97-65**.
- [3] **Shu, C.W., and Osher, S.**, 1988. Efficient Implementation of Essentially Non-oscillatory Shock-Capturing Schemes, *Journal of Computational Physics*, **77**, 439-471.
- [4] **Roe, P.L.**, 1981. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes, *Journal of Computational Physics*, **43**, 357-372.
- [5] **Abgrall, R. et al.**, 1999. High Order Approximations for Compressible Fluid Dynamics on Unstructured and Cartesian Meshes in *High-Order Methods for Computational Physics*, Springer-Verlag Berlin Heidelberg, 1-67.
- [6] **Jiang, G.S., and Shu, C.W.**, 1996. Efficient Implementation of Weighted ENO Schemes, *Journal of Computational Physics*, **126**, 202-228.
- [7] **Laney, C.B.**, 1998. Computational Gasdynamics, Cambridge University Press.
- [8] **Crnjaric-Zic, N., et al.**, 2006. On the Different Flux Splittings and Flux functions in WENO Schemes for Balance Laws, *Computers & Fluids*, **35**, 1074-1092.
- [9] **Harten, A., et al.**, 1987. Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III, *Journal of Computational Physics*, **71**, 231-303.
- [10] **Liu, X.D., S. Osher, and Chan, T.**, 1994. Weighted Essentially Non-oscillatory Schemes, *Journal of Computational Physics*, **115**, 200-212.
- [11] **Blazek, J.**, 2001. Computational Fluid Dynamics: Principles and Applications, Elsevier Science Ltd. The Netherlands.
- [12] **Suresh, A., and Huynh, H.T.**, 1997. Accurate Monotonicity-Preserving Schemes with Runge-Kutta Time Stepping, *Journal of Computational Physics*, **136**, 83-99.

- [13] **Qui, J., and Shu, C.W.,** 2002. On the Construction, Comparison, and Local Characteristic Decomposition for High-Order Central WENO Schemes, *Journal of Computational Physics*, **183**, 187-209.
- [14] **Aktürk, A.,** 2005. Two-Dimensional Finite Volume Weighted Essentially Non-Oscillatory Euler Schemes with Different Flux Algorithms, *M.Sc. Thesis*, METU, Department of Aerospace Engineering, Ankara.
- [15] **Gottlieb, S., and Shu, C.W.,** 1998. Total Variation Diminishing Runge-Kutta Schemes, *Mathematics of Computation*, **67**, 73-85.
- [16] **Hu, C., and Shu, C.W.,** 1999. Weighted Essentially Non-Oscillatory Schemes on Triangular Meshes, *Journal of Computational Physics*, **150**, 97-127.
- [17] **Titarev, V.A., and Toro, E.F.,** 2004. Finite-Volume WENO Schemes for Three-Dimensional Conservation Laws, *Journal of Computational Physics*, **201**, 238-260.
- [18] **Smit, J., et. al.,** 2005. Grid Adaptation with WENO Schemes for Non-Uniform Grids to solve Convection-Dominated Partial Differential Equations, *Chemical Engineering Science*, **60**, 2609-2619.
- [19] **LeVeque, R.J.,** 2002. Finite Volume Methods for Hyperbolic Problems, Cambridge University Press.
- [20] **Frink, N.T.,** 1994. Recent Progress Toward a Three-Dimensional Unstructured Navier-Stokes Flow Solver, 32<sup>nd</sup> *Aerospace Sciences Meeting*, Reno, Nevada, USA, January 10-13, AIAA-94-0061.
- [21] **Mazaheri, K., Roe, P.L.,** 1997. Numerical Wave Propagation and Steady-State Solutions: Soft Wall and Outer Boundary Conditions, *AIAA Journal*, **35**, 965-975.
- [22] **Whitfield, D.L., Janus, J.M.,** 1984. Three-Dimensional Unsteady Euler Equations Solution Using Flux Vector Splitting, *AIAA Paper* 84-1552.
- [23] **Tanehill, J.C., Dale, A.A., Pletcher R.H.,** 1997. Computational Fluid Mechanics and Heat Transfer, Second Edition, Taylor & Francis.
- [24] **NASA Langley Research Center,** 1997. CFL3D User's Manual (Version 5.0)
- [25] **Shu, C.W., et. al.,** 1991. High Order ENO Schemes Applied to Two- and Three-Dimensional Compressible Flow, *ICASE REPORT*, **91-38**.
- [26] **Jiang, G.S., and Wu, C.C.,** 1999. A High-Order WENO Finite Difference Scheme for the Equations of Ideal Magnetohydrodynamics, *Journal of Computational Physics*, **150**, 561-594.
- [27] **Shu, C.W., and Osher, S.,** 1989. Efficient Implementation of Essentially Non-oscillatory Shock-Capturing Schemes II, *Journal of Computational Physics*, **83**, 32-78.

- [28] **Yang, H.**, 1990. An Artificial Compression Method for ENO Schemes: The Slope Modification Method, *Journal of Computational Physics*, **89**, 125-160.
- [29] **Titarev, V.A., and Toro, E.F.**, 2005. WENO Schemes Based on Upwind and Centered TVD Fluxes, *Computers & Fluids*, **34**, 705-720.
- [30] **Woodward, P., and Colella, P.**, 1984. The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks, *Journal of Computational Physics*, **54**, 115-173.
- [31] **Cai, W., Shu, C.W.**, 1991. Uniform High Order Spectral Methods for One and Two Dimensional Euler Equations, *ICASE REPORT*, **91-26**.
- [32] **Remacle, J.F., et. al.**, 2003. An Adaptive Discontinuous Galerkin Technique with an Orthogonal Basis Applied to Compressible Flow, *SIAM Review*, **45**, 53-72.
- [33] **Yamaleev, N.K, Carpenter, M.H.**, 2002. On Accuracy of Adaptive Grid Methods for Captured Shocks, *NASA/TM-2002-211415*.
- [34] **Sultan, H.M**, 2005. Development of an Unstructured Cartesian Flow Solver, *Dr-Ing. Thesis*, RWTH-AACHEN University, Institute of Aerodynamics, Aachen.
- [35] **North Atlantic Treaty Organization**, 1985. Test Cases for Inviscid Flow Filed Methods, *Report of Fluid Dynamics Panel Working Group 07*, North Atlantic Treaty Organization Advisory Group for Aerospace Research And Development, AGARD-AR-211.
- [36] **Yee, H.C**, 1989. A Class of High-Resolution Explicit and Implicit Shock-Capturing Methods, *NASA/TM-101088*.
- [37] **Mandella, M.**, 1987. Experimental and Analytical Studies of Compressible Vortices, *Ph.D. Thesis*, Stanford University, Department of Applied Physics, Calif.
- [38] **Mandella, M., and Bershader, D.**, 1987. Quantitative Study of Compressible Vortices: Generation, Structure, and Interaction with Airfoils, January 10-13, AIAA-87-0328.
- [39] **Ladeinde, F., et. al.**, 2006. The First High-Order CFD Simulation of Aircraft: Challenges and Opportunities, *44<sup>th</sup> AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, USA, January 9-12, AIAA-2006-1526.

## APPENDIX-A. ALTERNATIVE WAY FOR FINITE DIFFERENCE WENO

This appendix section discusses an alternative way for finite difference WENO approximation. This alternative way is mathematically equivalent to the normal way, which discussed in section 4.2, but more compact and need less computational effort from the code. This way of approximation is presented in [26].

As we already know, the reconstructions for three  $3^{rd}$  order candidate stencils for a  $5^{th}$  order WENO at  $x = x_{i+\frac{1}{2}}$  are as the following (see figure 4.2, table 3.1, and equation 5.16).

$$u_{i+\frac{1}{2}}^0 = \frac{1}{3}\bar{u}_{i-2} - \frac{7}{6}\bar{u}_{i-1} + \frac{11}{6}\bar{u}_i \quad (\text{A.1a})$$

$$u_{i+\frac{1}{2}}^1 = -\frac{1}{6}\bar{u}_{i-1} + \frac{5}{6}\bar{u}_i + \frac{1}{3}\bar{u}_{i+1} \quad (\text{A.1b})$$

$$u_{i+\frac{1}{2}}^2 = \frac{1}{3}\bar{u}_i + \frac{5}{6}\bar{u}_{i+1} - \frac{1}{6}\bar{u}_{i+2} \quad (\text{A.1c})$$

Then, the WENO approximation takes the form

$$u_{i+\frac{1}{2}} = w_0 u_{i+\frac{1}{2}}^0 + w_1 u_{i+\frac{1}{2}}^1 + w_2 u_{i+\frac{1}{2}}^2$$

replacing the weight  $w_1 = 1 - w_0 - w_2$  and using equations (A.1), we get

$$\begin{aligned} u_{i+\frac{1}{2}} &= \frac{1}{12}(-\bar{u}_{i-1} + 7\bar{u}_i + 7\bar{u}_{i+1} - \bar{u}_{i+2}) \\ &+ \frac{1}{3}w_0(\bar{u}_{i-2} - 3\bar{u}_{i-1} + 3\bar{u}_i - \bar{u}_{i+1}) + \frac{1}{6}\left(w_2 - \frac{1}{2}\right)(\bar{u}_{i-1} - 3\bar{u}_i + 3\bar{u}_{i+1} - \bar{u}_{i+2}) \end{aligned} \quad (\text{A.2})$$

This can be rewritten in the form

$$u_{i+1/2}^- = \frac{1}{12}(-\bar{u}_{i-1} + 7\bar{u}_i + 7\bar{u}_{i+1} - \bar{u}_{i+2}) - \varphi_N\left(\Delta\bar{u}_{i-3/2}, \Delta\bar{u}_{i-1/2}, \Delta\bar{u}_{i+1/2}, \Delta\bar{u}_{i+3/2}\right) \quad (\text{A.3})$$

where  $\Delta\bar{u}_{i+1/2} = \bar{u}_{i+1} - \bar{u}_i$  and

$$\varphi_N(a, b, c, d) = \frac{1}{3}w_0(a - 2b + c) + \frac{1}{6}\left(w_2 - \frac{1}{2}\right)(b - 2c + d) \quad (\text{A.4})$$

the linear (optimal) weights, nonlinear weights and smoothness indicators are defined respectively as,  $\varepsilon$  is from (4.20)

$$\alpha_0 = \frac{1}{(\varepsilon + \beta_0)^2}, \quad \alpha_1 = \frac{6}{(\varepsilon + \beta_1)^2}, \quad \alpha_2 = \frac{3}{(\varepsilon + \beta_2)^2} \quad (\text{A.5a})$$

$$w_0 = \frac{\alpha_0}{\alpha_0 + \alpha_1 + \alpha_2}, \quad w_2 = \frac{\alpha_2}{\alpha_0 + \alpha_1 + \alpha_2} \quad (\text{A.5b})$$

$$\beta_0 = 13(a - b)^2 + 3(a - 3b)^2 \quad (\text{A.5c})$$

$$\beta_1 = 13(b - c)^2 + 3(b + c)^2 \quad (\text{A.5d})$$

$$\beta_2 = 13(c - d)^2 + 3(3c - d)^2 \quad (\text{A.5e})$$

Similarly, the reconstruction for the other cell boundary  $x = x_{i-1/2}$  in the stencil, see figure 3.2, is written as the following (with shifting one grid point)

$$u_{i+1/2}^+ = \frac{1}{12}(-\bar{u}_{i-1} + 7\bar{u}_i + 7\bar{u}_{i+1} - \bar{u}_{i+2}) + \varphi_N\left(\Delta\bar{u}_{i+5/2}, \Delta\bar{u}_{i+3/2}, \Delta\bar{u}_{i+1/2}, \Delta\bar{u}_{i-1/2}\right) \quad (\text{A.6})$$

For finite difference solutions, we can use Lax-Friedrichs flux splitting, and for systems solutions we can use characteristic decomposition, as already discussed in section 4.3.2 and 4.3.3. For a flux splitting, we use equation (4.33), so, equation (A.3) and (A.6) are added together to form the numerical flux at  $x = x_{i+1/2}$  that is given in equation (A.7) below.

$$\begin{aligned}\hat{f}_{i+1/2} = \frac{1}{12} \left( -\hat{f}_{i-1} + 7\hat{f}_i + 7\hat{f}_{i+1} - \hat{f}_{i+2} \right) &- \varphi_N \left( \Delta \hat{f}_{i-3/2}^+, \Delta \hat{f}_{i-1/2}^+, \Delta \hat{f}_{i+1/2}^+, \Delta \hat{f}_{i+3/2}^+ \right) \\ &+ \varphi_N \left( \Delta \hat{f}_{i+5/2}^-, \Delta \hat{f}_{i+3/2}^-, \Delta \hat{f}_{i+1/2}^-, \Delta \hat{f}_{i-1/2}^- \right) \quad (\text{A.7})\end{aligned}$$

Characteristic decompositions for systems can be applied in the following manner.

$$\begin{aligned}\hat{F}_{i+1/2} = \frac{1}{12} \left( -\hat{F}_{i-1} + 7\hat{F}_i + 7\hat{F}_{i+1} - \hat{F}_{i+2} \right) \\ + \sum_{s=1}^m \left\{ -\varphi_N \left( L^s \cdot \Delta F_{i-3/2}^{+,s}, L^s \cdot \Delta F_{i-1/2}^{+,s}, L^s \cdot \Delta F_{i+1/2}^{+,s}, L^s \cdot \Delta F_{i+3/2}^{+,s} \right) \right. \\ \left. + \varphi_N \left( L^s \cdot \Delta F_{i+5/2}^{-,s}, L^s \cdot \Delta F_{i+3/2}^{-,s}, L^s \cdot \Delta F_{i+1/2}^{-,s}, L^s \cdot \Delta F_{i-1/2}^{-,s} \right) \right\} R^s \quad (\text{A.8})\end{aligned}$$

where  $L^s$  and  $R^s$  states the corresponding component of left and right eigenvectors at  $x = x_{i+1/2}$ , respectively.

For finite difference decompositions, finding the numerical flux by this way reduces the number of operations in a code, and thus it is more efficient for long time simulations or large systems. CPU time comparison for this alternative way is given in section 9.3.

## **APPENDIX-B. INTRODUCTION TO THE CODE**

In this Appendix section, an introduction to the code generated for this thesis project will be given.

The code generation is one of the main and the challenging part of this thesis project. The aim of this code generation work was writing a flexible code, which can be easily implemented to different kind of problems and can be treated as a parametric 2D Euler solver.

First of all, I want to express my gratitude to Prof. Chi-Wang Shu, because of sending me his sample finite difference WENO code for vortex evolution problem via e-mail. Except this and some other sample codes, the code was generated from scratch. Firstly, a scalar one-dimensional finite difference code was written in FORTRAN 77. Then it was extended to include systems of conservation laws. After that, this code was extended to include both scalar and vectorial finite volume decompositions. At this stage, it was seen that writing in FORTRAN 77 is not feasible because of the expectations from the code, then all of the work upgraded to Fortran 90. Afterwards, the code was extended to include two-dimensional scalar and two vectorial problems (vortex evolution and shock vortex interaction) both for finite difference and finite volume decompositions. At that time, the complexity of the code had been getting to an unaffordable level because it was containing both scalar, vectorial, finite difference and finite volume case. Therefore, the code generation work was restricted to include only two-dimensional vectorial finite volume decomposition, and a new code in Fortran 90 generated for this case. In fact, finite volume decomposition for WENO is more difficult and costs more CPU time than finite difference decomposition, however, it was chosen, that is because it is more close to an unstructured decomposition than finite difference, and a future extension to an unstructured decomposition would be easier by that way.

The capabilities and features of the code can be listed as the following; an Euler structured finite volume 5<sup>th</sup> order WENO solver, either component- or characteristic-wise decompositions, can use global or local Lax-Friedrichs flux, can

implement  $3^{rd}$  order TVD or  $4^{th}$  order non-TVD time decompositions, can use non-uniform coefficients for rectangular Cartesian geometries that is discussed in section 6, a number of different boundary conditions, which is discussed in section 7, multiblock approach for conforming grids, ability to add dummy cells to the physical topology in general coordinates, can decide and implement local order reduction, ability to handle coordinate transformation from curvilinear to uniform rectangular domain, can use a convergence criteria, gives output in TECPLOT format, either time advancing or end time plots. Time advancing plots makes it simpler to create a movie for the problem cases. Sample input, topology and grid files are presented at the end of this Appendix section.

At this point order reduction will be discussed. As already briefly discussed at the end of section 9.2, order reduction is necessary for WENO in some cases. WENO reconstruction assigns nearly zero weight to the candidate stencils which cross a discontinuity. However, if the solution contains two discontinuities which are close to each other, the reconstruction procedure will not be able to find a smooth stencil and spurious oscillations can appear [17]. In addition to that, while finding a high order reconstruction polynomial for values which have a large difference between two consequent nodes and where one of them near a zero point, system may crash due to finding a negative pressure. In order to prevent from these effects local order reduction implemented to the code. This technique is working as the following; after finding reconstructions for each Gaussian point, these reconstructed values (density and pressure) are compared to the values of the cell centers. If the reconstructed value found to be above the value of the cell centers, which is scaled by a given constant, then code implements a local order reduction to that  $5^{th}$  order stencil. Firstly, implementing a  $3^{rd}$  order WENO and checking again, if the problem persists then implementing a MUSCL type reconstruction. After that, numerical flux can be calculated with these new values, details can be found in [17].

Global time stepping procedure for general coordinates is implemented from reference [11, page187]. A global time stepping must be used for unsteady calculations. Finding the spectral radius of the convective flux Jacobians in each direction, using the area of the cell and CFL number is sufficient to find the global time step value. This time step value has to be found once in each time step and the same value must be used for each Runge-Kutta time sub-steps.

Norm error calculations are implemented to the code.  $L_1$  and  $L_\infty$  errors are calculated for accuracy tests, which were given in tables in sections 9 and 10. And for implementation of convergence criteria  $L_2$ -Norms are calculated in density variable, which are normalized by the value of first iteration [11, page395]. Implementing convergence criteria and deciding for a converged solution is not easy for an unsteady solver, however, this criteria is implemented to the code especially for flow past a cylinder problem and NACA0012 airfoil calculations in order to follow from screen and decide the situation of the solution while running.

In code generation, especially for metrics calculations, boundary conditions and input files handling, Blazek's structured Euler solver, which comes with [11], is taken as a reference. Below, main input, grid topology, and grid coordinates input files are presented for the flow about NACA0012 airfoil problem. Afterwards, subroutine invocations chart (not a flow chart) of the code is given in figure B.1, which is generated by Understand for Fortran program.

```
*****
* Main Input File *
* Problem : Flow About NACA0012 Airfoil Cross Section *
* File Name : naca0012__input.txt *
* Note: - *
* *
!-----
##### Files/Names (max 40 char) #####
2D NACA0012 WENO5-LF FV !Name of the Problem (in)
naca0012__grd.txt !Name of the Grid File (in)
naca0012__tpl.txt !Name of the Grid Topology File (in)
naca0012.plt !Name of the Plot File (out)
naca0012__cnv.plt !Name of the Convergence History File (out)
../inputs_fv_2d/ !Location of Input Files (w.r.t. exe file) (in)
!-----
##### WENO-5th Parameters #####
#CFL Number
0.6 !CFL Number
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
#Characteristic-wise
Y !Y:Characteristic-wise - N:Component-wise
```

```

!--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
#Numerical Flux (Lax-Friedrichs Flux)
G                                !G:Global(LF) - L:Local(LLF)
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
#Runge-Kutta Time Discretization
3                                !3:TVD RK3 - 4:nonTVD RK4
!-----
##### Physics Parameters #####
#Flow Characteristics
E                                !I:Internal Flow - E:External Flow
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
#Gas Constants
1.4                              !Specific Heat Ratio
1005.0                          !Constant Pressure Spec. Heat Coeff. [J/kgK]
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
##Internal Flow Physics##
0.0                              !Inlet Total Pressure [Pa]
0.0                              !Inlet Total Temperature [K]
0.0                              !Inlet Flow Angle (w.r.t. x-axis) [deg]
0.0                              !Outlet Static Pressure [Pa]
0.0                              !Outlet Approx. Flow Angle (w.r.t. x-axis) [deg]
0.0                              !Inlet to Outlet Approx. Static Pressure Ratio
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
##External Flow Physics##
1.2                              !Infinity Mach Number
1.0E+5                          !Infinity Static Pressure [Pa]
288.0                           !Infinity Static Temperature [K]
N                                !Farfield Vortex Correction (Y:Yes N:No)
#Airfoil Parameters
0.25                            !x-Coordinate of Momentum Coefficient Ref. Point
0.0                              !y-Coordinate of Momentum Coefficient Ref. Point
1.0                              !Chord Length
7.0                              !Angle of Attack [deg]
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
#Problem Type#
0                                !0:Internal & External Flow Problems
                                !1:Vortex Evolution

```

```

!2:Shock Vortex Interaction
!3:Double Mach Reflection
!4:Forward Facing Step
!5:Supersonic Flow Past a Cylinder
!6:Planar Shock Interaction with an Airfoil

!--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
##Problem Dependent Physics (Prob 1 to 6)##
0.0          !Initial Flow Density
0.0          !Initial Flow Velocity-u
0.0          !Initial Flow Velocity-v
0.0          !Initial Flow Pressure
#Problem [1 & 2]#
0.0          !Vortex Strength
0.0          !Vortex Center Coordinate (in x-axis)
0.0          !Vortex Center Coordinate (in y-axis)
#Problem [2]#
0.0          !Critical Radius for Vortex Max. Strength
0.0          !Decay Rate of the Vortex
0.0          !Mach Number for the Shock
0.0          !Location of the Normal Shock (in x-axis)
#Problem [3 & 6]#
0.0          !Slope of Shock (w.r.t. x-axis) (for prob. 3)
0.0          !Mach Number of Shock
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
# End Time
0.05         !End Time (sec)
!-----
##### Order Reduction Parameters #####
N            !Y:(Yes) - N:(No)
1.0          !Order Reduction Coefficient
!-----
##### Convergence Parameters #####
N            !Use Convergence Criteria (Y:Yes - N:No)
1.0E-3       !Convergence Tolerance
!-----
##### Grid Parameters #####
#Multi-Block Grid

```

```

N                                !Y:(Yes) Multi-Block - N:(No) Single-Block
#Create Uniform or Non-Uniform Grid
N                                !Create Grid (Y:Yes N:No)
1                                !1:Uniform Cartesian - 2:Clustered Non-Uniform
0.0                             !x-axis Coordinate of Clustering Point
0.0                             !Clustering Parameter for x-axis (0<=pr<inf)
0.0                             !y-axis Coordinate of Clustering Point
0.0                             !Clustering Parameter for y-axis (0<=pr<inf)
!-----
##### Output File Parameters #####
#Plot Type
Y                                !Time Advanced Plot (Y:Yes) (N:No end_time plot)
0.002                           !Time Increment for Time Advanced Plot
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
#i-Cut / j-Cut Plot Locations
0.5                             !Location for i-Cut between [0-1] (0=0 1=imax)
0.5                             !Location for j-Cut between [0-1] (0=0 1=jmax)
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
#Quantities to Plot
Y                                !Density
Y                                !Velocity-u
Y                                !Velocity-v
Y                                !Static Pressure
Y                                !Total Pressure
N                                !Sound Speed
Y                                !Local Mach Number
N                                !Isentropic Mach Number
N                                !Static Temperature
N                                !Total Temperature
Y                                !Entropy
N                                !Static Enthalpy
N                                !Total Enthalpy
N                                !Total Pressure Loss
Y                                !Pressure Coefficient Over Wall (Cp)
!--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
#Number of Decimal Digits to Write (max. 15 digits)
15

```

```

*****

* Grid Topology Input File *
* Problem : Flow About NACA0012 Airfoil Cross Section *
* File Name : naca0012__tpl.txt *
* Note: If Grid Transformation (Y) Min.Max.Coords. Belong to Computational Grid *
* *

!-----
#Transform to Computational Grid
Y !Y:(Yes) Comp. Grid - N:(No) Physical Grid
!-----
#Cell Numbers
180 !Maximum Cell Number in i-Direction (imax)
49 !Maximum Cell Number in j-Direction (jmax)
!-----
#Coordinates
0.0 !Min. x-Coordinate
1.0 !Max. x-Coordinate
0.0 !Min. y-Coordinate
1.0 !Max. y-Coordinate
!-----
#Segments
6 !Number of Segments
!-----
#Boundary Conditions for Segments
# btype side sbeg send sside ssbeg ssend sblock
# farfield
5 1 1 49 0 0 0 0
# cut
2 2 1 30 2 180 151 1
# wall
7 2 31 150 0 0 0 0
# cut
2 2 151 180 2 30 1 1
# farfield
5 3 1 49 0 0 0 0
# farfield
5 4 1 180 0 0 0 0

```

```

*****
* Grid Coordinates Input File *
* Problem : Flow About NACA0012 Airfoil Cross Section *
* File Name : naca0012__grd.txt *
* Note: - *
* *
!-----
#Node Numbers
180 !Maximum Cell Number in i-Direction (imax)
49 !Maximum Cell Number in j-Direction (jmax)
!-----
#Grid Coordinates (x,y)
0.110000000000000E+02 0.000000000000000E+00
0.942231002863000E+01 0.000000000000000E+00
0.809349062184000E+01 0.000000000000000E+00
0.697427653827000E+01 0.000000000000000E+00
0.603159615745000E+01 0.000000000000000E+00
0.523759424752000E+01 0.000000000000000E+00
0.456880887192000E+01 0.000000000000000E+00
0.400547811337000E+01 0.000000000000000E+00
0.353095612966000E+01 0.000000000000000E+00
0.313122128614000E+01 0.000000000000000E+00
0.279446183096000E+01 0.000000000000000E+00
0.251072686994000E+01 0.000000000000000E+00
0.227163232798000E+01 0.000000000000000E+00
0.207011320823000E+01 0.000000000000000E+00
0.190021482881000E+01 0.000000000000000E+00
0.175691686802000E+01 0.000000000000000E+00
0.163598501910000E+01 0.000000000000000E+00
0.153384587078000E+01 0.000000000000000E+00
... ...
... ...
... ...
... ...
... ...

```

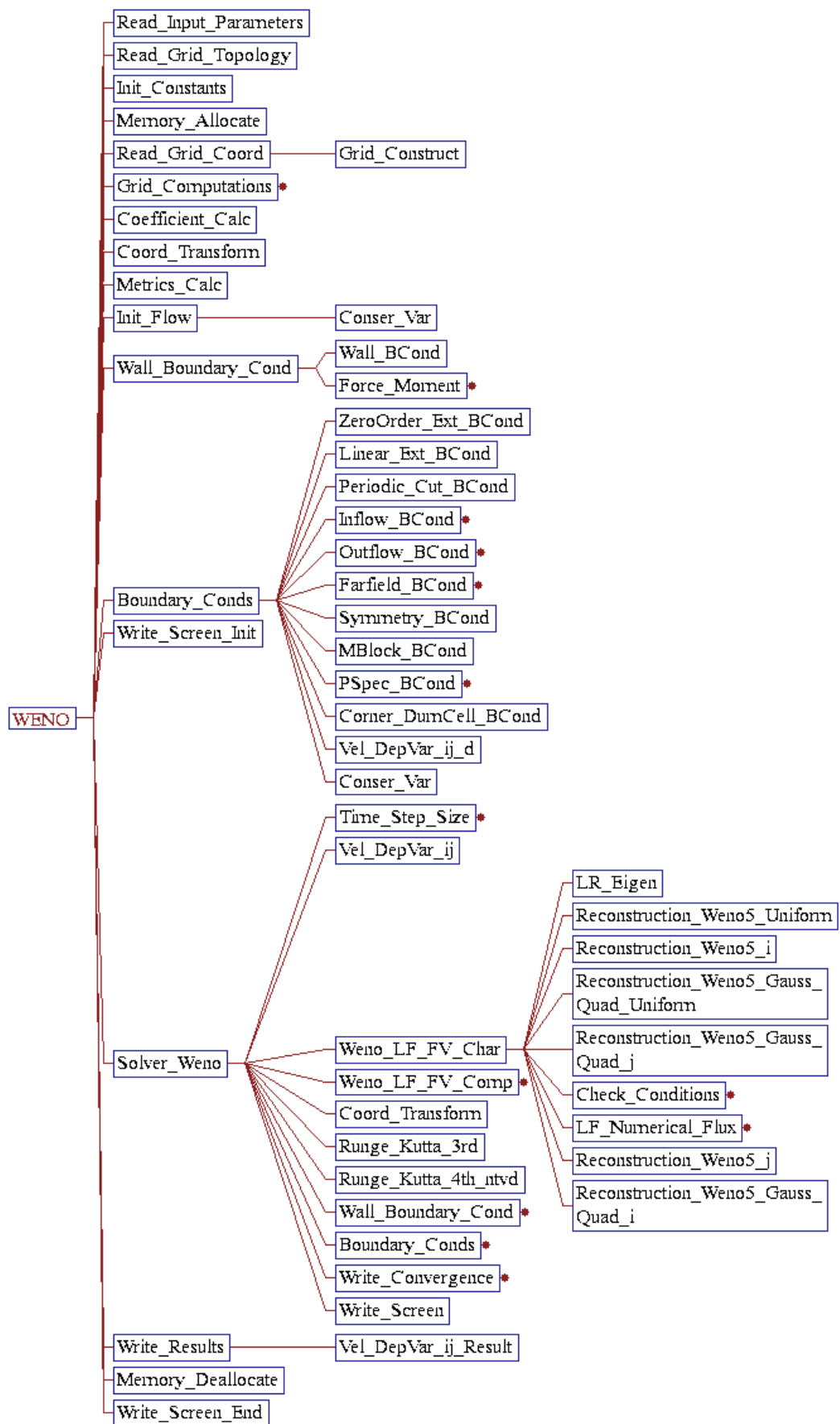


**Figure B.1 : Subroutine Invocation Chart**

## **CIRRICULUM VITAE**

Saygın Ayabakan was born in Ankara, Turkey on June 03, 1980. He attended to Bursa Science High School and Karsiyaka High School after graduating from Karsiyaka Anatolian School. He received his B.Sc. degree in Aeronautical Engineering from Aeronautics and Aerospace Engineering Department of Istanbul Technical University. He then entered the same department in Istanbul Technical University to have a M.Sc. degree in Aeronautical Engineering. He is currently hired by a company for a project based work, and working in TUBITAK Technology Development Area, in Gebze, Turkey. E-mail address: sayginayabakan@yahoo.com