Boundary Conditions in Computational Fluid Dynamics

by

LAWRENCE AOUN, B.Sc.

A thesis submitted to

The Faculty of Graduate Studies and Research

In partial fulfilment of

The requirements for the degree of

Master of Science

School of Mathematics and Statistics

Ottawa-Carleton Institute for Mathematics and Statistics

Carleton University

Ottawa, Ontario, Canada

© copyright

February 2004, Lawrence Aoun

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-89821-0

Our file Notre référence

ISBN: 0-612-89821-0

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

إلى والدي ... الذي علمني دروسا لا يمكن لأي مدرسة أن تعلمها
إلى والدي ... رمز الصبر والتضحية
إلى أخي ... شريكي في كل شيء

ABSTRACT

Subsonic outflow boundaries require the specification of one boundary condition in order to account for the incoming Riemann invariant. As the exact form of the incoming Riemann invariant at the outlet is often unknown, improvisations are made which subsequently introduce reflection errors into the numerical solution. A considerable effort has been put forth over the past ten years with the intention of circumventing the problems introduced when using classical outflow boundary treatments.

This study is a development of the existing literature dealing with boundary condition specification in CFD problems, with a particular focus on the buffer zone technique of Ta'asan and Nark. With its foundation in the original formulation proposed by Streett, to date, applications of Ta'asan and Nark's technique have been vague, with little emphasis having been placed on the appropriate selection of relevant parameters. In an attempt to remove much of the empiricism involved in the use of this and other similar approaches, detailed numerical studies were conducted based on the one-dimensional Euler equations in both their non-characteristic and characteristic form. The various results obtained here provide substantial incentive for researchers who are seeking viable outflow boundary condition alternatives, to consider the use of the buffer zone technique.

ABSTRACT	III
LIST OF FIGURES	VIII
LIST OF TABLES.....	XV
NOMENCLATURE	XVI
1.0 INTRODUCTION	1
1.1 Overview And A Brief History of Computational Fluid Dynamics	1
1.2 The Navier-Stokes Equations	3
1.3 Dimensional Analysis and the Simplification of the Governing Equations of Fluid Dynamics	6
1.4 Derivation of the Characteristic form of the Non-Linear Euler Equations in one Dimension.....	13
1.5 Discrete Solutions to the Equations of Fluid Flow	18
1.5.1 The Numerical Stability of Discretization Schemes	24
1.6 The Role of Riemann Invariants In Computational Fluid Dynamics	31
1.6.1 The Higher Dimensional Application of Riemann Invariants	36
1.7 Wave Reflection at Outflow Boundaries.....	38
1.7.1 Physical vs. Non-Physical Modes	41
2.0 LITERATURE SURVEY ON OUTFLOW BOUNDARY TREATMENT	48
2.1 Classification of Outflow Boundary Treatments	48
2.2 Non-Reflecting Boundary Conditions (NRBCs).....	49
2.2.1 Absorbing Boundary Conditions of Engquist and Majda	50
2.2.2 NRBCs of Hedstrom, Thompson, Poinso and Lele	52
2.2.3 NRBC of Rudy and Strikwerda	59

2.3	Non-Reflecting Buffer Zones (NRBZs).....	65
2.3.1	Buffer Zone Technique of Ta'asan and Nark	65
2.3.2	PML Methods of Berenger and Hayder, Hu, and Hussaini.....	78
2.4	Motivation for the Current Study.....	84
3.0	THE GOVERNING EQUATIONS SOLVED BY <i>EE_SOLVER</i> – MOTIVATION AND ADVANTAGES	86
3.1	The Buffer Zone Technique Applied to the One Dimensional Euler Equations	87
3.2	The Runge-Kutta Time-Integration Scheme Employed in <i>EE_Solver</i>	90
3.3	Numerical Stability Analysis for the <i>EE_Solver</i> Runge-Kutta Scheme	95
3.4	Appropriate Values for The Buffer Zone Parameters U' and μ	101
4.0	SIMULATION RESULTS AND DISCUSSION.....	106
4.1	Reflective Boundary Difficulties in Navier-Stokes Simulations.....	106
4.2	Boundary Reflection Studies With <i>EE_Solver</i>	113
4.3	Test Case 1 Simulation Results With an Outflow Buffer Zone	127
4.3.1	Definition of U' and its Connection With β and μ	127
4.3.2	Optimizing U'_{Npts} , BW, and Buffer Zone Node Count.....	129
4.3.3	Test Case 1 Simulations Using Optimal Buffer Zone Parameters.....	134
4.4	Steady-State Simulations.....	137
4.5	The Buffer Zone Modification With the Euler Equations in Characteristic Form.....	139

5.0	IMPLEMENTATION OF THE BUFFER ZONE TECHNIQUE INTO FULL NAVIER-STOKES ALGORITHMS.....	143
5.1	Numerically Solving the Governing Equations of Fluid Dynamics	144
5.2	Pseudo-Time Stepping and Preconditioning (Artificial Compressibility) ..	147
5.2.1	Artificial Compressibility	149
5.3	The Buffer Zone Method in the Context of Artificial Compressibility	153
5.3.1	Artificial Compressibility in the Steady state Limit	157
5.4	Generalizing the Definitions of U' and μ for Arbitrary Three-Dimensional Domains	159
6.0	CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE STUDIES	169
6.1	Future Research	171
	ACKNOWLEDGMENTS	173
	REFERENCES	175
	APPENDIX A – THE DERIVATION OF THE GOVERNING EQUATIONS OF FLUID DYNAMICS	185
A.1	Conservation of Mass	185
A.2	Conservation of Momentum	187
A.3	Conservation of Energy	188
	APPENDIX B – COMMON CFD TECHNIQUES AND THE <i>FS</i> ALGORITHM	191
B.1	Finite-Difference Method	193
B.2	Finite-Element Method	194
B.3	Finite-Volume Method	196

B.4	Spatial Grids.....	197
B.4.1	Structured Grids	198
B.4.2	Unstructured Grids	199
B.4.3	Multi Block Structured Grids.....	200
B.4.4	Control Volume	202
B.4.4.1	Cell-Centered Control Volume Scheme.....	202
B.4.4.2	Vertex-Centered Control Volume Scheme	205
B.4.4.3	<i>FS</i> Control Volume	207
B.4.5	Numerical Discretization	207
B.4.5.1	Pseudo-Time Stepping and Preconditioning (Artificial Compressibility)	210
B.4.5.1.1	Artificial Compressibility	211
B.4.6	Relaxation Scheme	215
B.4.7	ADI vs. MSIP	218
B.4.8	The Multigrid Method	222
APPENDIX C –MATLAB CODE FOR THE <i>EE_SOLVER ALGORITHM</i>		229
C.1	<i>Char_Solver</i> Nomenclature	230
	<i>Char_Solver</i> Matlab Code.....	233
APPENDIX D – <i>FS</i> SIMULATION RESULTS, OPTIMAL PARAMETER SETTINGS AND COMPUTATIONAL DOMAINS.....		245

LIST OF FIGURES

Figure 1.3.1 – The importance of inertial effects on the deformation of a compound drop: (a) Non-creeping flow with $Re = 100$ (b) Creeping flow with $Re \ll 1$	10
Figure 1.3.2 – An example of boundary layer formation due to the presence of viscous forces at a solid boundary	11
Figure 1.5.1.2 – Stability regions for the Runge-Kutta schemes with $s = 1, \dots, 4$ stages, reproduced from Trefethen (1994)	29
Figure 1.6.1 – Portion of the x - t plane in which the solution to Eqn. (1.6.1.a) is determined by the initial data $u(x,0)$, $0 \leq x \leq L$, reproduced from Durran, 1999	33
Figure 1.6.2 – Two arbitrary characteristic lines for the initial boundary value problem given by Eqns. (1.6.1.a) – (1.6.1.d)	33
Figure 2.2.3.1 – Computational domain and corresponding boundary conditions for Rudy <i>et al.</i> 's test case 1, reproduced from Rudy <i>et al.</i> (1980).....	63
Figure 2.2.3.2 – Effect of α on the convergence rate for Rudy <i>et al.</i> 's test case 1, reproduced from Rudy <i>et al.</i> (1980)	64
Figure 2.2.3.3 – Effect of α on the convergence rate for Rudy <i>et al.</i> 's test case 2, reproduced from Rudy <i>et al.</i> (1980)	65
Figure 2.3.1.1 – A schematic of the buffer zone computational domain used in the work of Ta'asan and Nark (1995)	74
Figure 2.3.1.2 – An illustration of outflow boundary reflection as a function of buffer zone size, reproduced from Ta'asan and Nark (1995).....	77

Figure 2.3.1.3 – An illustration of outflow boundary reflection as a function of (Buffer Zone Size)/(Gaussian Width), reproduced from Ta’asan and Nark (1995)	78
Figure 2.3.2.1 – A graphical summary of the PML method, reproduced from Hayder <i>et al.</i> (1997)	80
Figure 2.3.2.2 – The percentage error in pressure fluctuations for varying node counts within the absorbing layer for numerical simulations of a free-shear layer, reproduced from Hayder <i>et al.</i> (1997)	82
Figure 2.3.2.3 – The percentage error in pressure fluctuations for varying node counts within the absorbing layer for numerical simulations of a shock- vorticity wave interaction problem, reproduced from Hayder <i>et al.</i> (1997)	83
Figure 3.3.1 – Stability region for an explicit two-stage Runge-Kutta scheme	97
Figure 3.3.2 – Stability region defined by Eqns. (3.3.8) and (3.3.9)	99
Figure 3.3.3 – A hypothetical stability region for the discretization of Eqn. (3.3.7) using a two-stage Runge-Kutta time integration scheme	101
Figure 4.1.1 – X-momentum equation residual profile for a two-dimensional laminar flow simulation, $Re_L = 40000$, $M_\infty = 0.03$	108
Figure 4.1.2 – The maximum absolute change in pressure during the abover laminar flow simulation	109
Figure 4.1.3 – Y-momentum equation residual profile for a two-dimensional steady pipe flow simulation, $Re_L = 40000$, $M_\infty = 0.03$	109
Figure 4.1.4 – X-momentum equation residual profile for a three-dimensional steady rectangular duct flow simulation, $Re_L = 40000$, $M_\infty = 0.03$.	111

Figure 4.1.5 – The maximum absolute change in pressure during the above rectangular duct flow simulation.	111
Figure 4.1.6 – X-momentum equation residual profile for a three-dimensional steady pipe flow simulation, $Re_L = 40000$, $M_\infty = 0.03$	112
Figure 4.1.7 – X-momentum equation artificial diffusion profile (at steady state) for a three-dimensional steady pipe flow simulation	112
Figure 4.2.1 – Summary of initial and boundary conditions for Test Case 1	114
Figure 4.2.2 – Test Case 1 computational domain. A finite thickness in the y direction is shown for aesthetic purposes	116
Figure 4.2.3 – p_{error} profile for a Test Case 1 <i>NonCharSolver</i> simulation showing pressure probes at various iteration counts – fixed density outflow boundary condition.....	119
Figure 4.2.4 – The first eight pressure probes from a <i>NonCharSolver</i> Test Case 1 simulation - fixed density outflow boundary condition	120
Figure 4.2.5 – The four pressure probes ⑨ – ⑫ taken from Fig. 4.2.4 for a <i>NonCharSolver</i> Test Case 1 simulation - fixed density boundary condition	122
Figure 4.2.6 – Normalized pressure distribution at steady state for a <i>NonCharSolver</i> simulation with fixed density outflow boundary condition.....	123
Figure 4.2.7 – X-momentum artificial diffusion distribution at steady state for a <i>NonCharSolver</i> simulation with fixed density outflow boundary condition	123
Figure 4.2.8 – The first eight pressure probes from a <i>NonCharSolver</i> Test Case 1 simulation - fixed density boundary condition	125

Figure 4.2.8 – Graphical representation of the buffer zone modification used in Test Case 1 simulations.....	126
Figure 4.3.2.1 – <i>NonCharSolver</i> Test Case 1 simulations for various U'_{Npts} values, with BW = 0.3 and 5 nodes in the buffer zone	131
Figure 4.3.2.2 – <i>NonCharSolver</i> Test Case 1 simulations for BW values of 0.1 and 0.2 and buffer zone node counts ranging from 2 to 5, with $U'_{Npts} = 10$.	132
Figure 4.3.3.1 – Test Case 1 p_{error} trends for two <i>NonCharSolver</i> simulations conducted with no buffer zone and optimal buffer zone, respectively	135
Figure 4.4.1 – Four <i>NonCharSolver</i> simulations indicating the necessity for a new optimal value of U'_{Npts} when Δt_{max} is used with various value of μ_{Npts}	138
Figure 4.5.1 – Three <i>CharSolver</i> simulations modifying different combinations of incoming and outgoing Riemann invariants	141
Figure 5.2.1.1 – Convergence sensitivity of a laminar boundary-layer simulation to artificial compressibility. Here $\beta = a_1 = a_2 = a_3 = a_4 = a_5$, reproduced from Bet <i>et al.</i> (1995).....	152
Figure 5.4.1 – A schematic of the buffer zone computational domain used in the work of Ta'asan and Nark (1995)	159
Figure 5.4.2 – Theoretical buffer zone additions to an arbitrary computational domain, $x_{min} \leq x \leq x_{max}, y_{min} \leq y \leq y_{max}$	161

Figure 5.4.3 – An arbitrary computational domain where, for some non-specified problem, the fluid velocity at the point (x_e, y_e) lying on the interface of the computational outflow boundary and the buffer zone has both an x and y component.....	162
Figure 5.4.4 – A schematic of a coordinate transformation from a physical domain (x, y) to a computational domain (ξ, η)	164
Figure A.1.1 – Arbitrary flow region R	185
Figure B.2.1 – Introduction to finite element methods in the context of the “ π problem” (a) continuum object, (b) a discrete approximation, (c) disconnected element, (d) generic element, reproduced from Young and MacPhedran (2002)	195
Figure B.4.1 – Example of (a) a structured grid and (b) an unstructured grid.....	198
Figure B.4.1.1 – Structured grid surrounding the front portion of a NASA/Langley space shuttle	199
Figure B.4.2.1 – Unstructured grid surrounding the front portion of a NASA/Langley space shuttle	200
Figure B.4.3.1 – Different grid distribution approaches, reproduced from Lin <i>et al.</i> (1991).....	201
Figure B.4.4.1.1 – Cell-centered control volume (two-dimensional)	203
Figure B.4.4.1.2 – A typical single grid cell.....	204
Figure B.4.4.2.1 – Vertex-centered control volume (two-dimensional)	206
Figure B.4.5.1.1.1 – Convergence sensitivity of a laminar boundary-layer simulation to artificial compressibility. Here $\beta = a_1 = a_2 = a_3 = a_4 = a_5$, reproduced from Bet <i>et al.</i> (1995)	214

Figure B.4.8.1 – Scaling coefficients for two-dimensional full-weighting	
restriction	224
Figure B.4.8.2 – Scaling coefficients for three-dimensional full-weighting	
restriction	225
Figure B.4.8.3 – An example of a V-cycle multigrid process with five grid levels..	227
Figure B.4.8.4 – Various multigrid strategies, reproduced from Bittencourt <i>et al.</i>	
(2000).....	228
Figure D.1 – Two-dimensional parallel plate computational domain used in <i>FS</i>	
simulations	245
Figure D.2 – Two-dimensional axisymmetric pipe computational domain used in <i>FS</i>	
simulations	246
Figure D.3 – Three-dimensional rectangular duct computational domain used in <i>FS</i>	
simulations	247
Figure D.4 – Three-dimensional (quarter)-pipe computational domain used in <i>FS</i>	
simulations	248
Figure D.1.1 – Experimental results for $c (= c_a) = 0.5$ and 1.0 (Rectangular Duct)	
(a) Center-line pressure distribution (b) Velocity distribution	
(c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals	249
Figure D.1.2 – Experimental results for $G = 1$ and 2 (Rectangular Duct)	
(a) Center-line pressure distribution (b) Velocity distribution	
(c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals	250
Figure D.1.3 – Experimental results for $G = 2$ and 3 (Rectangular Duct)	
(a) Center-line pressure distribution (b) Velocity distribution	
(c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals	251

Figure D.1.4 – Experimental results for RK = 2 and 4 (Rectangular Duct)

- (a) Center-line pressure distribution (b) Velocity distribution
- (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals 252

Figure D.1.5 – Experimental results comparing MSIP, ADI and ADI (no multigrid in i direction) (Rectangular Duct)

- (a) Center-line pressure distribution (b) Velocity distribution
- (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals 253

Figure D.1.6 – Experimental results comparing Interpolation Option 1 and 2 (Rectangular Duct)

- (a) Center-line pressure distribution (b) Velocity distribution
- (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals ... 254

Figure D.1.7 – Experimental results comparing the exchange of various data at periodic boundaries (Rectangular Duct)

- (a) Center-line pressure distribution (b) Velocity distribution
- (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals ... 255

Figure D.1.8 – Experimental results comparing $\kappa_4 = 0.005$ and $\kappa_4 = 0.01$ (3D Pipe)

- (a) Center-line pressure distribution (b) Velocity distribution
- (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals ... 256

Figure D.1.9 – Experimental results comparing values of $c = c_a$ (2-D Pipe)

- (a) Center-line pressure distribution (b) Velocity distribution
- (c) – (e) Continuity, X-, and Y-momentum equation residuals 257

LIST OF TABLES

Table 1.5.1.1 – The minimum number of Runge-Kutta stages needed to obtain specific order of accuracy, reproduced from Trefethen (1994)	30
Table 1.6.1 – Boundary condition specification and extrapolation required at subsonic and supersonic boundaries	36
Table 2.2.1.1 – Reduction in wave reflection due to two highly absorbing boundary conditions for the linearized shallow water equations, reproduced from Engquist <i>et al.</i> (1977)	52
Table 2.2.2.1 – The amount of error due to pressure wave reflection as a function of increasing Mach number, reproduced from Thompson (1986)	59
Table 4.3.2.2 – CPU comparisons for Test Case 1 optimization studies	134
Table 4.3.3.1 – Summary of the gains provided by the optimal implementation of the buffer zone technique	135
Table B.2.1 – An approximation for π using $\pi_n = n \sin(\frac{\pi}{n})$, reproduced from Roger and MacPhedran (2002).....	195
Table D.1 – Optimal paramter settings for the two-dimensional parallel plate simulations.....	245
Table D.2 – Optimal paramter settings for the two-dimensional axisymmetric pipe flow simulations.....	246
Table D.3 – Optimal paramter settings for the three-dimensional rectangular duct flow simulations.....	247
Table D.4 – Optimal paramter settings for the three-dimensional (quarter)-pipe flow simulations.....	248

NOMENCLATURE

Abbreviations

BW	Buffer width
<i>CharSolver</i>	Component of the author's numerical algorithm that solves the non-linear inviscid Euler equations (1-D) in characteristic form
CFD	Computational Fluid Dynamics
CFL	Courant- Friedrichs-Lewy number, used in calculating the inviscid (convective) time-step limit
<i>EE_Solver</i>	Two component numerical algorithm (<i>CharSolver</i> and <i>NonCharSolver</i>) written by the author, which solves the non-linear inviscid Euler equations (1-D) in both characteristic and non-characteristic form
<i>FS</i>	Flow Solver (a full Navier-Stokes solver developed by Dr. M.I. Yaras)
<i>NonCharSolver</i>	Component of the author's numerical algorithm that solves the non-linear inviscid Euler equations (1-D) in non-characteristic form
NRBC	Non-reflecting boundary condition
NRBZ	Non-reflecting buffer zone
PDE	Partial differential equation
RK	Runge-Kutta

English Symbols

a	Speed of sound, constant (real or complex) used in von Neumann analysis
$a_1 - a_5$	Scaling coefficients for the artificial compressibility method used in <i>FS</i>
$A_{c.v.}$	Surface area of control volume (m^2)
C^+, C, C^-	Three characteristic lines for subsonic flow upon which Riemann invariants propagate
c	Wave speed ($\frac{m}{s}$)
D	Diagonal matrix of eigenvalues for Euler equations
D_a	Artificial dissipation term (Chapter 3)
$d_{i+\frac{1}{2}}$	Component of artificial dissipation term (Chapter 3) – other terms are analogous
E	Flux vector in the x -direction
E_v	Viscous stress vector in the x -direction
E	Component of error term in Ta'asan and Nark's (1995) buffer zone work, error term in Hayder, Hu, and Hussaini's (1997) work
E_0	Component of error term in Ta'asan and Nark's (1995) buffer zone work
e_t	Total energy ($\frac{m^2}{s^2}$)
e	Internal energy due to molecular motion ($\frac{m^2}{s^2}$)
F	Flux vector in the y -direction
F_B	Body forces in the derivation of the energy equation

$\mathbf{F}_v$	Viscous stress vector in the y -direction
$\mathbf{G}$	Flux vector in the z -direction
$\mathbf{G}_v$	Viscous stress vector in the z -direction
$\mathbf{g}$	Gravitational acceleration vector ($\frac{m}{s^2}$)
$\mathbf{H}$	Source term vector
K_c, K_d	Scaling coefficients that dictate the inviscid (convective) and viscous (diffusive) limits of the artificial time-step, respectively, in FS
k_B	Bulk viscosity
k_m	Wave number used in von Neumann analysis
L	Length scale
$\mathbf{n}$	Surface-normal vector
$\mathbf{p}$	Boundary stresses in the derivation of the energy equation
p	Fluid static pressure (Pa)
p_{error}	Used a measure of the error introduced by reflection of outgoing Riemann invariants in the current work (Eqn. (4.2.1))
$\mathbf{Q}$	Vector of dependent variables (solution vector)
$\mathbf{q}$	Heat flux vector
q	Heat flux vector component (one for each direction)
$\mathbf{R}$	Jacobian matrix, residual vector
$\mathcal{R}$	Arbitrary three-dimensional region used in the derivation of the Navier- Stokes equations
$\mathbb{R}$	The set of real numbers
R	Universal gas constant
Re	Reynolds number

T	Fluid static temperature ($^{\circ}$ K)
t	Time (s)
U', U'	Parameter used in Ta'asan and Nark's (1995) buffer zone technique
U	Velocity scale
$\mathbf{u}$	Velocity vector
u	Velocity component in the x -direction ($\frac{m}{s}$)
$V_{c.v.}$	Fixed control volume of three-dimensional region R (m^3)
v	Velocity component in the y -direction ($\frac{m}{s}$)
w	Velocity component in the z -direction ($\frac{m}{s}$)
$\mathbf{x}$	Position vector (x, y, z)

Greek Symbols

α	Constant used in Rudy and Strikwerda's (1980) boundary condition
α_1, α_2	Constants used in Runge-Kutta integration
β	Parameter used in Ta'asan and Nark's (1995) buffer zone technique
Δ	Used to denote a change in a variable
$\varepsilon_2, \varepsilon_4$	Coefficients used in the calculation of the artificial diffusion terms of <i>EE_Solver</i> (Chapter 3)
γ	Specific heat ratio
κ_2, κ_4	Scaling coefficients used in artificial diffusion terms (Chapter 3)
λ	Constant component of bulk viscosity, often assumed to be equal to $-\frac{2}{3}\mu$, general symbol used to denote an eigenvalue

μ	Kinematic viscosity, parameter used in Ta'asan and Nark's (1995) buffer zone technique
η	Parameter used in Ta'asan and Nark's (1995) buffer zone technique
ρ	Density of fluid
τ	Reynolds stress tensor

Subscripts

0	Fixed point (i.e. x_0)
<i>c.v.</i>	Control volume
<i>e</i>	Indicates the interface between the outflow boundary and buffer zone in Ta'asan and Nark's (1995) buffer zone technique
<i>i, j, k</i>	Discrete grid indices
<i>in</i>	Flow variable value at the inflow boundary
<i>m</i>	Used to denote frequency in von Neumann analysis
<i>out</i>	Flow variable value at the outflow boundary
<i>t</i>	Time derivative, the total amount of a particular quantity (such as energy)
<i>x, y, z</i>	<i>x</i> , <i>y</i> , and <i>z</i> components
∞	Used to denote free-stream values of flow variables

Superscripts

*	Dimensionless quantities, intermediate quantities for RK integration
', "	Notations used to denote primitive variable forms
<i>n</i>	Time-step <i>n</i>

1.0 INTRODUCTION

1.1 OVERVIEW AND A BRIEF HISTORY OF COMPUTATIONAL FLUID DYNAMICS

At its very core, computational fluid dynamics (CFD) uses high-speed computers and numerical methods to seek approximate solutions to the mathematical equations governing fluid flow. For most practical applications, this implies the numerical solution of the Navier-Stokes equations, which are the equations describing the motion of a fluid. CFD continues to grow rapidly, due in large part to advancements in computing power. Well-established methods are finding broader applications, while research has opened the door to a vast array of alternatives.

The field of fluid dynamics stands as one of the oldest and broadest within engineering. In this intellectual sphere the Navier-Stokes equations certainly receive the greatest scrutiny, both from engineers and mathematicians alike. For a historical perspective into this vast field, the interested reader can find numerous sources, such as that of Darrigol (2002). According to the database Web of Science (Science Citation Index), there is an average of 15 – 20 published papers per week dedicated to this subject, a testament to its arduous nature. Moreover, scientists from numerous areas of expertise have made, at times, very specific contributions. In this fashion, the field has enjoyed its growth in a true spirit of camaraderie.

Consequently, within most numerical algorithms designed for fluid flow simulations, it is not unusual to find a collection of techniques from various sources, interwoven to produce desired results. Robustness then necessarily becomes an issue: that is, how applicable is a particular algorithm for dealing with a wide variety of fluid flow

regimes? With this in mind, the current study, in its scope and aim, is a cultivation of existing work and is motivated by a desire to improve the current state of outflow boundary treatment. To facilitate this process, a full Navier-Stokes solver (Flow Solver – *FS*) developed by one of the thesis advisors of the author is used to examine various flows. The effectiveness of this algorithm is well documented (e.g. Yaras (2002) and Yaras and Grosvenor (2002)). However, to make possible the detailed study of boundary treatments in a less elaborate setting, the author has written and conducted simulations with a two-component numerical algorithm designed to solve the non-linear Euler equations in one dimension. The algorithm, *EE_Solver*, solves the governing equations in both their characteristic form (first component – *CharSolver*) and their non-characteristic form (second component – *NonCharSolver*), and implements many of the techniques employed in *FS*. Chapter 3 presents a full description of the *EE_Solver* algorithm. All numerical simulations conducted with *EE_Solver* were carried out on a Compaq™ Presario with Intel® Celeron™ processor and 124 MB of RAM. For the interested reader, a detailed description of the techniques used within the *FS* algorithm can be found in Appendix B, while the *EE_Solver* Matlab code is given in Appendix C.

In order to guide the reader through this work, the following objectives are identified:

1. A presentation of the full three-dimensional Navier-Stokes equations.
2. An overview of the problematic nature of outflow boundary conditions within numerical algorithms designed to solve these equations.
3. A detailed examination of several techniques addressing this problem, including the buffer zone method of Ta'asan and Nark (1995) and Hedstrom's

(1979) non-reflecting boundary condition, as well as their influence on other, similar approaches.

4. An application of Ta'asan and Nark's method to the non-linear Euler equations in both non-characteristic and characteristic form.
5. A detailed optimization study of the more pertinent buffer zone parameters, accompanied by a novel treatment of the characteristic form of the Euler equations. Both of these objectives are presented for the first time within this work.
6. The extension of the buffer zone technique to full Navier-Stokes algorithms, also explored for the first time here.

Before engaging in further discussions regarding CFD, we first acquaint ourselves with the Navier-Stokes equations. The next section sets out to accomplish this task in a concise manner. A more thorough derivation of these equations can be found in Appendix A.

1.2 THE NAVIER-STOKES EQUATIONS

Applying the principles of conservation of mass, momentum, and energy to an arbitrary three-dimensional region R of fixed volume $V_{c.v.}$ within a compressible flow, having surface boundary $A_{c.v.}$ and outward normal $\mathbf{n}$, yields the following set known as the Navier-Stokes equations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad (1.2.1)$$

$$\rho \left[\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right] + \mathbf{u} \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right] = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{g} \quad (1.2.2)$$

$$\frac{\partial(\rho e_t)}{\partial t} + \nabla \cdot (\rho e_t \mathbf{u}) = \nabla \cdot (\kappa \nabla T) + \rho q - \nabla \cdot (p \mathbf{u}) + \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{u}) + (\rho \mathbf{g}) \cdot \mathbf{u} \quad (1.2.3)$$

where $\mathbf{u} = (u, v, w)$ is the velocity vector, ρ the density, p the static pressure, $\mathbf{g}$ the gravitational acceleration vector, $\boldsymbol{\tau}$ the shear stress tensor, q the internal heat source, T the total temperature, κ the thermal conductivity, and e_t the total energy. The continuity equation (1.2.1) follows from the principle that the accumulation of mass inside the control volume is equal to the net influx through the surface. In a similar fashion, the equation describing conservation of momentum (Eqn. (1.2.2)) results from the application of Newton's second law, while the derivation of Eqn. (1.2.3) for energy conservation has the first law of thermodynamics as its guiding physical principle. As mentioned earlier, Appendix A presents a detailed derivation of these governing equations.

Furthermore, for a Newtonian fluid the shear stresses are given by:

$$\tau_{xx} = 2\mu \frac{\partial u}{\partial x} + \lambda \nabla \cdot \mathbf{u} \quad \tau_{xy} = \tau_{yx} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \quad (1.2.4)$$

$$\tau_{yy} = 2\mu \frac{\partial v}{\partial y} + \lambda \nabla \cdot \mathbf{u} \quad \tau_{xz} = \tau_{zx} = \mu \left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) \quad (1.2.5)$$

$$\tau_{zz} = 2\mu \frac{\partial w}{\partial z} + \lambda \nabla \cdot \mathbf{u} \quad \tau_{yz} = \tau_{zy} = \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \quad (1.2.6)$$

The constant $k_B = \lambda + \frac{2}{3}\mu$ is referred to as the bulk viscosity of the fluid. This

parameter is often assumed negligible, implying that $\lambda = -\frac{2}{3}\mu$.

Since the shear viscosity, μ , and thermal conductivity, κ , are functions of T , Eqns. (1.2.1) – (1.2.3) represent a system of five equations in six unknowns (u, v, w, ρ, p , and T), and is thus underdetermined. For this reason, an equation of state relating two of the above variables is often incorporated as the necessary sixth equation. For example, when dealing with a perfect gas, one can use the following equation of state:

$$p = \rho RT \quad (1.2.7)$$

where R is the universal gas constant.

Alternatively, one may write the compact flux-vector form of the Navier-Stokes equations in Cartesian coordinates (without body forces or external heat addition) in the following manner:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} = \frac{\partial \mathbf{E}_v}{\partial x} + \frac{\partial \mathbf{F}_v}{\partial y} + \frac{\partial \mathbf{G}_v}{\partial z} \quad (1.2.8)$$

where,

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e_t \end{bmatrix} \quad (1.2.9)$$

$$\mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ (\rho e_t + p)u \end{bmatrix} \quad \mathbf{E}_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \end{bmatrix} \quad (1.2.10)$$

$$\mathbf{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ (\rho e_t + p)v \end{bmatrix} \quad \mathbf{F}_v = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} - q_y \end{bmatrix} \quad (1.2.11)$$

$$\mathbf{G} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ (\rho e_t + p)w \end{bmatrix} \quad \mathbf{G}_v = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} - q_z \end{bmatrix} \quad (1.2.12)$$

In seeking solutions to the Navier-Stokes equations by means of numerical methods, many researchers prefer the compact vector form, as it facilitates both the application of a discretization scheme and its subsequent coding into a numerical algorithm.

1.3 DIMENSIONAL ANALYSIS AND THE SIMPLIFICATION OF THE GOVERNING EQUATIONS OF FLUID DYNAMICS

Although the Navier-Stokes equations encompass all possible fluid flow regimes, for many problems of interest the use of all terms in these equations is unnecessary. At times, the contributions of specific terms in Eqns. (1.2.1) – (1.2.3) are found to be magnitudes of order greater than certain other terms. Such occurrences in particular types of fluid dynamics problems allow for the simplification of the Navier-Stokes equations. In order to expose which terms, if any, may be neglected, the governing equations are put into their non-dimensional form, thus forcing each variable to have a certain prescribed range of values (often between 0 and 1), while at the same time introducing fixed reference quantities as multiplicative factors. Furthermore, the non-dimensional form of the equations facilitates independent studies of characteristic

parameters such as the Mach number, Reynolds number, and Prandtl number (which arise as combinations of the reference quantities), as well as reducing round-off error in numerical simulations. One common non-dimensionalizing procedure uses the following definitions:

$$\begin{aligned}
 x^* &= \frac{x}{L} & y^* &= \frac{y}{L} & z^* &= \frac{z}{L} & t^* &= \frac{t}{L/V_\infty} \\
 u^* &= \frac{u}{V_\infty} & v^* &= \frac{v}{V_\infty} & w^* &= \frac{w}{V_\infty} & \mu^* &= \frac{\mu}{\mu_\infty} \\
 \rho^* &= \frac{\rho}{\rho_\infty} & p^* &= \frac{p}{p_\infty V_\infty^2} & T^* &= \frac{T}{T_\infty} & e^* &= \frac{e}{V_\infty^2}
 \end{aligned} \tag{1.3.1}$$

where the non-dimensional variables are denoted by an asterisk, freestream conditions are indicated by ∞ , and L is the reference length used in the Reynolds number:

$$\text{Re}_L = \frac{\rho_\infty V_\infty L}{\mu_\infty} \tag{1.3.2}$$

When this choice of non-dimensionalization is applied to the Navier-Stokes equations shown in Eqns. (1.2.8) – (1.2.12), the following non-dimensional forms are obtained:

$$\frac{\partial \mathbf{Q}^*}{\partial t^*} + \frac{\partial \mathbf{E}^*}{\partial x^*} + \frac{\partial \mathbf{F}^*}{\partial y^*} + \frac{\partial \mathbf{G}^*}{\partial z^*} = \frac{\partial \mathbf{E}_v^*}{\partial x^*} + \frac{\partial \mathbf{F}_v^*}{\partial y^*} + \frac{\partial \mathbf{G}_v^*}{\partial z^*} \tag{1.3.3}$$

with,

$$\mathbf{Q} = \begin{bmatrix} \rho^* \\ \rho^* u^* \\ \rho^* v^* \\ \rho^* w^* \\ \rho^* e_t^* \end{bmatrix} \tag{1.3.4}$$

$$\mathbf{E} = \begin{bmatrix} \rho^* u^* \\ \rho^* u^{*2} + p^* \\ \rho^* u^* v^* \\ \rho^* u^* w^* \\ (\rho^* e_i^* + p^*) u^* \end{bmatrix} \quad \mathbf{E}_v = \begin{bmatrix} 0 \\ \tau_{xx}^* \\ \tau_{xy}^* \\ \tau_{xz}^* \\ u^* \tau_{xx}^* + v^* \tau_{xy}^* + w^* \tau_{xz}^* - q_x^* \end{bmatrix} \quad (1.3.5)$$

$$\mathbf{F} = \begin{bmatrix} \rho^* v^* \\ \rho^* u^* v^* \\ \rho^* v^{*2} + p^* \\ \rho^* v^* w^* \\ (\rho^* e_i^* + p^*) v^* \end{bmatrix} \quad \mathbf{F}_v = \begin{bmatrix} 0 \\ \tau_{yx}^* \\ \tau_{yy}^* \\ \tau_{yz}^* \\ u^* \tau_{yx}^* + v^* \tau_{yy}^* + w^* \tau_{yz}^* - q_y^* \end{bmatrix} \quad (1.3.6)$$

$$\mathbf{G} = \begin{bmatrix} \rho^* w^* \\ \rho^* u^* w^* \\ \rho^* v^* w^* \\ \rho^* w^{*2} + p^* \\ (\rho^* e_i^* + p^*) w^* \end{bmatrix} \quad \mathbf{G}_v = \begin{bmatrix} 0 \\ \tau_{zx}^* \\ \tau_{zy}^* \\ \tau_{zz}^* \\ u^* \tau_{zx}^* + v^* \tau_{zy}^* + w^* \tau_{zz}^* - q_z^* \end{bmatrix} \quad (1.3.7)$$

and

$$e_i^* = e^* + \frac{u^{*2} + v^{*2} + w^{*2}}{2} \quad (1.3.8)$$

e^* is the non-dimensional form of the molecular internal energy, e . The components of the shear stress tensor and the internal heat source in non-dimensional form are given by:

$$\begin{aligned} \tau_{xx}^* &= \frac{2\mu^*}{3\text{Re}_L} \left(2 \frac{\partial u^*}{\partial x^*} - \frac{\partial v^*}{\partial y^*} - \frac{\partial w^*}{\partial z^*} \right) & \tau_{xy}^* = \tau_{yx}^* &= \frac{\mu^*}{\text{Re}_L} \left(\frac{\partial u^*}{\partial y^*} + \frac{\partial v^*}{\partial x^*} \right) \\ \tau_{yy}^* &= \frac{2\mu^*}{3\text{Re}_L} \left(2 \frac{\partial v^*}{\partial y^*} - \frac{\partial u^*}{\partial x^*} - \frac{\partial w^*}{\partial z^*} \right) & \tau_{xz}^* = \tau_{zx}^* &= \frac{\mu^*}{\text{Re}_L} \left(\frac{\partial w^*}{\partial x^*} + \frac{\partial u^*}{\partial z^*} \right) \end{aligned}$$

$$\begin{aligned}
\tau_{zz}^* &= \frac{2\mu^*}{3\text{Re}_L} \left(2 \frac{\partial w^*}{\partial z^*} - \frac{\partial u^*}{\partial x^*} - \frac{\partial v^*}{\partial y^*} \right) & \tau_{yz}^* = \tau_{zy}^* &= \frac{\mu^*}{\text{Re}_L} \left(\frac{\partial v^*}{\partial z^*} + \frac{\partial w^*}{\partial y^*} \right) \\
q_x^* &= -\frac{\mu^*}{(\gamma-1)M_\infty^2 \text{Re}_L \text{Pr}} \frac{\partial T^*}{\partial x^*} & q_y^* &= -\frac{\mu^*}{(\gamma-1)M_\infty^2 \text{Re}_L \text{Pr}} \frac{\partial T^*}{\partial y^*} \\
q_z^* &= -\frac{\mu^*}{(\gamma-1)M_\infty^2 \text{Re}_L \text{Pr}} \frac{\partial T^*}{\partial z^*}
\end{aligned} \tag{1.3.9}$$

where M_∞ is the freestream Mach number and Pr is the Prandtl number, given by:

$$M_\infty = \frac{V_\infty}{\sqrt{\gamma R T_\infty}}, \quad \text{Pr} = \frac{\mu c_p}{\kappa} \tag{1.3.10}$$

Here c_p is the specific heat at a constant pressure.

With this non-dimensionalization in place, we may then proceed to compare the values of Re , M_∞ , and Pr with various other terms in the Navier-Stokes equations.

Two of the most common simplifications occur when $\text{Re} \ll 1$ and $\text{Re} \gg 1$, commonly referred to as creeping flows and boundary-layer flows, respectively.

Creeping flow is a term used to describe a fluid that has effectively no inertia. In such

flows, the acceleration ($\frac{D\mathbf{u}}{Dt}$, where $\frac{D}{Dt}$ is the material derivative) is so small

compared to the viscous force per unit mass, that we may neglect it in the Navier-

Stokes equations. Fig. 1.3.1 shows the effect of inertia on a drop of a particular compound having two different Reynolds numbers.

Figure 1.3.1 – The importance of inertial effects on the deformation of a compound drop: (a) Non-creeping flow with $Re = 100$
(b) Creeping flow with $Re \ll 1$

Unlike the case of creeping flows where the large viscous contribution allows for the neglecting of the acceleration terms, higher Reynolds number flows must be analyzed slightly differently. In particular, we note that the stress tensors shown in Eqn. (1.3.9) appear in the derivatives of the right-hand side of Eqn. (1.3.3). Thus, prior to discarding any of the stress terms of Eqn. (1.3.9), we require not only that the Reynolds number be high, but that the magnitudes of the u , v , and w second derivatives be small. Consequently, any velocity components having a second derivative on the same order of magnitude as the Reynolds number cannot be neglected. Such is the case, say, of a two-dimensional fluid encountering friction (viscosity) at a solid wall. The decrease in momentum of the fluid layer nearest the wall is transferred to adjacent layers through fluid particle interaction, as shown in Fig. 1.3.2. The second derivative of any velocity component represents the amount curvature present in the velocity profile. Thus, for the example shown in Fig. 1.3.2, stress tensor terms involving $\frac{\partial u}{\partial y}$ cannot be neglected.

Figure 1.3.2 – An example of boundary layer formation due to the presence of viscous forces at a solid boundary

Conversely, when Re_L is high and the velocity curvature (i.e. the second derivatives of velocity) is low, the stress terms (τ) can be neglected, leaving the set of equations traditionally known as the Euler equations. More will be said shortly on the Euler equations; however, for a more detailed look at simplifications stemming from the non-dimensionalization of the Navier-Stokes equations, the interested reader may consult the textbook of White (1991).

This last assumption, commonly called the inviscid assumption, accounts for many of the problems studied in fluid dynamics. In the case of real fluids it is almost never true that we can completely neglect viscosity. Yet, for $Re > 1000$, many conclude that it is justifiable.

Simplification of the Navier-Stokes equations allows researchers to focus on particular aspects of CFD algorithms that would otherwise be difficult to observe in

the complex setting of the full equations. What is more, suggested improvements based on the results of simplified studies often prove to be beneficial.

In point of fact, the current work focuses on the inviscid case embodied by the non-linear Euler equations in one dimension. From a mathematical point of view the Euler equations are a set of coupled, hyperbolic differential equations whose use has several advantages. For example:

1. They allow for discontinuous solutions, hence applicable to flows with shocks or contact discontinuities.
2. One can determine characteristic quantities and associated characteristic speeds that are directly related to the physical behaviour of the fluid. In the one-dimensional case this form is well understood and easily accessible.

In addition, the non-linear Euler equations in one dimension represent a wide variety of flows. This, coupled with the fact that the parabolic nature of the full Navier-Stokes equations does not provide as much flexibility in terms of mathematical manipulation and physical understanding, makes the use of these equations advantageous. Yet, the foregoing reasons are presented on the basis of general mathematical and engineering principles, and make no reference to the main focus of this research. For this reason, throughout the next several chapters, as details regarding the current work become known, additional justification will be given to the use of these equations. In the meantime, we continue along a more fundamental path by considering the characteristic form of this one-dimensional set of equations in a more meticulous manner.

1.4 DERIVATION OF THE CHARACTERISTIC FORM OF THE NON-LINEAR EULER EQUATIONS IN ONE DIMENSION

For matters of brevity, from this point on we shall make reference simply to the one-dimensional Euler equations, which from Eqn. (1.2.11) are:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} = 0 \quad (1.4.1)$$

where,

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho e_t \end{bmatrix} \quad \text{and} \quad \mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (\rho e_t + p)u \end{bmatrix} \quad (1.4.2)$$

Again, these expressions are derived from Eqn. (1.2.11) by considering only one space dimension (x) and neglecting the viscous terms. For the present work, Eqn. (1.4.1) shall be referred to as the non-characteristic form of the Euler equations. The alternative characteristic form provides greater insight into the underlying physics of a given problem, most importantly perhaps, in the area of boundary conditions. As we shall see, the characteristic form cannot be found for many problems of interest, yet the information that it conveys is always applicable in the non-characteristic form. Accordingly, both forms play an important role in the current work.

In deriving the characteristic form we take advantage of a useful redefinition. More specifically, we make use of the fact that the total energy, e_t , can be written in terms of the speed of sound, a . The derivation of this equality begins by assuming we are dealing with a perfect gas, which in turn implies that the following relationship holds:

$$p = \rho e(\gamma - 1) \quad (1.4.3)$$

γ is a familiar constant which gives the ratio of specific heats at a constant pressure and volume (Anderson *et al*, 1984). Using $e = e_t - \frac{1}{2}u^2$ we have:

$$p = \rho(\gamma - 1) \left(e_t - \frac{1}{2}u^2 \right) \quad (1.4.4)$$

Solving for e_t in Eqn. (1.4.4) yields:

$$e_t = \frac{p}{\rho(\gamma - 1)} + \frac{1}{2}u^2 \quad (1.4.5)$$

The speed of sound, denoted by a , is defined as:

$$a^2 = \frac{dp}{d\rho} \quad (1.4.6)$$

which, for an isentropic process becomes:

$$a^2 = \gamma \frac{p}{\rho} \quad (1.4.7)$$

Substituting Eqn. (1.4.7) into Eqn. (1.4.5) produces the following expression for e_t :

$$e_t = \frac{a^2}{\gamma(\gamma - 1)} + \frac{1}{2}u^2 \quad (1.4.8)$$

This result proves to be important for simplifying terms in the characteristic form of Eqn. (1.4.1). In seeking the characteristic form however, we must first write Eqn. (1.4.1) in what is known as its primitive variable (or linearized) form, given below:

$$\frac{\partial \mathbf{Q}'}{\partial t} + \mathbf{A}' \frac{\partial \mathbf{Q}'}{\partial x} = 0 \quad (1.4.9)$$

where,

$$\mathbf{Q}' = \begin{bmatrix} \rho \\ u \\ p \end{bmatrix} \quad \text{and} \quad \mathbf{A}' = \begin{bmatrix} u & \rho & 0 \\ 0 & u & \frac{1}{\rho} \\ 0 & \rho a^2 & u \end{bmatrix} \quad (1.4.10)$$

Eqns. (1.4.1) and (1.4.9) represent the same physical laws written in slightly different forms. At this point, a similarity transformation is introduced by which the characteristic variables may be identified. To accomplish this, Eqn. (1.4.9) is expressed in the following equivalent manner:

$$\frac{\partial \mathbf{Q}'}{\partial \mathbf{Q}''} \frac{\partial \mathbf{Q}''}{\partial t} + \mathbf{A}' \frac{\partial \mathbf{Q}'}{\partial \mathbf{Q}''} \frac{\partial \mathbf{Q}''}{\partial x} = 0 \quad (1.4.11)$$

where the Jacobian matrix $\mathbf{R} = \frac{\partial \mathbf{Q}'}{\partial \mathbf{Q}''}$ is:

$$\mathbf{R} = \frac{\partial \mathbf{Q}'}{\partial \mathbf{Q}''} = \frac{\partial(Q'_1, Q'_2, Q'_3)}{\partial(Q''_1, Q''_2, Q''_3)} = \begin{bmatrix} \frac{\partial Q'_1}{\partial Q''_1} & \frac{\partial Q'_1}{\partial Q''_2} & \frac{\partial Q'_1}{\partial Q''_3} \\ \frac{\partial Q'_2}{\partial Q''_1} & \frac{\partial Q'_2}{\partial Q''_2} & \frac{\partial Q'_2}{\partial Q''_3} \\ \frac{\partial Q'_3}{\partial Q''_1} & \frac{\partial Q'_3}{\partial Q''_2} & \frac{\partial Q'_3}{\partial Q''_3} \end{bmatrix} \quad (1.4.12)$$

$$= \begin{bmatrix} 1 & \frac{\rho}{\sqrt{2}a} & \frac{\rho}{\sqrt{2}a} \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & \frac{\rho a}{\sqrt{2}} & \frac{\rho a}{\sqrt{2}} \end{bmatrix}$$

Rearranging Eqn. (1.4.11) we obtain:

$$\frac{\partial \mathbf{Q}''}{\partial t} + \mathbf{R}^{-1} \mathbf{A}' \mathbf{R} \frac{\partial \mathbf{Q}''}{\partial x} = 0 \quad (1.4.13)$$

or,

$$\frac{\partial \mathbf{Q}''}{\partial t} + \mathbf{D} \frac{\partial \mathbf{Q}''}{\partial x} = 0 \quad (1.4.14)$$

It can be shown that $\mathbf{D} = \mathbf{R}^{-1} \mathbf{A}' \mathbf{R}$ has the following form:

$$\mathbf{D} = \begin{bmatrix} u & 0 & 0 \\ 0 & u+a & 0 \\ 0 & 0 & u-a \end{bmatrix} \quad (1.4.15)$$

The columns of $\mathbf{R}$ are the right eigenvectors of $\mathbf{A}'$, and similarly, the rows of $\mathbf{R}^{-1}$ are the left eigenvectors of $\mathbf{A}'$. Furthermore, the structure of $\mathbf{Q}''$ is obtained from:

$$\mathbf{Q}'' = \int \mathbf{R}^{-1} d\mathbf{Q}'$$

$$\begin{aligned}
&= \begin{bmatrix} \int d\rho - \frac{dp}{a^2} \\ \frac{1}{\sqrt{2}} \int (du + \frac{dp}{\rho a}) \\ -\frac{1}{\sqrt{2}} \int (du - \frac{dp}{\rho a}) \end{bmatrix} \\
&= \begin{bmatrix} \int d\rho - \frac{dp}{a^2} \\ \frac{1}{\sqrt{2}} (u + \frac{2a}{\gamma-1}) \\ -\frac{1}{\sqrt{2}} (u - \frac{2a}{\gamma-1}) \end{bmatrix} \tag{1.4.16}
\end{aligned}$$

By expanding the vector form of Eqn. (1.4.16), we are left with the alternative characteristic form of the one-dimensional Euler equations:

$$\frac{\partial}{\partial t} \int \left(d\rho - \frac{dp}{a^2} \right) + u \frac{\partial}{\partial x} \int \left(d\rho - \frac{dp}{a^2} \right) = 0 \tag{1.4.17}$$

$$\frac{\partial}{\partial t} \left(u + \frac{2a}{\gamma-1} \right) + (u+a) \frac{\partial}{\partial x} \left(u + \frac{2a}{\gamma-1} \right) = 0 \tag{1.4.18}$$

$$\frac{\partial}{\partial t} \left(u - \frac{2a}{\gamma-1} \right) + (u-a) \frac{\partial}{\partial x} \left(u - \frac{2a}{\gamma-1} \right) = 0 \tag{1.4.19}$$

The terms $\int \left(d\rho - \frac{dp}{a^2} \right)$, $\left(u + \frac{2a}{\gamma-1} \right)$, and $\left(u - \frac{2a}{\gamma-1} \right)$ are referred to as the Riemann

invariants or characteristic variables. In general, hyperbolic partial differential

equations model wave motion or vibration of structures where the quantities

represented by the Riemann invariants propagate along characteristic lines. For

example, the Riemann invariant $\left(u + \frac{2a}{\gamma-1} \right)$ is propagated along the characteristic line

$C^+ = \frac{dx}{dt} = u + a$ with a speed of $(u + a)$. In a similar fashion, the Riemann invariants

$\left(d\rho - \frac{dp}{a^2}\right)$ and $\left(u - \frac{2a}{\gamma-1}\right)$ are propagated along the characteristic lines $C = \frac{dx}{dt} = u$

and $C^- = \frac{dx}{dt} = u - a$ with speeds of u and $(u - a)$, respectively. This property of

hyperbolic equations is quite important in CFD applications, specifically at computational boundaries. However, before we endeavor to tackle an assortment of CFD issues, we must first understand, in a general sense, how differential equations are numerically solved in a finite domain.

1.5 DISCRETE SOLUTIONS TO THE EQUATIONS OF FLUID FLOW

Until this point, our discussion has focused on the equations of fluid flow from a strictly theoretical point of view. Unfortunately, the mathematical complexity of many practical fluid flow problems precludes closed form solutions. To evade this problem, the governing equations must be solved through numerical approximations known as discretization schemes. Amongst the general classes of discretization schemes used for solving partial differential equations numerically, four of the most common include finite-difference, finite-element, finite-volume, and spectral methods.

Historically, the application of finite-difference methods to the governing equations of fluid mechanics has been the most common among the aforementioned discretization methods. This is primarily due to the simplicity of the schemes and associated structured grid system (the interested reader may consult Section B.4 for the difference between structured and unstructured grids). In finite-differencing, the

discrete representations of the partial derivatives are obtained from Taylor series expansions. The finite-difference equation that approximates the PDE is algebraic in nature, and one such equation is written for each grid point within the domain. In an alternative approach, derivatives are approximated by representing the function as a polynomial. The coefficients of the polynomial are computed by substitution of data (dependent variable) from a series of (usually) equally “spaced” values of the independent variable. Approximate values of the derivatives are then computed from the polynomial. It should be noted that finite-differencing may be used with unstructured grids, however, the process is much more difficult than the structured counterpart.

Although classical finite-difference methods have virtually disappeared from practical use in solid mechanics, these discretization techniques are still important in fluid mechanics applications, particularly with problems which allow for the domain to be discretized into a smooth structured grid.

Conversely, when the integral form of the equations is utilized (as opposed to the differential form), the domain of solution is divided into small cells. Such integral methods include finite-volume and finite-element schemes. For linear and certain non-linear problems, finite-element methods seem to be the dominant choice. However, for the highly non-linear problems of fluid mechanics, finite-volume techniques stand out as the front-runner. Both methods are suitable for use with unstructured grids in solving flows in and around complex geometries.

Spectral methods, which are a subset of the well-known Galerkin methods, rely on a spatial frequency decomposition of the entire domain using global (smooth) basis functions, as in the case of a Fourier series. The benefits of these methods are that derivatives may be computed exactly, with an infinite convergence rate in space (in terms of the order of accuracy), while well-suited basis functions may be chosen for each problem of interest. Despite these apparent advantages, spectral methods present implementation problems and at times fail to represent physical processes in spectral space. In other instances, it is the spatial decomposition that may pose a problem. Since the basis functions are global in nature, the handling of localized flow features or sharp gradients may be quite problematic. Be that as it may, results obtained from experiments where the use of spectral methods is permissible are often accurately impressive.

In general, the field of discretization schemes continues to receive enormous attention. Certain disciplines of engineering have devoted much attention to one particular discretization method, and consequently, have expended much energy endorsing their position. Nevertheless, what is clear at this point, given the diversity in fluid flow problems, is that no one discretization method stands above the rest, particularly when one seeks robustness in numerical simulations. Often combinations of techniques are used. For a more thorough description of discretization techniques, the reader may consult Sections B.1 – B.3 and B.4.5.

To investigate further the discrete solution of differential equations, we shall restrict our attention to finite-differencing schemes. Despite this adopted convention, many of the topics addressed here also apply to other discretization methods.

As an example, suppose we wish to solve a PDE for which $u(x, y)$ is the dependent variable in the square two-dimensional domain $0 \leq x \leq 1, 0 \leq y \leq 1$. Our discretized domain, or grid, is constructed by replacing $u(x, y)$ with $u_{i,j} = u(i\Delta x, j\Delta y) = u(x_i, y_j)$. In this way, grid points can be located according to particular values of i and j (or i, j , and k in three dimensions).

Defining derivatives on a discretized domain can be done by recalling the formal definition for the derivative of a function $u(x, y)$ at $x = x_0, y = y_0$:

$$\frac{\partial u}{\partial x} = \lim_{\Delta x \rightarrow 0} \frac{u(x_0 + \Delta x, y_0) - u(x_0, y_0)}{\Delta x} \quad (1.5.1)$$

If u is continuously differentiable, it is expected that $\frac{u(x_0 + \Delta x, y_0) - u(x_0, y_0)}{\Delta x}$ will be a fairly “reasonable” approximation to $\frac{\partial u}{\partial x}$, provided we use a “sufficiently small” Δx . Of course the Mean Value Theorem assures us that this representation is exact for some point within the interval $[x_0, x_0 + \Delta x]$. The error involved in using such an approximation, referred to as the truncation error, can be studied by considering the Taylor-series expansion for $u(x_0 + \Delta x, y_0)$ about (x_0, y_0) . We shall omit this straightforward exercise. Subsequently, in terms of our finite-difference notation, Eqn. (1.5.2) becomes:

$$\left(\frac{\partial u}{\partial x} \right)_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} \quad (1.5.3)$$

Similarly, a derivative with respect to y can be written as:

$$\left. \frac{\partial u}{\partial y} \right)_{i,j} = \frac{u_{i,j+1} - u_{i,j}}{\Delta y} \quad (1.5.4)$$

Often in the treatment of marching problems, the variation of the marching coordinate (usually time) is indicated by a superscript n , for example $u_{i,j}^n$. Thus, a time-derivative may have the following form:

$$\left. \frac{\partial u}{\partial t} \right)_{i,j} = \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} \quad (1.5.5)$$

The three examples shown in Eqns. (1.5.3) – (1.5.5) are among an infinite number of

difference representations that can be found for $\left. \frac{\partial}{\partial \Pi} \right)_{i,j}$, where Π is some

differentiation variable. As generic schemes, all three are known as “forward” differences, since the evaluation of the derivative, for example with respect to x , involves the subtraction of two points having indices i and $(i + 1)$. Although formal justification will not be given, it should be noted that these schemes are first-order accurate with respect to Δx , Δy , and Δt , respectively. Other common examples include “backward” and “central” difference operators shown below:

$$\text{“Backward Difference”} \quad \left. \frac{\partial u}{\partial x} \right)_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{\Delta x} \quad (1.5.6)$$

$$\text{“Centered Difference”} \quad \left. \frac{\partial u}{\partial x} \right)_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \quad (1.5.7)$$

Within *EE_Solver*, spatial derivatives receive a centered difference treatment (Eqn. (1.5.7)). This methodology holds true wherever computational nodes are available, otherwise appropriate one-sided differences are used, for instance at computational boundaries. Modifications to the finite differencing scheme shall be discussed in Section 1.7. The resulting discretized equations are then integrated in time using a second-order two-stage Runge-Kutta scheme. The concept of Runge-Kutta time integration receives greater attention in Chapter 3.

Solutions to hyperbolic equations are characterized by waves that propagate with little or no loss of amplitude. Thus, it is important for numerical schemes to avoid introducing non-physical dissipation; that is, to unrealistically decrease wave amplitude. Moreover, it is equally important that a numerical scheme not alter the speed at which waves propagate, a phenomenon known as dispersion. These two occurrences, dissipation and dispersion, are generally termed diffusion. Ordinarily, spatial dissipation can be observed in regions of high spatial gradients in flow variables, such as shock waves in high-speed flows. On the other hand, dispersion errors are those errors that arise because components of a solution, having different grid resolution requirements, propagate through the grid with slightly different speeds. Whenever this occurs, unphysical oscillations develop in the solution as a result of the addition or canceling of wave components (Flow, 2003). To avoid such disturbances, the *EE_Solver* introduces artificial diffusion by means of explicit terms, a common procedure found in many CFD algorithms (e.g. Jameson *et al.*, 1981). The connection between artificial diffusion and the Runge-Kutta time integration scheme shall be readdressed in detail in Chapter 3. The interested reader can consult, for example,

Anderson *et al.* (1984) for a thorough investigation into discretization techniques in the context of computational fluid dynamics.

Before continuing with our analysis of Riemann invariants, one last issue that must be discussed is the notion of stability in regards to numerical schemes.

1.5.1 THE NUMERICAL STABILITY OF DISCRETIZATION SCHEMES

It can be said that numerical stability is a concept applicable exclusively to marching problems or problems that are solved iteratively. A stable discretization scheme is one for which errors from any source (round-off, truncation, coding mistakes) are not permitted to accumulate as the calculation proceeds from one marching step to the next. The manner in which the difference operator is formed to produce a solution for successive steps is the central concern of stability analyses.

There are several procedures for carrying out a stability analysis of a discretization scheme (e.g. Anderson *et al.* (1984)), of which, the most utilized among linear problems is the von Neumann analysis. It should be noted that the use of von Neumann analysis with non-linear problems requires a great deal of justification, and in most instances, is not applicable. Hence, we limit the following description to linear problems, and once again stay within the framework of finite-differencing schemes.

In von Neumann analysis, we use the fact that the computed numerical solution, as well as the error, must satisfy the difference equation. In particular, the error, $\varepsilon(x, t)$, can be written as a series of the form:

$$\mathcal{E}(x, t) = \sum_m b_m(t) e^{ik_m x} \quad (1.5.1.1)$$

The period of the fundamental frequency ($m = 1$) is assumed to be $2L$, where L is the length of the interval of interest, and b_m can be taken to be of the form e^{at} , for some real or complex constant a . The wave number, k_m , may be written as:

$$k_m = \frac{m\pi}{L}, \quad m = 0, 1, \dots, M \quad (1.5.1.2)$$

Here M is the number of intervals of length Δx contained in the longer interval of length L . One will notice that the technique is essentially analogous to representing u as a Fourier series on $[0, L]$. Moreover, since the difference equation is linear, superposition allows us to examine the behavior of a single term of the series given in Eqn. (1.5.1.1). To facilitate our understanding of stability analyses, we consider Lax's method (1954) for solving the first-order wave equation:

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0 \quad (1.5.1.3)$$

where c is the wave speed (more will be said about the one-dimensional wave equation in the following section). Lax proposed the following first-order method for solving equations of this type:

$$u_j^{n+1} = \frac{u_{j+1}^n - u_{j-1}^n}{2} - c \frac{\Delta t}{\Delta x} \left(\frac{u_{j+1}^n - u_{j-1}^n}{2} \right) \quad (1.5.1.4)$$

Eqn. (1.5.1.4) is known as an explicit finite difference formulation since the solution (of u in this case) at time-step $(n + 1)$ is written as a function of variables at time-steps strictly less than $(n + 1)$. Conversely, implicit schemes are those formulations which depend on quantities computed at the current time-step. In this latter case, a matrix or iterative technique is required in order to obtain the solution. In general, the use of explicit schemes involves fewer calculations as opposed to implicit schemes. On the other hand, explicit formulations tend to suffer from stricter stability (time-step) bounds.

To analyze the stability of Lax's scheme, we assume the following solution form:

$$u_j^n = e^{at} e^{ik_m x} \quad (1.5.1.5)$$

Substituting Eqn. (1.5.1.5) into the difference Eqn. (1.5.1.4) results in:

$$e^{a\Delta t} = \cos(k_m \Delta x) - i \frac{c\Delta t}{\Delta x} \sin(k_m \Delta x) \quad (1.5.1.6)$$

In order for the numerical solution to remain bounded as Eqn. (1.5.1.4) is marched in time, we require the following stability condition:

$$\left| e^{a\Delta t} \right| \leq 1 \quad (1.5.1.7)$$

Clearly, if this condition holds, a general component of the error will not grow from one step to the next, according to the form of Eqn. (1.5.1.1) with $b_m = e^{at}$.

Combining Eqns. (1.5.1.6) and (1.5.1.7) gives:

$$\left| \cos(k_m \Delta x) - i \frac{c \Delta t}{\Delta x} \sin(k_m \Delta x) \right| \leq 1$$

$$\Rightarrow \left| c \frac{\Delta t}{\Delta x} \right| \leq 1 \quad (1.5.1.8)$$

This final expression (1.5.1.8) follows from the triangle inequality, and is commonly known as the Courant-Friedrichs-Lewy (CFL) condition.

An alternative, and perhaps more effective way of understanding the stability of numerical discretization schemes is through the use of graphical stability regions. For example, the stability condition (1.5.1.8) is equivalent to the requirement that the ellipse defined by $\left| \cos(k_m \Delta x) - i \frac{c \Delta t}{\Delta x} \sin(k_m \Delta x) \right|$ be contained in the unit circle, as shown in Fig. 1.5.1.1.

Figure 1.5.1.1 – The stability region for the finite-difference scheme given in Eqn. (1.5.1.4)

Interestingly enough, the numerical stability requirement for most explicit methods used to solve hyperbolic PDEs of this form is similar to this CFL condition.

Comparable analyses for systems of equations, particularly multi-dimensional problems, may be carried out in a like fashion. Irrespective of this fact, the notion of numerical stability is always present in the discrete solution of a set of equations.

Thus, one must continually keep in mind the trade-off between increased node-counts (decrease in Δx) to achieve greater spatial resolution, and the associated increase in computational time (decrease in Δt); the two being balanced by the problem's numerical stability bounds. Procedures such as the use of implicit schemes, implicit residual smoothing, local time-stepping, and multigrid acceleration techniques, can help to overcome the often restrictive stability bounds placed on Δt by the use of an explicit scheme such as that of Eqn. (1.5.1.4). Discussion of these topics may be found within Appendix B.

In reference to time marching procedures, the use of Runge-Kutta schemes proves to be rather beneficial, as they often allow for larger time-steps as compared to other popular discretization schemes.

In the words of Trefethen (1994), “Runge-Kutta formulas tend to be not very unique”. Using his notation, if we say that a particular method is to have s stages with an order of accuracy p , then for most values of s there are infinitely many Runge-Kutta formulas with the maximal order of accuracy p . In particular, Trefethen provides the following diagram outlining the stability regions for Runge-Kutta formulas having $s = 1$ to 4 stages:

Figure 1.5.1.2 – Stability regions for the Runge-Kutta schemes with $s = 1, \dots, 4$ stages, reproduced from Trefethen (1994)

In recent decades the classical explicit Runge-Kutta schemes have been slowly supplanted by implicit formulas. Despite the fact that, throughout the CFD literature many Runge-Kutta variations exist, the prevalent choice among researchers appears to still be a traditional, or variant, four-stage scheme. Normally, interest lies in obtaining the greatest order of accuracy with the fewest number of stages. Interestingly enough, if one requires p orders of accuracy, then one needs at least p Runge-Kutta stages. Moreover, Table 1.5.1.1 indicates that it is only up to and including $p = 4$ where p stages are required for an order of accuracy p , otherwise one is forced to use a greater number of stages:

Table 1.5.1.1 – The minimum number of Runge-Kutta stages needed to obtain specific order of accuracy, reproduced from Trefethen (1994)

Order p	Minimum number of stages s
1,2,3,4	1,2,3,4 (respectively)
5	6
6	7
7	9
8	11
9	12 - 17 (precise minimum unknown)
10	13 - 17 (precise minimum unknown)

Consequently, a four stage Runge-Kutta scheme often provides an optimal balance between numerical accuracy and computational expense. Nonetheless, the best we can affirm is that these theoretical observations are not always subject to strict adherence, for often in numerical experimentation, the most “intuitive” choice does not necessarily lead to the most accurate solution.

Now that we have a general understanding of how the equations of fluid flow are solved in a discrete manner, we return to our discussion on Riemann invariants and their fundamental connection to boundary conditions. In particular, boundary specification of data is crucial to the existence of solutions to hyperbolic systems. The well-posedness of any given problem depends largely on the specification of initial and boundary conditions. Ironically, through the specification of these conditions, numerical errors, which tend to pose the greatest challenge to convergence of numerical simulations, are introduced. Therefore, let us use the final two sections of this chapter in an attempt to unify what we have seen thus far.

1.6 THE ROLE OF RIEMANN INVARIANTS IN COMPUTATIONAL FLUID DYNAMICS

If the boundary of a computational domain is representative of a true physical boundary, then appropriate boundary conditions can generally be derived from physical principles, and subsequently be used in a numerical algorithm. More serious difficulties arise when the computational domain must be terminated at some arbitrary point, and an artificial boundary implemented at this location. The conditions imposed at this artificial boundary are intended to mimic the presence of the surrounding fluid, hence allowing any outgoing disturbances to pass through the boundary without generating spurious reflections back into the computational domain.

Boundary conditions for hyperbolic problems, such as those of time-dependent fluid flows are intimately related to the theory of characteristics and Riemann invariants introduced earlier. As we observed in Section 1.4, the fundamental aspect of such problems involves the propagation of waves along specific paths with characteristic speeds. In order to gain a deeper appreciation for the connection between boundary

conditions and Riemann invariants, the linear advection equation and the one-dimensional Euler equations in characteristic form are considered here. At the same time, we look at the ramifications of using numerical approximations to exact outflow boundary conditions.

To begin, consider again the linear one-dimensional advection or wave equation with the following initial and boundary conditions:

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0, \quad c > 0, c \in \mathbb{R} \quad (1.6.1.a)$$

$$\text{Initial condition: } u(x, 0) = f(x) \quad (1.6.1.b)$$

$$\text{Boundary condition: } u(0, t) = g(t) \quad (1.6.1.c)$$

$$f(0) = g(0) \quad (1.6.1.d)$$

Suppose that u is to be determined on some domain $0 \leq x \leq L$, $L \in \mathbb{R}$. If initial data is given on this domain, then we would like to know which boundary conditions guarantee the well-posedness of this problem. The general solution to Eqn. (1.6.1.a) is given by $u = F(x - ct)$, for some arbitrary function F . Thus, u will be constant along the characteristic curves: $(x - ct) = x_0$, for some constant $x_0 \in \mathbb{R}$.

Furthermore, the initial data will uniquely determine u in the shaded region of Fig. 1.6.1. The solution in the remaining section, $t > 0$, $0 \leq x \leq L$, is determined by the values of $u(0, t)$ at the inflow boundary, as shown in Figure 1.6.2.

Figure 1.6.1 – Portion of the x - t plane in which the solution to Eqn. (1.6.1.a) is determined by the initial data $u(x,0)$, $0 \leq x \leq L$, reproduced from Durran, 1999

Figure 1.6.2 – Two arbitrary characteristic lines for the initial boundary value problem given by Eqns. (1.6.1.a) – (1.6.1.d)

For the characteristic passing through (x_1, t_1) we may write:

$$u(x_1, t_1) = u(x_2, 0) = f(x_2) = f(x_1 - ct_1) \quad (1.6.2)$$

In actual fact, for any point (x, t) such that $x - ct \geq 0$, the solution can be related to the initial conditions by:

$$u(x, t) = f(x - ct), \quad x \geq ct \quad (1.6.3)$$

Consider now the point (x_3, t_3) . Using the method of characteristics we have:

$$u(x_3, t_3) = u(0, t^*) = g(t_3 - \frac{x_3}{c}) \quad (1.6.4)$$

This implies that the solution has dependency on the boundary condition $g(t)$ and not the initial condition. Thus, in general:

$$u(x, t) = g(t - \frac{x}{c}), \quad x < ct \quad (1.6.5)$$

Therefore, if we impose a boundary condition at $x = L$, the problem becomes ill-posed, as we have over-specified the solution. A consistent solution might be obtained if the correct value of $u(x, t)$ is specified at $x = L$, but even in this case, small perturbations in the boundary values shows that the solution does not depend continuously on the boundary condition, and thus remains ill-posed. Similar problems arise in computational fluid dynamics as a result of the dependency of a boundary condition on the solution, which in most cases is not known a priori.

The above result may be generalized as follows. Suppose for a system such as that given in Eqns. (1.4.8) and (1.4.13) with A' an $n \times n$ matrix and D having n real eigenvalues (since the equations are hyperbolic), D has m positive eigenvalues and $(n - m)$ negative eigenvalues, with spatial boundaries located at $x = 0$ and $x = L$. Then, since there exists m positive eigenvalues (corresponding to m Riemann

invariants) along which information is transmitted into the domain at $x = 0$, m boundary conditions are required at $x = 0$. Similarly, the $(n - m)$ negative eigenvalues imply that information is transmitted into the domain along characteristic lines at $x = L$, thus requiring $(n - m)$ boundary conditions there. Therefore, from Section 1.4 we know that a subsonic flow governed by the one-dimensional Euler equations requires specification of two boundary conditions at the inflow ($x = 0$), corresponding to the characteristics u and $(u + a)$, and one boundary condition at the outflow ($x = L$), a consequence of the characteristic $(u - a)$. This idea is summarized in Table 1.6.1 for a general computational domain. The Mach number, M , is an important non-dimensional parameter defined as the ratio of an object's speed in a particular medium to the ambient speed of sound in that same medium. When $M < 1$ we refer to the object's speed as being subsonic, whereas $M > 1$ denotes supersonic flow.

In essence, the characteristic form “dissects” a given problem by identifying the individual components that influence the final solution. Mathematically this translates into a certain number of waves entering and exiting each inlet and outlet boundary. More importantly, for time-dependent boundary value problems, those “incoming” waves identified by this mathematical dissection each require a boundary condition. The ideal situation would be to specify the associated Riemann invariant in each case. Unfortunately, as we shall observe in the next section, higher dimensional boundary treatments based on the notion of Riemann invariants present a much greater implementation challenge.

Table 1.6.1 – Boundary condition specification and extrapolation required at subsonic and supersonic boundaries

	INFLOW		OUTFLOW	
	Subsonic	Supersonic	Subsonic	Supersonic
Number of B.C. to be Specified	2	3	1	0
Number of B.C. to be Extrapolated	1	0	2	3

1.6.1 THE HIGHER DIMENSIONAL APPLICATION OF RIEMANN INVARIANTS

The problems encountered thus far have involved two independent variables, particularly, time-dependent partial differential equations in one spatial dimension. The consequences of this assumption are clear: wave propagation occurs either in the positive or negative spatial direction. When there are more than two independent variables, which represent the majority of the problems studied in fluid dynamics, the situation becomes more difficult. The characteristic curves now become characteristic manifolds in a higher dimensional space-time. As Lappas *et al.* (1999) point out, these manifolds can be viewed as being generated, locally, by appropriate vector fields. There are now more than two available directions for the propagation of waves and the equations no longer simplify the same way as in the one-dimensional cases. The work of Lappas *et al.* is an excellent resource in this area. In order to emphasize this difficulty more clearly, consider the following illustrative example involving the three-dimensional Euler equations for a compressible flow:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} = 0 \quad (1.6.1.1)$$

where $\mathbf{Q}$, $\mathbf{E}$, $\mathbf{F}$, and $\mathbf{G}$ are given in Eqns. (1.2.9) – (1.2.12). As in Section 1.4, this set of equations may be written in primitive variable form. That is:

$$\frac{\partial Q}{\partial t} + A \frac{\partial Q}{\partial x} + B \frac{\partial Q}{\partial y} + C \frac{\partial Q}{\partial z} = 0 \quad (1.6.1.2)$$

where,

$$A = \frac{\partial E}{\partial x} \left(\frac{\partial Q}{\partial x} \right)^{-1}, \quad B = \frac{\partial F}{\partial y} \left(\frac{\partial Q}{\partial y} \right)^{-1}, \quad C = \frac{\partial G}{\partial z} \left(\frac{\partial Q}{\partial z} \right)^{-1} \quad (1.6.1.3)$$

Moreover, the matrices **A**, **B**, and **C** can be put into diagonal forms **L**, **M**, and **N** by the following similarity transformations:

$$PAP^{-1} = L, \quad SBS^{-1} = M, \quad TCT^{-1} = N \quad (1.6.1.4)$$

Eqn. (1.6.1.2) then becomes:

$$\frac{\partial Q}{\partial t} + P^{-1}LP \frac{\partial Q}{\partial x} + S^{-1}MS \frac{\partial Q}{\partial y} + T^{-1}NT \frac{\partial Q}{\partial z} = 0 \quad (1.6.1.5)$$

This can be thought of as the analogous form of Eqn. (1.4.13) in three-dimensional space. However, this is as close to the characteristic form (Eqns. (1.4.17) – (1.4.19)) as we can come unless **P**, **S**, and **T** are equal (that is, **A**, **B**, and **C** are simultaneously diagonalizable). Hence, in many cases, determining the Riemann invariants can be rather difficult and at times, impossible. It is predominantly for this reason that the non-characteristic form of the governing equations is preferred for numerical implementation. Nonetheless, the hyperbolic nature of the problem implies that the

same number of waves will continue to enter and exit particular boundaries, albeit the connection with boundary condition specification is more complex.

These difficulties and the availability of experimental information have led to improvisations wherein flow variables such as temperature, pressure, or velocity are specified in place of the incoming Riemann invariants. The major ramification of such alternative boundary conditions can be called, as a general name, reflection. This phenomenon continues to receive enormous attention in the field of CFD, and is the subject of our next section.

1.7 WAVE REFLECTION AT OUTFLOW BOUNDARIES

Ideally, we would like for incident waves reaching a particular boundary from within the computational domain (i.e. outgoing Riemann invariants) to pass through the boundary as if it were not there; that is, to behave according to the exact solution. Unfortunately, the more a boundary condition deviates from the exact physics of the problem (dictated by the Riemann invariant), the greater its negative influence on the numerical solution.

To illustrate how errors are generated when approximations are made to the exact Riemann invariant boundary conditions, consider the last two equations in the characteristic form of the Euler equation set (1.4.17) – (1.4.19), namely:

$$\frac{\partial}{\partial t} \left(u + \frac{2a}{\gamma-1} \right) + (u+a) \frac{\partial}{\partial x} \left(u + \frac{2a}{\gamma-1} \right) = 0 \quad (1.7.1)$$

$$\frac{\partial}{\partial t} \left(u - \frac{2a}{\gamma-1} \right) + (u-a) \frac{\partial}{\partial x} \left(u - \frac{2a}{\gamma-1} \right) = 0 \quad (1.7.2)$$

Furthermore, suppose we wish to solve the two advection equations on the domain $0 \leq x \leq L$. As illustrated in Section 1.4, the exact value of the Riemann invariants for Eqns. (1.7.1) and (1.7.2) are known. Nevertheless, our intent here is to study reflection due to “non-ideal” boundary conditions, and so to facilitate this, we assume that a zero gradient boundary condition is imposed at both $x = 0$ and $x = L$. Without loss of generality, $x = 0$ shall be taken as the outflow boundary. To evaluate the strength of the reflection generated at this outlet, we examine the behaviour of a unit-amplitude incident wave as it reflects off the outflow boundary. Thus, assume u has the following form:

$$u(x, t) = \sin[k_i(x - (u - a)t)] + r \sin[k_r(x - (u + a)t) + \phi] \quad (1.7.3)$$

In Eqn. (1.7.3), k is the wave number, r is the amplitude of the reflected wave, and the subscripts i and r refer to incident and reflected wave, respectively. Finally, ϕ represents the phase of the reflected wave. The first term of the sum in Eqn. (1.7.3) corresponds to the unit-amplitude wave propagating toward the $x = 0$ boundary, while the second term signifies an arbitrary reflected wave traveling away from that boundary. In addition, the amplitude, wave number, and phase of this reflected wave are determined by the boundary condition at the outflow boundary. For that reason, if a zero gradient condition $\frac{\partial u}{\partial x} = 0$ is specified at $x = 0$, then we have:

$$k_i \cos[k_i (u - a)t] + r k_r \cos[k_r (u + a)t + \phi] = 0 \quad (1.7.4)$$

Since Eqn. (1.7.4) must hold for all values of t , the two terms must be linearly dependent functions of time. As a consequence, we have that the frequencies of the incident and reflected waves must be the same, and that $\phi = 0$ (or $\phi = \pi$). The wave number of the reflected wave can thus be written as:

$$k_r = k_i \left(\frac{u - a}{u + a} \right) \quad (1.7.5)$$

Combining this result with Eqn. (1.7.4) gives us the amplitude of the reflected wave:

$$r = -\frac{k_i}{k_r} = -\frac{u + a}{u - a} \quad (1.7.6)$$

From Eqns. (1.7.5) and (1.7.6) one may see that if $u > 0$, the wavelength and the amplitude of the reflected wave are greater than those of the incident wave.

Following the initial reflection at $x = 0$, the reflected wave will traverse across the domain and reflect a second time at $x = L$. A similar argument shows that the amplitude of the reflection at $x = L$ due to the zero gradient boundary condition is given by:

$$\hat{r} = -\frac{u - a}{u + a} \quad (1.7.7)$$

Thus, after reflection takes place at both $x = 0$ and $x = L$, the amplitude of the spurious wave will be $r\hat{r} = 1$, implying that no energy has escaped through the boundaries. As

one might expect, the same reflection problem is encountered when we alternatively specify $\frac{\partial a}{\partial x} = 0$. Clearly the zero gradient boundary condition is not an appropriate substitute for the exact Riemann invariant boundary condition. It should be noted of course, that more accurate approximations exist when the Riemann invariants are not explicitly known. Nonetheless, the above example adequately identifies outflow boundary reflection as a potential roadblock to convergence.

The presence of incoming Riemann invariants or waves is a mathematical property of the governing equations in question. Boundary conditions in CFD problems are required in order to account for the existence of these incoming waves. The last subsection of this chapter considers the errors that may result when one seeks the numerical solution of a fluid flow problem on a discretized domain.

1.7.1 PHYSICAL VS. NON-PHYSICAL MODES

Once physically appropriate well-posed boundary conditions have been formulated, the problem is ready for numerical solution. Unfortunately, the process of replacing continuous spatial derivatives with finite differences often introduces additional errors into the solution procedure. Two primary reasons account for these difficulties. To begin with, finite difference formulas often require boundary information which is not needed in the well-posed continuous problem. The second difficulty stems from the fact that most finite-difference formulations introduce non-physical modes that propagate in a direction opposite to, and adversely affect, the correct physical solution.

Consider once again the linear one-dimensional wave equation shown in Eqn.

(1.6.1.a). If the x domain is periodic or unbounded, individual mode solutions may be sought in the form of a traveling wave:

$$u(x, t) = e^{i(kx - \omega t)} \quad (1.7.1.1)$$

where k is the wave number and ω is the frequency. Substitution of Eqn. (1.7.1.1) into Eqn. (1.6.1.a) implies that the traveling wave will satisfy the governing equations provided its frequency satisfies the dispersion relation:

$$\omega = ck \quad (1.7.1.2)$$

The wave travels with constant amplitude at a phase speed $\frac{\omega}{k} = c$. These waves are nondispersive, that is, their phase speed is independent of the wave number. Also independent of the wave number is the group velocity, $\frac{\partial \omega}{\partial k} = c$. The energy associated with an isolated “packet” of waves travels at a speed equal to the group velocity. Unfortunately, in the discrete form of the equations, the notions of phase speed and group velocity become altered.

Spatial discretization errors can be studied by considering the semi-discrete form of Eqn. (1.6.1.a) wherein only the spatial derivative is replaced by a discrete representation:

$$\frac{\partial u}{\partial t} + c \left[\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} \right] = 0, \quad 0 \leq x \leq L \quad (1.7.1.3)$$

for $j = 0, \dots, N$ discrete grid points. Eqn. (1.7.1.3) cannot be used to calculate u_0 and u_N as this would require information ahead of $x = 0$ and beyond $x = L$. In order to compare with the comments of the previous section, let us assume that $c < 0$, making $x = L$ an inflow boundary. With this convention, a boundary condition $u(L, t) = g(t)$ is required for a well-posed continuous problem (i.e. one incoming Riemann invariant), with no boundary condition at $x = 0$. The boundary condition at $x = L$ may be used to specify u_N , however, $u(0, t)$ must be determined, in some manner, by the interior solution. As central-differencing cannot be used to obtain u_0 , some employ the following simple extrapolation:

$$u_0 = u_1 \quad (1.7.1.4)$$

Eqn. (1.7.1.4) is the zero-gradient boundary condition discussed in the previous section, and from the comments made there, one might expect reflections from this choice. As a matter of fact, reflection does occur, however, it is fundamentally different than before. In particular, Eqns. (1.7.1) and (1.7.2) support physical modes that propagate both to the right and to the left. The zero-gradient boundary conditions used in Section 1.7 generated reflections from the outgoing physical modes into the incoming physical modes because $\frac{\partial u}{\partial x} = 0$ inadequately approximated the exact Riemann invariants. Conversely, all physical solutions to Eqn. (1.6.1.a) travel in the same direction, and so, reflections that are brought upon by the use of the zero-gradient boundary condition (1.7.1.4) involve non-physical modes. To get an idea of how much reflection is produced through the interaction of physical and non-physical modes, we assume solutions to Eqn. (1.7.1.3) of the following form:

$$u(x,t) = e^{i(kj\Delta x - \omega t)} \quad (1.7.1.5)$$

Following substitution of Eqn. (1.7.1.5) into Eqn. (1.7.1.3), we obtain the following dispersion relation:

$$\omega = c \frac{\sin k\Delta x}{\Delta x} \quad (1.7.1.6)$$

Since ω is real, there is no change in wave amplitude with time, and thus, no amplitude error. On the other hand, the phase speed is given by:

$$\frac{\omega}{k} = c \frac{\sin k\Delta x}{k\Delta x} \quad (1.7.1.7)$$

Unlike the Fourier analysis solutions obtained with the continuous problem, the discrete solutions will be dispersive as the phase speed is now a function of k . More specifically, if the grid resolution is good (typically $k\Delta x \ll 1$), the Taylor series approximation $\sin x \approx x - \frac{x^3}{6}$ can be used to show that the phase-speed error is second order in $k\Delta x$. Of course this error becomes more pronounced as spatial resolution decreases. In fact, the worst resolution occurs when the grid has a wavelength of $2\Delta x$ and wave number $k = \frac{\pi}{\Delta x}$. From Eqn. (1.7.1.7) we can see that this amounts to a significant error; specifically, a phase speed of zero.

A similar argument holds for the group velocity:

$$\frac{\partial \omega}{\partial k} = c \cos k \Delta x \quad (1.7.1.8)$$

The group velocity of well-resolved waves is approximately correct. Unfortunately, as the spatial resolution diminishes accuracy is greatly hindered. For example, the group velocity of the $2\Delta x$ wave is $-c$, indicating that energy actually propagates backwards.

For each ω there are two different wave numbers satisfying the dispersion relation (1.7.1.6). If k_1 is the first wave number satisfying the dispersion relation, then

$k_2 = \frac{\pi}{\Delta x} - k_1$ is the second. The group velocities corresponding to these two wave

numbers are equal in magnitude but opposite in sign. A division can be made

between physical modes, which lie in the region $0 \leq k \Delta x < \frac{\pi}{2}$, and computational

modes, having $\frac{\pi}{2} \leq k \Delta x \leq \pi$. Substitution of this information into Eqn. (1.7.1.8)

shows that the physical modes travel in the same direction as the true physical solution, while the computational modes propagate in the opposite direction (the $4\Delta x$ wave has a group velocity of zero).

In order to quantify the strength of the reflection introduced when using boundary condition (1.7.1.4), we assume that a disturbance oscillating at the frequency ω is present at the left boundary. As this disturbance must satisfy Eqn. (1.7.1.3) it takes the following form:

$$\left(\alpha e^{ikj\Delta x} + \beta e^{i(\pi - k\Delta x)j} \right) e^{-i\omega t} \quad (1.7.1.9)$$

If we let the first term in Eqn. (1.7.1.9) denote the incident wave (with $\alpha = 1$) and the second term the reflected wave (with $\beta = r$), we may rewrite the disturbance as:

$$\left(e^{ikj\Delta x} + r(-1)^j e^{-ikj\Delta x} \right) e^{-i\omega t} \quad (1.7.1.10)$$

For $0 < x < L$ a single mode solution with $r = 0$ is allowable, however, the zero-gradient boundary condition requires a nonzero amplitude in both modes. Thus, substitution of Eqn. (1.7.1.10) into Eqn. (1.7.1.4) yields:

$$1 + r = e^{ik\Delta x} - r e^{-ik\Delta x} \quad (1.7.1.11)$$

Solving for the amplitude we have:

$$r = i e^{ik\Delta x} \tan \frac{k\Delta x}{2} \quad (1.7.1.12)$$

From Eqn. (1.7.1.12) we conclude that:

$$|r| = \left| \tan \frac{k\Delta x}{2} \right| \quad (1.7.1.13)$$

One can see from Eqn. (1.7.1.13) that when the incident wave is well-resolved ($k\Delta x \ll 1$), very little reflection occurs. In contrast, the magnitude of the reflection coefficient is equal to 1 for $4\Delta x$, and rises to infinity for a $2\Delta x$ wave. Despite these errors, when the zero-gradient boundary condition is applied at the outflow boundary

for a problem involving the linear advection equation, no waves shorter than $4\Delta x$ will ever reach the outlet as the group velocities of these short waves are directed upstream. In contrast, $2\Delta x$ and $4\Delta x$ waves reaching the inflow boundary produce larger reflections, rendering the zero-boundary condition unacceptable at the inlet. For further discussions regarding reflection from computational boundaries see the work of Matsuno (1966) and Nitta (1962).

Undoubtedly, carrying out a similar analysis for the full Navier-Stokes equations in three dimensions poses a greater challenge. Nevertheless, the above examples provide important insight into the difficulties introduced into a numerical solution of differential equations when non-ideal boundary conditions are used.

Having this theoretical background in place, we are now equipped to examine how researchers have attempted to formulate outflow boundary conditions with the ultimate aim of reducing, or even eliminating, spurious reflections of outgoing characteristics back into the computational domain.

2.0 LITERATURE SURVEY ON OUTFLOW BOUNDARY TREATMENT

As Roe (1986) writes, “A recurring frustration in Computational Fluid Dynamics is the apparent difficulty of giving numerical expression to very simple statements.”

The statement of concern here is specifically that a boundary be transparent to waves of any kind. Although exact boundary conditions that ensure well-posedness can be derived for the Euler equations (Kreiss (1970) and Enquist and Majda (1977)), it is not the case for the Navier-Stokes equations. In the following sections we describe, in a fairly comprehensive manner, a subset of the various techniques that exist for the treatment of outflow boundaries in CFD simulations, as well as outlining their strengths and weaknesses. While presenting the entire contribution herein is an unrealistic task, we feel the following survey is representative of the most widely used methods. From this we make suggestions on how the treatment of numerical boundary conditions can be made more efficient, both in terms of the reduction of wave reflection and the computational time involved.

2.1 CLASSIFICATION OF OUTFLOW BOUNDARY TREATMENTS

There are two main techniques used for the treatment of outflow boundaries. In general, they can be described as absorbing (non-reflecting) boundary conditions and absorbing (non-reflecting) buffer zones. The term “non-reflecting boundary condition” is somewhat of a misnomer, perhaps due to the variety of other names that have been used (radiating, absorbing, silent, transmitting, transparent, open, free-space, and one-way boundary conditions). More appropriately, the non-reflecting boundary condition is defined as one which does not give rise to spurious reflections. With a small modification, such a condition can allow for the possibility of incoming

waves, that in general would be present, to enter the domain. In addition, since both aforementioned absorbing techniques share this property, we shall simplify matters by referring to them as non-reflecting boundary conditions (NRBCs) and non-reflecting buffer zones (NRBZs).

In the first approach, one designs a local boundary condition approximating a non-local absorbing one. It is well known that in one-dimensional problems, perfectly absorbing boundary conditions can be readily devised (Engquist and Majda (1977)). Unfortunately, in three dimensions complications arise due to the varying angle of incidence of waves.

In the second approach a region referred to as a buffer zone is added to the computational domain adjacent to the boundaries of interest, and the original equations are modified within this buffer zone to produce the desired absorption. With this background knowledge in place, let us now consider each of these techniques in greater depth.

2.2 NON-REFLECTING BOUNDARY CONDITIONS (NRBCs)

In his 1991 review paper, Givoli explains that in devising a new NRBC one has in mind at least some of the following goals:

1. The problem within the interior domain together with the boundary conditions is mathematically well-posed.
2. The problem within the interior domain together with the boundary conditions is a good approximation of the original problem in the infinite domain.

3. The boundary condition should be highly compatible with the numerical scheme used within the interior domain.
4. The numerical method employed together with the boundary conditions must result in a stable numerical scheme.
5. The amount of spurious reflection generated by the boundary conditions is small.
6. The use of the boundary conditions does not involve a large computational effort.
7. In time-dependent schemes where only the steady state solution is sought, the numerical scheme should reach the steady state rapidly.

By industry standards, methods that successfully address points 5 and 6 (assuming the other, more fundamental conditions are satisfied) are deemed practical for application.

2.2.1 ABSORBING BOUNDARY CONDITIONS OF ENGQUIST AND MAJDA

Perhaps the most referenced work on NRBCs is that of Engquist and Majda (1977). In this work, perfectly absorbing boundary conditions for general classes of wave equations are developed based on the theory of pseudo-differential operators (Treves (1980)). Unfortunately, these boundary conditions must be non-local in both space and time, and thus are not useful with most numerical algorithms. As a result, hierarchies of highly absorbing local boundary conditions that approximate the theoretical non-local conditions are derived. For example, three such boundary conditions are considered for the two-dimensional scalar wave equation:

$$\frac{\partial^2 w}{\partial t^2} = \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}, \quad t, x \geq 0 \quad (2.2.1.1)$$

$$\left. \frac{\partial w}{\partial x} - \frac{\partial w}{\partial t} \right|_{x=0} = 0 \quad (2.2.1.2)$$

$$\left. \frac{\partial^2 w}{\partial t \partial x} - \frac{\partial^2 w}{\partial t^2} + \frac{1}{2} \frac{\partial^2 w}{\partial y^2} \right|_{x=0} = 0 \quad (2.2.1.3)$$

$$\left. \frac{\partial^3 w}{\partial t^2 \partial x} - \frac{\partial^3 w}{\partial t^3} - \frac{1}{4} \frac{\partial w^3}{\partial y^2 \partial x} + \frac{3}{4} \frac{\partial^3 w}{\partial y^2 \partial t} \right|_{x=0} = 0 \quad (2.2.1.4)$$

Each of these boundary conditions is well-posed and perfectly absorbing at normal incidence. When the angle of incidence is increased to 45° , boundary conditions (2.2.1.2) – (2.2.1.4) reflect waves with respective amplitudes of 17%, 3%, and 0.5% as compared to the amplitude of the incident wave. Similarly, two highly absorbing boundary conditions for the linearized shallow water equations were derived and their effectiveness in reducing wave reflection is shown in Table 2.2.1.1. The methodology employed is the same in both cases and can be found in the paper by Engquist and Majda (1977). Using procedures similar to those examined in Section 1.7, Engquist and Majda derive the following reflection factor relationship for boundary condition (2.2.1.3):

$$|r| = \left| \frac{1 - \cos \theta}{1 + \cos \theta} \right| \quad (2.2.1.5)$$

Eqn. (2.2.1.5) gives a clear indication of the relationship between angle of incidence and reflection. As one would expect, boundary condition (2.2.1.3) is perfectly nonreflecting when the incident waves approach the boundary along a line normal to

the boundary. Reflection factors for the remaining boundary conditions may be found in the work of Engquist and Majda (1977).

Table 2.2.1.1 – Reduction in wave reflection due to two highly absorbing boundary conditions for the linearized shallow water equations, reproduced from Engquist *et al.* (1977)

Boundary condition at $x = 0$:	$\ u \text{ reflected}\ _2$	$\ v \text{ reflected}\ _2$	$\ \varphi \text{ reflected}\ _2$
Perfect reflection $u = v = 0$:	0.621	0.702	0.695
First absorbing approximation:	0.103	0.067	0.053
Second absorbing approximation:	0.024	0.024	0.013

For the first absorbing approximation of Table 2.2.1.1, reflection of u is reduced to 16.6% of the original total reflection norm $\left(\frac{0.103}{0.621} \times 100\right)$. Similarly, reflection of v is reduced to 9.5% of its original total reflection quantity and φ to 7.6%. The second absorbing approximation fares better with 3.9%, 3.4%, and 1.9% for the same three variables. The problems associated with the implementation of this technique shall be addressed in Section 2.2.3.

2.2.2 NRBCs OF HEDSTROM, THOMPSON, POINSOT AND LELE

Thompson (1987) extends Hedstrom's (1979) concept of non-reflecting boundary conditions to the Euler equations in both rectangular and spherical coordinates for the one-dimensional case, and rectangular coordinates for a two-dimensional problem. Poinsot and Lele (1991) modify this work for application to the two-dimensional

Euler and Navier-Stokes equations. The following description relies on the above contributions.

Let us first consider the one-dimensional case in orthogonal (but not necessarily rectangular) coordinates. The derivation of the three dimensional analogue is essentially the same and can be found in the work of Thompson (1986).

Suppose we have a system of n equations in n dependent variables. Let $\mathbf{Q}$ be the vector of conservative variables satisfying:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \mathbf{C} = 0 \quad (2.2.2.1)$$

where $\mathbf{F}$ is the flux vector and $\mathbf{C}$ is an inhomogeneous term not containing derivatives (which often arises from the divergence operation in nonrectangular geometries). As described in Section 1.4, Eqn. (2.2.2.1) can also be written in a primitive variable fashion with a vector of dependent variables $\mathbf{Q}'$:

$$\frac{\partial \mathbf{Q}'}{\partial t} + \mathbf{A} \frac{\partial \mathbf{Q}'}{\partial x} + \mathbf{C}' = 0 \quad (2.2.2.2)$$

where $\mathbf{A}$ is an $n \times n$ matrix. The choice of vector $\mathbf{Q}'$ is not unique (whereas the choice of conserved variables is), and can in fact be equal to $\mathbf{Q}$. For our purposes we shall assume that $\mathbf{Q}'$ and $\mathbf{Q}$ are distinct. Moreover, the two systems are related according to the following expressions:

$$\frac{\partial \mathbf{Q}}{\partial t} = \mathbf{W} \frac{\partial \mathbf{Q}'}{\partial t} \quad (2.2.2.3)$$

$$\frac{\partial \mathbf{F}}{\partial x} = \mathbf{Y} \frac{\partial \mathbf{Q}'}{\partial x} \quad (2.2.2.4)$$

with,

$$W_{ij} = \frac{\partial Q_i}{\partial Q'_j} \quad (2.2.2.5)$$

$$Y_{ij} = \frac{\partial F_i}{\partial Q'_j} \quad (2.2.2.6)$$

and,

$$\mathbf{A} = \mathbf{W}^{-1} \mathbf{Y} \quad (2.2.2.7)$$

$$\mathbf{C}' = \mathbf{W}^{-1} \mathbf{C} \quad (2.2.2.8)$$

where $\mathbf{W}$ and $\mathbf{Y}$ are also $n \times n$ matrices.

Let $\mathbf{l}_i$ and $\mathbf{r}_i$ be the set of left (row) and right (column) eigenvectors of $\mathbf{A}$ satisfying:

$$\mathbf{l}_i \mathbf{A} = \lambda_i \mathbf{l}_i \quad (2.2.2.9)$$

$$\mathbf{A} \mathbf{r}_i = \lambda_i \mathbf{r}_i \quad (2.2.2.10)$$

Our focus is on hyperbolic systems, making the n eigenvalues of $\mathbf{A}$ real. We order the eigenvalues such that $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$. Utilizing a similarity transformation, a diagonal matrix $\mathbf{L}$ is obtained:

$$\mathbf{P} \mathbf{A} \mathbf{P}^{-1} = \mathbf{L} \quad (2.2.2.11)$$

where the rows of $\mathbf{P}$ are the left eigenvectors $\mathbf{l}_i$, the columns of $\mathbf{P}^{-1}$ are the right eigenvectors $\mathbf{r}_i$, and the matrix $\mathbf{L}$ is diagonal with $L_{ii} = \lambda_i$. Subsequently, multiplying Eqn. (2.2.2.2) by $\mathbf{P}$ gives:

$$\mathbf{P} \frac{\partial \mathbf{Q}'}{\partial t} + \mathbf{LP} \frac{\partial \mathbf{Q}'}{\partial x} + \mathbf{PC}' = 0 \quad (2.2.2.12)$$

Eqn. (2.2.2.12) can also be written in component form as:

$$\mathbf{l}_i \frac{\partial \mathbf{Q}'}{\partial t} + \lambda_i \mathbf{l}_i \frac{\partial \mathbf{Q}'}{\partial x} + \mathbf{l}_i \mathbf{C}' = 0 \quad (2.2.2.13)$$

Eqn. (2.2.2.12) is the “characteristic” equation corresponding to the original problem given by Eqn. (2.2.2.1) or (2.2.2.2). Let us define a new function $\mathbf{V}$ by:

$$dV_i = \mathbf{l}_i d\mathbf{Q}' + \mathbf{l}_i \mathbf{C}' dt \quad (2.2.2.14)$$

With this definition Eqn. (2.2.2.13) becomes:

$$\frac{\partial V_i}{\partial t} + \lambda_i \frac{\partial V_i}{\partial x} = 0 \quad (2.2.2.15)$$

Eqn. (2.2.2.15) represents a set of wave equations with characteristic velocities λ_i .

Each of the wave amplitudes V_i is constant along the curve C_i in the x - t plane defined

by $\frac{dx}{dt} = \lambda_i$, as we observed in Section 1.4 (that is, each V_i represent a Riemann

invariant having characteristic speed λ_i). Unfortunately, as alluded to in Section 1.6.1, Eqn. (2.2.2.14) can only be constructed if $\mathbf{A}$ and $\mathbf{C}$ are constant everywhere, or if no more than two differentials appear on the right hand side of Eqn. (2.2.2.14). If these conditions are not met, then the coefficients of Eqn. (2.2.2.14) must satisfy Pfaff's condition for the integrability of differential forms in order for the functions V_i to exist, a condition not met by the Navier-Stokes equations. Nevertheless, Eqn. (2.2.2.13) holds true independent of Eqn. (2.2.2.14). The formulation of the boundary conditions is then straightforward.

Let us assume that we wish to solve Eqn. (2.2.2.1) over the region $a \leq x \leq b$. As a consequence, this implies solving an initial boundary value problem, since both initial data in the region $a \leq x \leq b$ and time dependent boundary conditions at $x = a$ and $x = b$ are needed to ensure the problem is well-posed.

Outgoing waves (in one dimension) are those having $\lambda_i \leq 0$ at $x = a$, and/or $\lambda_i \geq 0$ at $x = b$. These waves depend only on information at and within the computational boundaries, and thus, those equations in the form of (2.2.2.13) denoting outgoing waves can be solved directly, or in another equivalent form. Incoming waves, on the other hand, are those given by $\lambda_i > 0$ at $x = a$, and/or $\lambda_i < 0$ at $x = b$. Unlike outgoing waves, incoming waves depend on data exterior to the boundary, and numerical approximations to Eqn. (2.2.2.13) not involving exterior data will be unstable. For time dependent problems, Hedstrom's (1978) non-reflecting boundary condition has the property of minimizing reflections caused by outgoing waves. This boundary condition, generalized to multidimensional problems and non-rectangular coordinates by Thompson (1986), can be mathematically expressed as:

$$\left. \frac{\partial V_i}{\partial t} \right|_{x=a,b} = 0 \quad (2.2.2.16)$$

In terms of the wave amplitude V_i of (2.2.2.14) we have:

$$\left(l_i \frac{\partial Q'}{\partial t} + l_i C' \right) \Big|_{x=a,b} = 0 \quad (2.2.2.17)$$

To articulate, Hedstrom's condition states that the amplitudes of the incoming waves are constant in time at the boundaries. As the presence of a wave is recognized by its change in amplitude, this statement is equivalent to there being no incoming waves at these boundaries.

The characteristic form of the equations and the non-reflecting boundary condition can then be easily combined as follows:

$$\left(l_i \frac{\partial Q'}{\partial t} + \mathcal{L}_i + l_i C' \right) \Big|_{x=a,b} = 0 \quad (2.2.2.18)$$

where,

$$\mathcal{L}_i = \begin{cases} \lambda_i l_i \frac{\partial Q'}{\partial t}, & \text{for outgoing waves} \\ 0, & \text{for incoming waves} \end{cases} \quad (2.2.2.19)$$

Eqn. (2.2.2.1) is solved at the interior points $a \leq x_i \leq b$, for $0 \leq i \leq I$, while the boundary equation (in the form of Eqn. (2.2.2.21) below) is solved at the boundary

points x_i , for $i < 0$ or $i > I$. Lastly, to obtain an equation written in terms of the conservative variables, we let $\mathcal{F}$ be the column vector the components of which are $\mathcal{F}_i$, and write:

$$\mathbf{P} \frac{\partial \mathbf{Q}'}{\partial t} + \mathcal{F} + \mathbf{P} \mathbf{C}' = 0 \quad (2.2.2.20)$$

which leads to,

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{W} (\mathbf{P}^{-1} \mathcal{F} + \mathbf{C}') = 0 \quad (2.2.2.21)$$

Table 2.2.2.1 shows the results from Thompson's first test case. The subscripts "s" and "u" denote shocked and un-shocked quantities respectively. This experiment involves a single shock wave starting at $x = 0.5$ and moving into a uniform stationary medium. Ahead of the shock we have $\rho = p = 1$ and $u = 0$. Thompson recognizes that a reflection is generated when an outgoing subsonic shock wave crosses the boundary. Moreover, the reflection takes the form of a constant amplitude perturbation to the post-shock solution, which travels inward from the boundary at the speed of sound relative to the moving fluid. Thus, as a convenient measure of reflection, Thompson uses the relative error in the pressure after the shock has crossed the boundary, defined as the difference between the numerical and analytical pressure values, divided by the analytical pressure.

Table 2.2.2.1 – The amount of error due to pressure wave reflection as a function of increasing Mach number, reproduced from Thompson (1986)

Mach number	Pressure ratio	Relative error (%)
M	$R = p_s / p_u$	$100 \times (p_{num} - p_{an}) / p_{an}$
0.50	2.504	0.08
0.60	3.096	0.19
0.70	3.891	0.33
0.80	5.000	0.48
0.90	6.635	0.68
0.92	7.058	0.71
0.94	7.524	0.74
0.95	7.775	0.77
0.96	8.040	0.78
0.97	8.319	0.81
0.98	8.614	0.82
0.99	8.926	0.82
0.995	9.089	0.80
1.00	9.257	0.77

2.2.3 NRBC OF RUDY AND STRIKWERDA

For simulations involving a subsonic outflow boundary, the Navier-Stokes equations require specification of one boundary condition. Common practice is to specify static pressure at such outflow boundaries:

$$p_{out} = p_{\infty} \quad (2.2.3.1)$$

Oliger *et al.* shows that this condition is well-posed and, in many instances, works very well computationally. Unfortunately, when one seeks steady state solutions to the Navier-Stokes equations, boundary condition (2.2.3.1) can introduce severe pressure wave disturbances back into the computational domain. Furthermore, the standing waves that they produce are damped at a very slow rate by numerical viscosity, hence inhibiting convergence to a steady state. This factor led Rudy and Strikwerda (1980) to reconsider the present state of non-reflecting boundary

conditions. In doing so, they focus mainly on the contributions outlined in Sections (2.2.1) and (2.2.2).

If we let $\bar{\rho}, \bar{u}, \bar{v}$, and $\bar{p}$ represent a steady state solution to the Euler equations and let $\tilde{\rho}, \tilde{u}, \tilde{v}$, and $\tilde{p}$ be perturbations about that state, then the linearized Navier-Stokes equations to first order in the perturbation variables are:

$$\begin{bmatrix} \tilde{\rho} \\ \tilde{u} \\ \tilde{v} \\ \tilde{p} \end{bmatrix}_t + \begin{bmatrix} \bar{u} & \bar{p} & 0 & 0 \\ 0 & \bar{u} & 0 & \bar{\rho}^{-1} \\ 0 & 0 & \bar{u} & 0 \\ 0 & \gamma \bar{p} & 0 & \bar{u} \end{bmatrix} \begin{bmatrix} \tilde{\rho} \\ \tilde{u} \\ \tilde{v} \\ \tilde{p} \end{bmatrix}_x + \begin{bmatrix} \bar{v} & 0 & \bar{p} & 0 \\ 0 & \bar{v} & 0 & 0 \\ 0 & 0 & \bar{v} & \bar{\rho}^{-1} \\ 0 & 0 & \gamma \bar{p} & \bar{v} \end{bmatrix} \begin{bmatrix} \tilde{\rho} \\ \tilde{u} \\ \tilde{v} \\ \tilde{p} \end{bmatrix}_y = 0 \quad (2.2.3.2)$$

Rudy and Strikwerda set out to compare their technique to that of Engquist and Majda (1977) and Hedstrom (1979). Unfortunately, the method of Engquist and Majda cannot be applied to non-linear equations (a clear disadvantage), which explains Rudy and Strikwerda's choice to use the linearized version of the equations.

As their region of interest, Rudy *et al.* use the half-plane $x \geq 0$. To ensure that the flow be subsonic at their boundary $x = L$, they require that:

$$0 < \bar{u} < \bar{c} = (\gamma \bar{\rho}^{-1} \bar{p})^{\frac{1}{2}} \quad (2.2.3.3)$$

Applying the first absorbing approximation of Engquist and Majda gives:

$$p - \bar{\rho} \bar{c} u = \tilde{p} - \bar{\rho} \bar{c} \tilde{u} \quad (2.2.3.4)$$

When this boundary condition is implemented for a problem in which all the steady state variables are known, convergence is much faster than if one were to simply apply boundary condition (2.2.3.1). Unfortunately, in general the steady state values of all the variables are not known a priori. A similar problem is encountered with Engquist and Majda's second absorbing approximation:

$$\frac{\partial}{\partial t}(\tilde{p} - \bar{\rho} \bar{c} \bar{u}) + c_1 \tilde{p} + c_2 \tilde{u} + c_3 \tilde{v} + c_4 \tilde{p} + c_5 \frac{\partial \tilde{v}}{\partial y} = 0 \quad (2.2.3.5)$$

where the coefficients c_i involve $\bar{\rho}$, $\bar{u}$, $\bar{v}$, and $\bar{p}$ and their spatial derivatives.

In the context of Eqn. (2.2.3.2), Hedstrom's NRBC amounts to:

$$\frac{\partial p}{\partial t} - \rho c \frac{\partial u}{\partial t} = 0 \quad (2.2.3.6)$$

Although this boundary condition requires no a priori knowledge of the steady state variables, the steady state solution is dependent on the initial data. In fact, the steady state pressure need not be equivalent to $\bar{p}$. Thus, Hedstrom's technique emphasizes the transient solution as opposed to the steady state one. The existence of these problems led Rudy and Strikwerda to formulate their boundary condition as:

$$\frac{\partial p}{\partial t} - \rho c \frac{\partial u}{\partial t} + \alpha(p - \bar{p}) = 0 \quad (2.2.3.7)$$

where α is a constant. The advantage in this case is that when a steady state solution is achieved, the outflow pressure will have the desired value. As one might expect, choosing an appropriate value of α can be cumbersome. Rudy *et al.* present a rigorous method for determining α when the linearized constant coefficient one-dimensional system of fluid flow equations is used. Also presented, are empirical studies involving two test problems governed by the two-dimensional compressible Navier-Stokes equations. The computational domain and boundary treatment for the first of these two problems is shown in Fig. 2.2.3.1. Outlined in this figure is the simulation initial and boundary conditions for a uniform subsonic flow in the x direction with an interior two-dimensional perturbation at $t = 0$. The subscript “int” refers to initial disturbances and the subscript “ ∞ ” is used to denote free-stream values. The grid consists of 21 nodes in both the x and y directions. In the second of these two test problems, the initial and boundary conditions remain the same, while the length in the y - grid direction is doubled, along with the corresponding node count.

Figure 2.2.3.1 – Computational domain and corresponding boundary conditions for Rudy *et al.*'s test case 1, reproduced from Rudy *et al.* (1980)

In the context of these simulations, three boundary conditions for density are studied.

The first of these is the current modification, the form of which is:

$$\rho_{i,j}^{n+1} = \frac{p_{i,j}^{n+1}}{(\gamma - 1)T_{i,j}^{n+1}} \quad (2.2.3.8)$$

where $p_{i,j}$ is computed from the finite difference approximation to Eqn. (2.2.3.7).

The second boundary condition is that of constant pressure at the outflow boundary.

In this case $\rho_{i,j}^{n+1}$ is computed from Eqn. (2.2.3.8) with $p_{i,j}^{n+1} = p_\infty$. The final boundary condition involves the extrapolation of density, given by:

$$\rho_{i,j}^{n+1} = \rho_{i-1,j}^{n+1} \quad (2.2.3.9)$$

It should be noted that the constant-pressure boundary condition is obtained in the limit of Eqn. (2.2.3.8) as $\alpha \rightarrow \infty$. Experiments were carried out to observe the effect of α on the convergence rate using three different values of the free-stream Mach number. The results for the two test cases are shown in Figs. 2.2.3.2 and 2.2.3.3.

Figure 2.2.3.2 – Effect of α on the convergence rate for Rudy *et al.*'s test case 1, reproduced from Rudy *et al.* (1980)

Figure 2.2.3.3 – Effect of α on the convergence rate for Rudy *et al.*'s test case 2, reproduced from Rudy *et al.* (1980)

2.3 NON-REFLECTING BUFFER ZONES (NRBZs)

Now that we have examined non-reflecting boundary conditions in some detail, we turn to the second major type of boundary treatment: non-reflecting buffer zones.

2.3.1 BUFFER ZONE TECHNIQUE OF TA'ASAN AND NARK

Cerjan *et al.* (1985), Sochacki *et al.* (1987), and Hanson and Petschek (1976) present attenuating “buffer-domain” techniques where the amplitudes of the displacement are gradually reduced when approaching the boundary. Alternatively, rather than attenuating the waves, Kishoni and Ta'asan (1994), and later on, Ta'asan and Nark (1995), modify the equations of motion within the buffer zone in such a way that their direction of propagation is biased toward the outer boundary until no backward

propagation is possible. This technique was first developed by Streett (1986) in the context of spectral collocation discretization methods. In general, the viscous terms responsible for streamwise-elliptic behavior are multiplied by a smooth attenuation function which goes to zero at the outer edge of the buffer zone. At the same time, to avoid upstream feedback due to the acoustic or pressure terms, the imposed base- or mean-flow is modified in the buffer zone so that the total streamwise velocity at the outer edge of this buffer domain is supersonic (which is equivalent to saying that all waves are convected out of the domain at this boundary) (Macaraeg (1992)).

To see how Ta'asan *et al.* apply their buffer zone technique, consider again Eqn. (2.2.2.2):

$$\frac{\partial Q'}{\partial t} + A \frac{\partial Q'}{\partial x} + C' = 0$$

In general, the buffer zone equations are constructed by changing the $\frac{\partial}{\partial t}$ term to:

$$\frac{\partial}{\partial t} + \mathbf{U}' \cdot \nabla \quad (2.3.1.1)$$

Here $\mathbf{U}'$ is a parameter that is used to impose an artificial velocity field within the buffer zone. The choice of $\mathbf{U}'$ is to be made such that, in the vicinity of the outflow boundary all waves are convected outwards. Moreover, $\mathbf{U}'$ is set to zero within the computational domain, since no modification of the equations is carried out on this portion of the grid. As the mean velocity field is artificially changed in the vicinity of the boundary, the efficiency of the numerical method is adversely affected. This diminished efficiency is caused by the need for reducing the time step size within the

buffer zone in order to counteract the increased velocity, and hence, to ensure a stable computation. These particular notions follow from what we have seen in Section 1.5.1 regarding the stability of discretization schemes. In order to alleviate this problem, the acoustic or pressure-wave speed is lowered within the buffer zone. The accomplishment of this is facilitated by multiplying the true convective operator by a variable μ . The value of μ is set to 1 on the interior domain (where no alteration of the equations takes place) and is decreased to a specified value at the outer edge of the buffer zone. Thus the full, modified version of Equation (2.2.2.2) in the buffer zone is:

$$\frac{\partial \mathbf{Q}'}{\partial t} + (\mathbf{U}' + \mu \mathbf{A}) \frac{\partial \mathbf{Q}'}{\partial x} + \mathbf{C} = 0 \quad (2.3.1.2)$$

Specific values of $\mathbf{U}' = \begin{bmatrix} U' & 0 & 0 \\ 0 & U' & 0 \\ 0 & 0 & U' \end{bmatrix}$ and $\mu = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & \mu \end{bmatrix}$ can vary depending on

the nature of each problem. In order to gain a greater appreciation for the relationship between $\mathbf{U}'$ and μ , a stability analysis for the system of equations considered by Ta'asan and Nark (1995) is given below.

In this work, the two researchers utilized an absorbing buffer zone for the following linear problem describing acoustic wave propagation:

$$\frac{\partial p}{\partial t} + \mathbf{U} \cdot \nabla p + c^2 \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) = 0 \quad (2.3.1.3)$$

$$\frac{\partial u}{\partial t} + \mathbf{U} \cdot \nabla u + \frac{\partial p}{\partial x} = 0 \quad (2.3.1.4)$$

$$\frac{\partial v}{\partial t} + \mathbf{U} \cdot \nabla v + \frac{\partial p}{\partial y} = 0 \quad (2.3.1.5)$$

where p is the pressure, u and v are the x and y components of the perturbation velocity respectively, $\mathbf{U}$ is the mean velocity field, and c is the acoustic wave speed. Gaussian signals of different widths can be chosen for initial conditions. Once again, the buffer zone equations are constructed by first making changes according to Eqn. (2.3.1.1), yielding:

$$\frac{\partial p}{\partial t} + (\mathbf{U}' + \mathbf{U}) \cdot \nabla p + c^2 \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) = 0 \quad (2.3.1.6)$$

$$\frac{\partial u}{\partial t} + (\mathbf{U}' + \mathbf{U}) \cdot \nabla u + \frac{\partial p}{\partial x} = 0 \quad (2.3.1.7)$$

$$\frac{\partial v}{\partial t} + (\mathbf{U}' + \mathbf{U}) \cdot \nabla v + \frac{\partial p}{\partial y} = 0 \quad (2.3.1.8)$$

After the appropriate application of the buffer zone technique, the full, modified equations in the buffer zone are:

$$\frac{\partial p}{\partial t} + (\mathbf{U}' + \mathbf{U}) \cdot \nabla p + \mu c^2 \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) = 0 \quad (2.3.1.9)$$

$$\frac{\partial u}{\partial t} + (\mathbf{U}' + \mathbf{U}) \cdot \nabla u + \mu \frac{\partial p}{\partial x} = 0 \quad (2.3.1.10)$$

$$\frac{\partial v}{\partial t} + (\mathbf{U}' + \mathbf{U}) \cdot \nabla v + \mu \frac{\partial p}{\partial y} = 0 \quad (2.3.1.11)$$

No indication is given regarding the exact finite-differencing scheme used in this work. However, we wish to shed some light on the connection between stability and the parameters $\mathbf{U}'$ and μ . In order to do so, we assume that Eqns. (2.3.1.9) –

(2.3.1.11) are discretized using Lax's method. Furthermore, assuming solutions of the form:

$$\begin{aligned} p &= \hat{p}e^{i(kx+ly)} \\ u &= \hat{u}e^{i(kx+ly)} \\ v &= \hat{v}e^{i(kx+ly)} \end{aligned} \quad (2.3.1.12)$$

the discrete representation of the equations becomes:

$$\begin{aligned} \hat{p}^{n+1} &= \cos(k\Delta x)\hat{p}^n - \tilde{U}'\frac{\Delta t}{\Delta x}(i\sin(k\Delta x))\hat{p}^n - \tilde{V}'\frac{\Delta t}{\Delta y}(i\sin(l\Delta y))\hat{p}^n \\ &\quad - \mu c^2 \Delta t \left[\frac{i\sin(k\Delta x)}{\Delta x}\hat{u}^n + \frac{i\sin(l\Delta y)}{\Delta y}\hat{v}^n \right] \end{aligned} \quad (2.3.1.13)$$

$$\begin{aligned} \hat{u}^{n+1} &= \cos(k\Delta x)\hat{u}^n - \tilde{U}'\frac{\Delta t}{\Delta x}(i\sin(k\Delta x))\hat{u}^n - \tilde{V}'\frac{\Delta t}{\Delta y}(i\sin(l\Delta y))\hat{u}^n \\ &\quad - \mu \frac{\Delta t}{\Delta x}(i\sin(k\Delta x))\hat{p}^n \end{aligned} \quad (2.3.1.14)$$

$$\begin{aligned} \hat{v}^{n+1} &= \cos(k\Delta x)\hat{v}^n - \tilde{U}'\frac{\Delta t}{\Delta x}(i\sin(k\Delta x))\hat{v}^n - \tilde{V}'\frac{\Delta t}{\Delta y}(i\sin(l\Delta y))\hat{v}^n \\ &\quad - \mu \frac{\Delta t}{\Delta x}(i\sin(l\Delta y))\hat{p}^n \end{aligned} \quad (2.3.1.15)$$

where $\tilde{U}' = U' + U$ and $\tilde{V}' = V' + V$ with $U' = (U', V')$ and $U = (U, V)$, and $\mu = \mu$.

In matrix form, this system can be written as:

$$\begin{bmatrix} \hat{p}^{n+1} \\ \hat{u}^{n+1} \\ \hat{v}^{n+1} \end{bmatrix} = \tilde{A}' \begin{bmatrix} \hat{p}^n \\ \hat{u}^n \\ \hat{v}^n \end{bmatrix} \quad (2.3.1.16)$$

where,

$$\tilde{\mathbf{A}}' = \begin{bmatrix} \cos(k\Delta x) - \tilde{U}' \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) & -\mu c^2 \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) & -\mu c^2 \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) \\ -\tilde{V}' \frac{\Delta t}{\Delta y} (i \sin(l\Delta y)) & \cos(k\Delta x) - \tilde{U}' \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) & 0 \\ -\mu \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) & -\tilde{V}' \frac{\Delta t}{\Delta y} (i \sin(l\Delta y)) & \cos(k\Delta x) - \tilde{U}' \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) \\ -\mu \frac{\Delta t}{\Delta y} (i \sin(l\Delta y)) & 0 & -\tilde{V}' \frac{\Delta t}{\Delta y} (i \sin(l\Delta y)) \end{bmatrix} \quad (2.3.1.17)$$

Stability of this finite-differencing scheme is maintained provided $|\lambda_{\max}| \leq 1$, where

$\lambda_{\max}$ is the largest eigenvalue of $\tilde{\mathbf{A}}'$. The eigenvalues of $\tilde{\mathbf{A}}'$ are:

$$\lambda_1 = \cos(k\Delta x) - \tilde{U}' \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) - \tilde{V}' \frac{\Delta t}{\Delta y} (i \sin(l\Delta y)) \quad (2.3.1.18)$$

$$\lambda_2 = \cos(k\Delta x) - \tilde{U}' \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) - \tilde{V}' \frac{\Delta t}{\Delta y} (i \sin(l\Delta y)) + i\mu c \Delta t \left[\frac{\sin^2(k\Delta x) + \sin^2(l\Delta y)}{\Delta x \Delta y} \right]^{\frac{1}{2}} \quad (2.3.1.19)$$

$$\lambda_3 = \cos(k\Delta x) - \tilde{U}' \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) - \tilde{V}' \frac{\Delta t}{\Delta y} (i \sin(l\Delta y)) - i\mu c \Delta t \left[\frac{\sin^2(k\Delta x) + \sin^2(l\Delta y)}{\Delta x \Delta y} \right]^{\frac{1}{2}} \quad (2.3.1.20)$$

From this we obtain the following stability condition for the Lax method applied to

Eqns. (2.3.1.9) – (2.3.1.11):

$$\left| \cos(k\Delta x) - (U' + U) \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) - (V' + V) \frac{\Delta t}{\Delta y} (i \sin(l\Delta y)) + i\mu c \Delta t \left[\frac{\sin^2(k\Delta x) + \sin^2(l\Delta y)}{\Delta x \Delta y} \right]^{\frac{1}{2}} \right| \leq 1 \quad (2.3.1.21)$$

Upon examining Eqn. (2.3.1.21), the balance between U' and μ becomes more evident. The presence of $U' = (U', V')$, both in Eqns. (2.3.1.9) – (2.3.1.11) and Eqn. (2.3.1.21), visibly affects the mean flow, U . In particular, as U' increases it adversely affects the time-step limit dictated by Eqn. (2.3.1.21). Thus, to offset the requirement for smaller time-steps in order to maintain stability, the effective acoustic wave speed, μc , is accordingly decreased throughout the buffer zone.

In order to analyze the movement of waves at the end of the buffer zone, we again consider solutions of the form:

$$\begin{pmatrix} \hat{p} \\ \hat{u} \\ \hat{v} \end{pmatrix} e^{i(\omega t + \mathbf{k} \cdot \mathbf{x})} \quad (2.3.1.22)$$

Here $\mathbf{k} = (k, l)^T$ and $\mathbf{x} = (x, y)^T$, where ω can be either positive or negative.

Substituting this solution into Eqns. (2.3.1.6) – (2.3.1.8), we obtain:

$$\begin{pmatrix} \omega + \mathbf{a} \cdot \mathbf{k} & c^2 k & c^2 l \\ k & \omega + \mathbf{a} \cdot \mathbf{k} & 0 \\ l & 0 & \omega + \mathbf{a} \cdot \mathbf{k} \end{pmatrix} \begin{pmatrix} \hat{p} \\ \hat{u} \\ \hat{v} \end{pmatrix} = 0 \quad (2.3.1.23)$$

where $\mathbf{a} = U' + U$. In order for this equation to have nonzero solutions, the determinant must be zero. That is:

$$(\omega + \mathbf{a} \cdot \mathbf{k})[(\omega + \mathbf{a} \cdot \mathbf{k})^2 - c^2(k^2 + l^2)] = 0 \quad (2.3.1.24)$$

Eqn. (2.3.1.24) defines the different possible propagating waves. As mentioned earlier, the choice of $\mathbf{a}$ is to be made such that in the vicinity of the outer boundary of the buffer zone all waves are outgoing. The solutions to Eqn. (2.3.1.24) are:

$$\omega = -\mathbf{a} \cdot \mathbf{k} \quad (2.3.1.25)$$

$$\omega = -\mathbf{a} \cdot \mathbf{k} \pm c |\mathbf{k}| \quad (2.3.1.26)$$

Eqns. (2.3.1.25) and (2.3.1.26) correspond to the following group velocities:

$$\mathbf{V} = \mathbf{a} \quad (2.3.1.27)$$

$$\mathbf{V} = \mathbf{a} \pm c \frac{\mathbf{k}}{|\mathbf{k}|} \quad (2.3.1.28)$$

In order for the waves to move towards the outflow boundary, $\mathbf{a}$ must be defined as follows:

$$\mathbf{a} = \gamma \mathbf{n} \quad \gamma > c \quad (2.3.1.29)$$

where $\mathbf{n}$ is the outward-pointing normal to the boundary. Finally, to ensure that Eqn. (2.3.1.29) guarantees only outgoing waves, we check if $\mathbf{V} \cdot \mathbf{n} > 0$. Specifically:

$$\begin{aligned} & \mathbf{V} \cdot \mathbf{n} \\ &= \left(\mathbf{a} \pm c \frac{\mathbf{k}}{|\mathbf{k}|} \right) \cdot \mathbf{n} \end{aligned}$$

$$\begin{aligned}
&= \left(\gamma \mathbf{n} \pm c \frac{\mathbf{k}}{|\mathbf{k}|} \right) \cdot \mathbf{n} \\
&= (\gamma \mathbf{n}) \cdot \mathbf{n} \pm \left(\frac{c \mathbf{k}}{|\mathbf{k}|} \right) \cdot \mathbf{n} \\
&= \gamma |\mathbf{n}| \pm \left(\frac{c \mathbf{k}}{|\mathbf{k}|} \right) \cdot \mathbf{n}
\end{aligned} \tag{2.3.1.30}$$

From this last expression we see that in order for $\mathbf{V} \cdot \mathbf{n} > 0$, we must have $\gamma > c$. A similar argument holds for $\mathbf{V} = \mathbf{a}$. Note that, as c is the acoustic wave speed or speed of sound, this result agrees with our earlier claim that the buffer zone technique accelerates the flow to supersonic speeds at the outer edge of the buffer zone.

One possible definition for the artificial velocity vector $\mathbf{U}'$ is:

$$U' = \beta(x - x_e)^m, \quad V' = 0, \quad \text{on the } x = \text{const. sides} \tag{2.3.1.31}$$

$$V' = \beta(y - y_e)^m, \quad U' = 0, \quad \text{on the } y = \text{const. sides} \tag{2.3.1.32}$$

Here (x_e, y_e) are the values of (x, y) at the edge of the interior domain, as can be seen in Fig. 2.3.1.1. It should be noted that only a buffer zone in the x -direction is shown in Fig. 2.3.1.1; however, the treatment in the y -direction is analogous.

Figure 2.3.1.1 – A schematic of the buffer zone computational domain used in the work of Ta'asan and Nark (1995)

When both velocity components are non-zero, the following treatment is carried out there:

$$U' = \beta(x - x_e)^m, \quad V' = \beta(y - y_e)^m \quad (2.3.1.33)$$

The exponent m dictates how quickly U' increases as we traverse through the buffer zone. Clearly as m increases we experience a smoother transition between the interior domain and the buffer zone, while U' rapidly increases as we approach the outermost edge. Intuitively, it appears that a smooth transition is desirable and in fact, some numerical experimentation has supported this hypothesis. In particular, based on preliminary trials Ta'asan and Nark arrive at a value of $m = 8$ as the optimal choice for their work. For the present, this remains an area that might benefit from further parametric studies. The constant β is used to impose the correct value of the normal

component of the velocity at the outer edge of the buffer zone, ensuring supersonic flow there. In cases where the maximum interior velocity is not known a priori, the selection of U' (or β) and μ (or η) becomes empirical in nature. More will be said about this issue within the numerical results of Chapter 4. For the problem studied by Ta'asan and Nark, the final converged velocity profile is known before hand. Consequently, a value of 1.3 times this maximum interior velocity guarantees supersonic flow at the outer edge of the buffer zone. Furthermore, the researchers chose to decrease μ from 1 to 0.1 in the buffer zone, which is accomplished by using the following relations:

$$\mu = 1 - \eta(x - x_e)^m \quad \text{on the } x = \text{const. side} \quad (2.3.1.34)$$

$$\mu = 1 - \eta(y - y_e)^m \quad \text{on the } y = \text{const. side} \quad (2.3.1.35)$$

$$\mu = 1 - \eta((x - x_e)^m + (y - y_e)^m) \quad \text{at the corners} \quad (2.3.1.36)$$

η , which is similar in function to β , is a constant which allows the proper value of μ to be set at the edge of the buffer zone. Unlike the value β , which must be chosen large enough to permit only outgoing waves at the outer edge of the buffer zone, there is more flexibility in choosing η , provided the resulting U' and μ terms maintain stability of the finite-differencing scheme. That is, for a particular U' and time-step Δt , μ must be chosen to decrease gradually throughout the buffer zone so that its value at the outer edge is large enough to counteract the increased velocity. The simultaneous action of μ has as its main function, the maintenance of numerical stability. As we shall see in Chapter 4, if the time-step is chosen small enough, μ (or η) is not required for this task, even as β reaches its threshold value. The

ramifications of this result are discussed in Chapter 4, along with further insight into the connection between U' and μ .

Ta'asan and Nark apply their buffer zone method to the following problem with initial conditions corresponding to zero perturbation velocity and a pressure that is expressed as a Gaussian distribution imposed on a mean pressure field:

$$p = p_o + p' e^{-\alpha r^2} \quad (2.3.1.37)$$

$$u = 0 \quad (2.3.1.38)$$

$$v = 0 \quad (2.3.1.39)$$

where p_o is the mean pressure, p' the amplitude of the imposed pressure distribution, α an arbitrary constant which controls the width of the Gaussian, and $r^2 = (x - x_o)^2 + (y - y_o)^2$ with (x_o, y_o) the center of the Gaussian. The mean pressure value and velocity field is specified and the results, shown in Figs. 2.3.1.2 and 2.3.1.3, appear to agree with Ta'asan *et al*'s predictions. Namely, as the buffer zone size increases, the amount of reflection, as measured by a "reflection factor",

$\left(\frac{E}{E_o} \right)$, is reduced. The numerator, E , of this reflection factor is defined by:

$$E = \sqrt{\frac{\sum (p_{i,j}^B - p_{i,j}^I)^2}{N}} \quad (2.3.1.40)$$

where $p_{i,j}^B$ is the calculated pressure within the buffer zone, $p_{i,j}^I$ corresponds to the calculated pressure for the infinite domain, and N is the number of points in the interior domain. This value is then normalized by E_o , which is defined as:

$$E_0 = \sqrt{\frac{\sum (p_{i,j}^G - p_a)^2}{N}} \quad (2.3.1.41)$$

Here, $p_{i,j}^G$ is the pressure in the initial Gaussian and p_a is the ambient pressure. In this way, it is probable that the same numerical errors will be present in both simulations (infinite and finite domain), thus leaving reflection as the main source of solution inaccuracies.

Figure 2.3.1.2 – An illustration of outflow boundary reflection as a function of buffer zone size, reproduced from Ta'asan and Nark (1995)

Figure 2.3.1.3 – An illustration of outflow boundary reflection as a function of (Buffer Zone Size)/(Gaussian Width), reproduced from Ta'asan and Nark (1995)

Among the most notable aspects not discussed within this study, are computational time issues and node count optimization within the buffer zone. Studies involving both of these parameters are carried out in the current work.

Several variants of the buffer zone technique first introduced by Streett (1986) have been used in flow simulations. For example, Colonius, Lele, and Moin (1993) use buffer zones in which the solutions are filtered. Perhaps, the most common methods are those that are based on the technique of Berenger (1994), known as the perfectly-matched-layer technique (PML), described in the next section.

2.3.2 PML METHODS OF BERENGER AND HAYDER, HU, AND HUSSAINI

In PML, the “absorbing-layer” equations are obtained by splitting the governing equations in the coordinate directions and introducing absorption coefficients in each split equation. Following Berenger, whose work focuses mainly on the equations of

electromagnetics, Hu (1996) uses a similar technique to study the linearized Euler equations, and provides analytic results for uniform flow problems.

Using the concept of operator splitting, Hayder *et al.* (1997) study a slight variant of the PML method applied to the Euler equations in the context of shock-vortex interactions, plane free shear flow, and axisymmetric jet flows. In order to see how the PML method is implemented, consider the two-dimensional Euler equations written in primitive variable form:

$$\text{continuity:} \quad \frac{\partial \rho}{\partial t} + u \frac{\partial \rho}{\partial x} + v \frac{\partial \rho}{\partial y} = -\rho \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \quad (2.3.2.1)$$

$$\text{x-momentum:} \quad \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = -\frac{1}{\rho} \frac{\partial p}{\partial x} \quad (2.3.2.2)$$

$$\text{y-momentum:} \quad \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} \quad (2.3.2.3)$$

$$\text{energy:} \quad \frac{\partial p}{\partial t} + u \frac{\partial p}{\partial x} + v \frac{\partial p}{\partial y} = -\gamma p \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \quad (2.3.2.4)$$

Within the absorbing layer, the equations are split into a locally one-dimensional set and artificial damping terms are added. For example, after operator splitting and the addition of these damping terms, the energy equation becomes:

$$\frac{\partial p_1}{\partial t} = -u \frac{\partial p}{\partial x} - \gamma p \frac{\partial u}{\partial x} - \sigma_x p_1 \quad (2.3.2.5)$$

$$\frac{\partial p_2}{\partial t} = -v \frac{\partial p}{\partial y} - \gamma p \frac{\partial v}{\partial y} - \sigma_y p_2 \quad (2.3.2.6)$$

These represent the fundamental equations dealt with in the absorbing layer. Here, u and v are velocity components in the x and y directions, and $p = p_1 + p_2$. Locally one-dimensional equations for the other variables are obtained in a similar fashion.

The values of the damping factors, σ_x and σ_y , are given in Fig. 2.3.2.1:

Absorbing Layer		$\sigma_x = 0$ $\sigma_y > 0$
		$\sigma_x > 0$ $\sigma_y > 0$
<div style="text-align: center;"> </div>		
Interior Domain $\sigma_x = 0$ $\sigma_y = 0$		
		Absorbing Layer $\sigma_x > 0$ $\sigma_y = 0$

Figure 2.3.2.1 – A graphical summary of the PML method, reproduced from Hayder *et al.* (1997)

In regards to Hayder *et al.*'s shock-vortex interaction transient simulations, the computational domain is constructed such that the shock serves as a boundary on the left with an outflow boundary on the right, and the two remaining y -direction boundaries being periodic. A fourth-order Runge-Kutta scheme is used for time-integration of the equations, and the spatial derivatives are discretized with a compact sixth-order scheme. An absorbing layer is constructed so that it abuts the right outflow boundary, and only in the x -direction. As with Ta'asan and Nark's experiments, in order to measure the contamination due to reflection, the results are compared with a reference solution obtained by computing the flow in a much larger domain with the same spatial and temporal resolution. More specifically, a root-

mean-square (rms) error (E) for pressure is used to quantify this reflection according to the following formula:

$$E = \frac{100}{|p'_{\max}|} \sqrt{\frac{\sum_{j=1}^N (p' - p)^2}{N}} \quad (2.3.2.7)$$

where p' is the pressure from the reference solution, $|p'_{\max}|$ is its maximum amplitude, p is the calculated pressure, and N is the number of grid points in the y -direction. E measures the numerical error in the solution, which includes both direct and induced errors due to the interaction of residual reflections from the outflow boundary with the flow and the shock. It should be noted that this error is not computed over the entire interior domain but at the ordinate four grid points upstream of the interface between the computational domain and the absorbing layer.

Figs. 2.3.2.2 and 2.3.2.3 show E as a function of the layer thickness, while at the same time, confirm two results regarding buffer zone techniques that are widely accepted and are at the same time quite intuitive. In particular, as the number of nodes within the “absorbing layer” increases, accuracy is improved, of course at the expense of greater CPU time per iteration. Unfortunately, very few have sought to optimize this selection, and as we shall see in Chapter 4, it appears that a certain maximum exists for the number of nodes to be included within the buffer zone, after which CPU time increases with no observable benefits.

Secondly, (which is a problem, generally speaking, in computational fluid dynamics) non-linear effects significantly increase the difficulty involved in numerical

computation. The results depicted in Fig. 2.3.2.3 for the shock-vorticity wave interaction problem required a domain three times larger than that of the free shear layer trial, in which case, the use of the linearized Euler equations in Cartesian coordinates was employed. Moreover, a greater node count within the absorbing layer was needed for the latter.

Experiments typically involved node counts within this layer varying between 10 and 50. After 3000 time steps of the non-linear case, the percent error in pressure fluctuation for the 5.4% and 9% node counts within the absorbing layer are 3.5% and 4% respectively.

Figure 2.3.2.2 – The percentage error in pressure fluctuations for varying node counts within the absorbing layer for numerical simulations of a free-shear layer, reproduced from Hayder *et al.* (1997)

Figure 2.3.2.3 – The percentage error in pressure fluctuations for varying node counts within the absorbing layer for numerical simulations of a shock-vorticity wave interaction problem, reproduced from Hayder *et al.* (1997)

The fundamental difference between the modification suggested in the PML method and that of Ta'asan and Nark's, is that the latter's damping term is based on gradients in space. Ta'asan and Nark's treatment of the equations physically "accelerates" the flow in the buffer region, while for PML, the very opposite takes place. Moreover, since the technique of Ta'asan *et al.* sets out to make outward propagating waves supersonic, it is hypothesized that fewer nodes are required within their buffer zone, and perhaps in this way, the application of the buffer zone technique is less sensitive to the nature of the problem.

In fact, there is a strong body of evidence suggesting that the number of nodes required in an absorbing layer is dependent on factors such as Mach number and flow direction. For the buffer zone technique of Ta'asan and Nark, this does not appear to be the case. The number of nodes for such a technique has greater dependency on the

choice of computational stencil. More will be said about this issue in Chapter 4. In regards to both methods, the modifications taking place within the buffer or absorbing layer are physically meaningless.

2.4 MOTIVATION FOR THE CURRENT STUDY

As it stands, very little work has been published dealing with the buffer zone technique originally proposed by Streett (1986). In fact, the paper itself in which Streett introduces this concept makes no aspect of the technique explicit to reader (Streett, 1986). Subsequent work has been only slightly clearer. Ta'asan and Nark's (1995) contribution, although demonstrating the increase in efficiency of the technique as buffer width increases, also fails to address the selection of crucial parameters and their optimal values. Aside from their paper, very little is even mentioned about the buffer zone technique of Streett, including within Giovoli's (1991) review paper. Although absorbing techniques, such as those of Berenger and Hayder *et al.*, have been regarded as computationally expensive (e.g. Klemp and Lilly (1978) and Israeli and Orzag (1981)), this has not been the case with techniques that accelerate the flow through a similar zone, such as the technique of Ta'asan and Nark (1995). These reasons, combined with the demonstrated theoretical of the buffer zone technique, makes it a very attractive possibility for numerical implementation. Seeing as, very little has been documented regarding optimal buffer zone parameters, buffer widths, and node counts within the buffer zone, the present work focuses on the technique of Ta'asan *et al.* in an attempt to further reduce the empiricism associated with the choice of buffer zone parameters and bring to light many of the key issues involved in their implementation.

In order to study this technique more efficiently, the *EE_Solver* algorithm has been developed for numerical simulations. Before presenting the results of these simulations, we dedicate the next chapter to describing this algorithm in greater detail.

3.0 THE GOVERNING EQUATIONS SOLVED BY *EE_SOLVER* – MOTIVATION AND ADVANTAGES

Ideally, the implementation of buffer zone techniques (as with the majority of methods used in numerical algorithms) into full Navier-Stokes solvers should be free of as much empiricism as possible. As noted earlier, until this point the introduction of parameters such as U' and μ in Ta'asan *et al.*'s buffer zone method or σ in Hayder *et al.*'s damping version is done without complete knowledge of the physical ramifications. To date, no optimization studies have been performed to determine whether modifications to the governing equations in such buffer zones may be done in a more efficient manner. Given the simultaneous action of several numerical components within a typical full Navier-Stokes solver, the inclusion of buffer zone treatments risk being cumbersome. Moreover, there is no guarantee that such modifications will provide sufficient numerical benefit. If our aim then is to study and hopefully optimize the buffer zone technique of Ta'asan *et al.*, a full Navier-Stokes solver would not provide the optimal setting for such an objective.

The aforementioned reasons represent the chief motivation for our use of the non-linear Euler equations in one spatial dimension, both in their characteristic and non-characteristic forms. This proves to be doubly beneficial since the non-characteristic form may be considered to be a simplified, but insightful adaptation of the full Navier-Stokes equations, while modifications to the equations in their characteristic form can easily be explained from a physical point of view. Lastly, the fact that viscous effects are often assumed to be negligible at outflow boundaries is yet further justification for the use of these equations instead of their full Navier-Stokes counterparts.

To facilitate this study, a numerical algorithm, denoted by *EE_Solver*, was developed and used by the author to solve this equation set. The details of this algorithm constitute the major component of this chapter, while the results of simulations conducted with *EE_Solver* are presented in Chapter 4.

Thus, we begin by presenting the buffer zone modification to the Euler equations, both in characteristic and non-characteristic form. The two modified sets of equations are solved in the subcomponents of *EE_Solver*, referred to as *CharSolver* and *NonCharSolver*, respectively (Appendix C). As outlined in Section 1.7, within *EE_Solver*, space derivatives are discretized with centered differencing, with appropriate modifications at computational boundaries. The discrete equations are then integrated in time using a second-order two-stage Runge-Kutta technique, described in Section 3.2.

3.1 THE BUFFER ZONE TECHNIQUE APPLIED TO THE ONE DIMENSIONAL EULER EQUATIONS

In Section 1.4 we outlined the derivation from the non-characteristic form of the Euler equations to the characteristic form. Following this same rationale, we now set out to derive the characteristic form of the equations including the buffer zone modification. The obvious starting point is then the modified form of the non-characteristic equations:

$$\frac{\partial Q'}{\partial t} + (U' + \mu A') \frac{\partial Q'}{\partial x} = 0 \quad (3.1.1)$$

Here the matrix U' is defined as $U'I$, where I is the identity matrix, and

$U' = \beta(x - x_e)^m$. The precise form of U' will be discussed shortly. It should be noted that we chose not to modify the equation set originally referred to as the non-characteristic form, namely Eqn. (1.4.1). Rather, our starting point was its linearized form, Eqn. (1.4.9). If the former had been carried out, the following equation would have been obtained:

$$\frac{\partial Q}{\partial t} + U' \frac{\partial Q}{\partial x} + \mu \frac{\partial E}{\partial x} = 0 \quad (3.1.2)$$

Consequently, Eqn. (3.1.1) simply becomes an alternative form of Eqn. (3.1.2), and so either starting point will suffice.

Another important aspect to point out is that the non-characteristic and characteristic forms of the equations are solved slightly differently within the *EE_Solver* algorithm. For the non-characteristic case, all derivatives are appropriately discretized and marched in time according to the Runge-Kutta scheme of Section 3.2. The three equations are solved for u , a , and ρ followed by the use of the ideal gas law to find p . The characteristic form, as we shall see below, receives a somewhat different treatment.

Following the derivation of Section 1.4, from Eqn. (3.1.1) we arrive at the following system of equations:

$$\frac{\partial Q''}{\partial t} + \tilde{D} \frac{\partial Q''}{\partial x} = 0 \quad (3.1.3)$$

where,

$$\tilde{\mathbf{D}} = \begin{bmatrix} U' + \mu u & 0 & 0 \\ 0 & U' + \mu(u + a) & 0 \\ 0 & 0 & U' + \mu(u - a) \end{bmatrix} \quad (3.1.4)$$

Expanding the vector equation given by (3.1.3) results in:

$$\frac{\partial}{\partial t} \int \left(d\rho - \frac{dp}{a^2} \right) + [U' + \mu u] \frac{\partial}{\partial x} \int \left(d\rho - \frac{dp}{a^2} \right) = 0 \quad (3.1.5)$$

$$\frac{\partial}{\partial t} \left(u + \frac{2a}{\gamma - 1} \right) + [U' + \mu(u + a)] \frac{\partial}{\partial x} \left(u + \frac{2a}{\gamma - 1} \right) = 0 \quad (3.1.6)$$

$$\frac{\partial}{\partial t} \left(u - \frac{2a}{\gamma - 1} \right) + [U' + \mu(u - a)] \frac{\partial}{\partial x} \left(u - \frac{2a}{\gamma - 1} \right) = 0 \quad (3.1.7)$$

This system for the Riemann invariants $\int \left(d\rho - \frac{dp}{a^2} \right)$, $\left(u + \frac{2a}{\gamma - 1} \right)$, and $\left(u - \frac{2a}{\gamma - 1} \right)$, is

analogous to that given by Eqns. (1.4.17) – (1.4.19). Eqns. (3.1.6) and (3.1.7) are solved in *CharSolver* as a system of two equations in two unknowns, namely u and a .

The next variable to be solved for is the density, which is obtained by rearranging the discretized version of Eqn. (3.1.5) as given below:

$$\frac{\rho_i^{n+1} - \rho_i^n}{\Delta t} - \frac{1}{(a_i^n)^2} \left[\frac{p_i^{n+1} - p_i^n}{\Delta t} \right] = -(U' + \mu u_i^n) \left\{ \frac{\rho_{i+1}^n - \rho_{i-1}^n}{2\Delta x} - \frac{1}{(a_i^n)^2} \left[\frac{p_{i+1}^n - p_{i-1}^n}{2\Delta x} \right] \right\} \quad (3.1.8)$$

As one can clearly see from the discrete Eqn. (3.1.8), both density and pressure are present as unknown quantities. To evade this problem we use the following

relationship derived from the ideal-gas equation of state and the expression for the speed of sound for such gases:

$$(a_i)^2 = \gamma R T_i = \frac{\gamma P_i}{\rho_i} \quad (3.1.9)$$

Subsequently, the discrete version of Eqn. (3.1.5) can be written as:

$$\begin{aligned} & (\rho_i^{n+1} - \rho_i^n) - \frac{1}{\gamma (a_i^n)^2} [(a_i^{n+1})^2 \rho_i^{n+1} - (a_i^n)^2 \rho_i^n] = \\ & - \Delta t (U' + \mu u_i^n) \left\{ \frac{\rho_{i+1}^n - \rho_{i-1}^n}{2\Delta x} - \frac{1}{\gamma (a_i^n)^2} \left[\frac{(a_{i+1}^n)^2 \rho_{i+1}^n - (a_{i-1}^n)^2 \rho_{i-1}^n}{2\Delta x} \right] \right\} \end{aligned} \quad (3.1.10)$$

In order to avoid carrying out an implicit calculation, as we have both ρ and a at time-step $(n + 1)$, the $(a_i^{n+1})^2$ term is “lagged” in time. In other words, we use the approximation $(a_i^{n+1})^2 = (a_i^n)^2$. This choice is in keeping with other similar methods (e.g. Hoffmann *et al.*, 2000). Once the density is obtained, the value of the static pressure follows easily from Eqn. (3.1.9). Having the discretized version of the governing equations allows us to carry out a suitable solution method, the subject of our next section.

3.2 THE RUNGE-KUTTA TIME-INTEGRATION SCHEME EMPLOYED IN *EE_SOLVER*

Both the *CharSolver* and *NonCharSolver* components of the *EE_Solver* algorithm implement the same second-order two-stage Runge-Kutta time-integration scheme. To outline the main differences between a classical two-stage Runge-Kutta scheme and the one used herein, consider once again Eqn. (3.1.3). If we leave the time-

derivative alone on the left-hand side of this equation and denote the remaining right-hand side terms by $-\mathbf{R}$, we have:

$$\frac{\partial \mathbf{Q}'}{\partial t} = -\mathbf{R} \quad (3.2.1)$$

A classic two-stage Runge-Kutta scheme for this set of equations is:

$$\begin{aligned} (\mathbf{Q}')_i^* &= (\mathbf{Q}')_i^n - \Delta t \mathbf{R}_i^n \\ (\mathbf{Q}')_i^{n+1} &= (\mathbf{Q}')_i^n - 0.5 \Delta t (\mathbf{R}_i^n + \mathbf{R}_i^*) \end{aligned} \quad (3.2.2)$$

The superscript * denotes an intermediate calculation, which is commonly referred to as a half time-step. In contrast, the current two-stage scheme for the same set of equations is:

$$\begin{aligned} (\mathbf{Q}')_i^* &= (\mathbf{Q}')_i^n \\ (\mathbf{Q}')_i^{**} &= (\mathbf{Q}')_i^* - \alpha_1 \Delta t \mathbf{R}_i^* \\ (\mathbf{Q}')_i^{***} &= (\mathbf{Q}')_i^* - \alpha_2 \Delta t \mathbf{R}_i^{**} \\ (\mathbf{Q}')_i^{n+1} &= (\mathbf{Q}')_i^{***} \end{aligned} \quad (3.2.3)$$

This deviates from “standard” Runge-Kutta schemes, since we have now included scaling parameters α_1 and α_2 . This variation is consistent with the approach of many well-known CFD algorithms, and is done to reduce computing memory requirements (e.g. Jameson *et al.*, 1981). In fact, the choice of $\alpha_1 = \frac{1}{2}$ and $\alpha_2 = 1$ results in the traditional scheme shown in Eqn. (3.2.2). Moreover, in order to maintain second-order temporal accuracy we require that $\alpha_1 = \frac{1}{2}$ and $\alpha_2 = 1$. In view of this fact, α_1 and α_2 were fixed at these values.

In principle, the discrete form of Eqn. (3.2.1) is sufficient to produce unique solutions given complete specification of initial and boundary conditions. However, since the continuum that exists in space is only represented in a discrete manner, the damping that exists in real life through the viscosity of fluids is insufficient to reduce the spurious oscillations that may develop in numerical solutions. Many existing algorithms rely on the artificial diffusion that is produced by virtue of the specific discretization used for each derivative in the governing equations, particularly the spatial convection term. This, however, does not allow for any control over the extent of artificial diffusion introduced into the solution. An alternative approach is to minimize the artificial diffusion inherent in the discretized derivatives, and then introduce the desired amount of artificial diffusion through terms explicitly added to the governing equations being solved. The extent of this diffusion would then be chosen to eliminate decoupling of the pressure field over odd- and even-numbered nodes on the spatial grid, and to damp oscillations in regions of high spatial gradients in flow variables, such as shock waves in high-speed flows.

Consequently, following the work of Jameson *et al.* (1981), an artificial diffusion term, D_a , consisting of second- and fourth-order differences of the solution vector in the spatial-grid direction, is added to the right hand side of the discretized form of Eqn. (3.2.1) within the *EE_Solver* algorithm:

$$D_a = \left(d_{i+\frac{1}{2}} - d_{i-\frac{1}{2}} \right) \quad (3.2.4)$$

The terms in each bracket on the right-hand side denote artificial diffusion along the x grid direction. For example:

$$d_{i+\frac{1}{2}} = \lambda_{i+\frac{1}{2}} \left\{ \varepsilon_{2, i+\frac{1}{2}} (Q_{i+1}'' - Q_i'') - \varepsilon_{4, i+\frac{1}{2}} (Q_{i+2}'' - 3Q_{i+1}'' + 3Q_i'' - Q_{i-1}'') \right\} \quad (3.2.5)$$

Here $\lambda_{i+\frac{1}{2}}$ is the spectral radius of the inviscid flux Jacobian matrix, $\mathbf{R}$, found in Eqn.

(1.4.12). The spectral radius is a common scaling parameter for the artificial dissipation term.

The second- and fourth-order nature of the finite differencing is clearly evident in Eqn. (3.2.5). The coefficients ε_2 and ε_4 are given by:

$$\begin{aligned} \varepsilon_{2, i+\frac{1}{2}} &= \kappa_2 \max(v_{i+1}, v_i) \\ \varepsilon_{4, i+\frac{1}{2}} &= \max\left(0; \left(\kappa_4 - \varepsilon_{2, i+\frac{1}{2}}\right)\right) \end{aligned} \quad (3.2.6)$$

where,

$$v_i = \frac{|p_{i+1} - 2p_i + p_{i-1}|}{p_{i+1} + 2p_i + p_{i-1}} \quad (3.2.7)$$

κ_2 and κ_4 are scaling coefficients chosen so that the solution is free of decoupling oscillations in space without having the artificial diffusion mask the physical process.

These two quantities appear to be slightly problem-dependent, but typically values

tend to be in the following ranges: $\kappa_2 = \frac{1}{4}$ and $\frac{1}{256} \leq \kappa_4 \leq \frac{1}{16}$ (Swanson *et al.*,

1991). The above values were adequate for converging all *EE_Solver* simulations.

The second- and fourth-order differences are first- and third-order accurate on a uniform mesh. As per the pressure switch given in Eqn. (3.2.7), the second-order difference term is activated only in regions of high-pressure gradients such as shocks in compressible flows. In these regions the solution accuracy therefore reduces to first-order in space. Fortunately, this usually accounts for a very small portion of the domain, hence minimally affecting solution accuracy. Elsewhere, only the fourth-order difference term remains active, in which case the formal spatial accuracy of the solution is driven by the order of accuracy of the convective and diffusive spatial derivatives, which are chosen to be formally second-order accurate. Finally, these operators require modification at computational domain boundaries, and the approach of Swanson and Turkel (1987) is adopted in the *EE_Solver* algorithm. Thus, the full equation set solved in *EE_Solver* is given by:

$$\frac{\partial \mathbf{Q}}{\partial t} = -\mathbf{R} + \mathbf{D}_a \quad (3.2.8)$$

where $\mathbf{Q}$ and $\mathbf{R}$ will be defined for either the characteristic or non-characteristic form, and $\mathbf{D}_a = [D_a \quad D_a \quad D_a]^T$.

As in Chapter 1, our next step here involves the implementation of a stability analysis to the *EE_Solver* discretization scheme; an undertaking that is now connected with the choice of the buffer zone parameters U' and μ . In Section 3.3 we carry out this numerical stability analysis, followed by a discussion regarding the choice of the parameters U' and μ in Section 3.4.

3.3 NUMERICAL STABILITY ANALYSIS FOR THE *EE_SOLVER* RUNGE-KUTTA SCHEME

A von Neumann stability analysis of the computational scheme (3.2.3), with $\alpha_1 = \frac{1}{2}$

and $\alpha_2 = 1$, applied to Eqn. (3.2.1) yields the following condition:

$$|\lambda_{\max}| \leq 1 \quad (3.3.1)$$

where, $\lambda_{\max}$ is the largest eigenvalue of:

$$\mathbf{I} + \tilde{\mathbf{D}} \frac{i \sin(k\Delta x)}{\Delta x} \Delta t + \left(\tilde{\mathbf{D}} \frac{i \sin(k\Delta x)}{\Delta x} \right)^2 \frac{(\Delta t)^2}{2} \quad (3.3.2)$$

The eigenvalues of this system are:

$$\begin{aligned} \lambda_1 &= 1 + [U' + \mu u] \frac{i \sin(k\Delta x)}{\Delta x} \Delta t + \left([U' + \mu u] \frac{i \sin(k\Delta x)}{\Delta x} \right)^2 \frac{(\Delta t)^2}{2} \\ \lambda_2 &= 1 + [U' + \mu(u+a)] \frac{i \sin(k\Delta x)}{\Delta x} \Delta t + \left([U' + \mu(u+a)] \frac{i \sin(k\Delta x)}{\Delta x} \right)^2 \frac{(\Delta t)^2}{2} \\ \lambda_3 &= 1 + [U' + \mu(u-a)] \frac{i \sin(k\Delta x)}{\Delta x} \Delta t + \left([U' + \mu(u-a)] \frac{i \sin(k\Delta x)}{\Delta x} \right)^2 \frac{(\Delta t)^2}{2} \end{aligned} \quad (3.3.3)$$

Thus, given the restrictions imposed on the various parameters, the stability

requirement outlined in Eqn. (3.3.1) amounts to $|\lambda_2| \leq 1$. This follows from the fact

that U' , μ , u , and a are each greater than zero. Written more formally, the stability

requirement for the current discretization scheme is:

$$\left| 1 + [U' + \mu(u + a)] \frac{i \sin(k \Delta x)}{\Delta x} \Delta t + \left([U' + \mu(u + a)] \frac{i \sin(k \Delta x)}{\Delta x} \right)^2 \frac{(\Delta t)^2}{2} \right| \leq 1 \quad (3.3.4)$$

It should be noted however that this analysis is not quite complete, since we have neglected to incorporate the artificial diffusion terms. The two-stage Runge-Kutta scheme with $\alpha_1 = \frac{1}{2}$ and $\alpha_2 = 1$ has its stability region depicted in Fig. 3.3.1. This stability region was shown pictorially in Chapter 1 (Fig. 1.5.1.2) along with stability regions for alternate stage Runge-Kutta schemes. The z term shown in Fig. 3.3.1 is produced from the application of von Neumann analysis. When the Runge-Kutta scheme given in Eqn. (3.2.3) is applied to Eqn. (3.2.1), we find that

$z = [U' + \mu(u + a)] \frac{i \sin(k \Delta x)}{\Delta x}$. Traditional Runge-Kutta stability regions show the real part of $z \Delta t$ plotted against its imaginary component, and for the two-stage case, the set of acceptable values produce the elliptical region shown in the Fig. 3.3.1.

Thus, when von Neumann analysis is applied to a spatially discretized set of equations, as above, the use of the two-stage Runge-Kutta scheme will be stable provided $z \Delta t$ falls within the region shown in Fig. 3.3.1, for all values of z .

Figure 3.3.1 – Stability region for an explicit two-stage Runge-Kutta scheme

With this mind, it then becomes clear that in the absence of dissipation terms, the Runge-Kutta scheme applied to Eqn. (3.2.1) will not be stable. This follows from the fact that $z\Delta t = [U' + \mu(u + a)] \frac{i \sin(k\Delta x)}{\Delta x} \Delta t$ is purely imaginary and thus lies outside the acceptable stability region.

Conversely, when we discretize the spatial derivatives of Eqn (3.1.3), this time including the dissipation terms, we obtain the following expression (dropping the " notation):

$$\begin{aligned} \frac{\partial Q}{\partial t} = & -\tilde{D}_i \left[\frac{Q_{i+1} - Q_{i-1}}{2\Delta x} \right] + 4(u+a)\varepsilon_2 \left[\frac{Q_{i+1} - 2Q_i + Q_{i-1}}{(\Delta x)^2} \right] \\ & - 4(u+a)\varepsilon_4 \left[\frac{Q_{i+2} - 4Q_{i+1} + 6Q_i - 4Q_{i-1} + Q_{i-2}}{(\Delta x)^4} \right] \end{aligned} \quad (3.3.5)$$

The question then becomes: how does the inclusion of these dissipation terms allow for a stable Runge-Kutta scheme? To answer this question, let us consider the following generic fourth-order “advection-diffusion”-type equation:

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = v \frac{\partial^2 u}{\partial x^2} - \mu \frac{\partial^4 u}{\partial x^4} \quad (3.3.6)$$

where c can be thought of as the wave speed and v and μ viscosity terms. The two terms on the right hand side of Eqn. (3.3.6) are analogous to our artificial dissipation terms introduced above. Although few, if any, physical processes fit into the framework of Eqn. (3.3.6), the equation is still useful in describing the effects of our second- and fourth-order artificial dissipation terms ($u_t + cu_x = vu_{xx}$ is the more familiar advection-diffusion equation). Following spatial discretization and the insertion of a normal mode solution $u = e^{ikx} \hat{u}$, we obtain the following amplification factor expression:

$$\hat{u}_{t,k} = \left\{ -c \frac{i \sin(k\Delta x)}{k\Delta x} + \frac{2v}{(\Delta x)^2} (\cos(k\Delta x) - 1) - \frac{2\mu}{(\Delta x)^4} (\cos(2k\Delta x) - 4\cos(k\Delta x) + 3) \right\} \hat{u}_k \quad (3.3.7)$$

If we denote the expression in the brackets by z , then after time discretization we have:

$$Re(z) = \frac{2v\Delta t}{(\Delta x)^2}(\cos(k\Delta x) - 1) - \frac{2\mu\Delta t}{(\Delta x)^4}(\cos(2k\Delta x) - 4\cos(k\Delta x) + 3) \quad (3.3.8)$$

$$Im(z) = -c \frac{\Delta t}{\Delta x} \sin(k\Delta x) \quad (3.3.9)$$

As k varies, the above two equations parametrically map out the region shown in Fig.

3.3.2:

Figure 3.3.2 – Stability region defined by Eqns. (3.3.8) and (3.3.9)

Consequently, Δx and Δt must be chosen so that, for all values k , the region shown in Fig. 3.3.2 is contained within that shown in Fig. 3.3.1. Mathematically, one choice that satisfies this condition is as follows:

$$\left| \frac{4v\Delta t}{(\Delta x)^2} + \frac{16\mu\Delta t}{(\Delta x)^4} \right| < 2 \quad (3.3.10)$$

$$\left| c \frac{\Delta t}{\Delta x} \right| < \left| \frac{4v\Delta t}{(\Delta x)^2} + \frac{16\mu\Delta t}{(\Delta x)^4} \right| \quad (3.3.11)$$

While the first inequality makes certain that the extreme point $\left(-\frac{4v\Delta t}{(\Delta x)^2} - \frac{16\mu\Delta t}{(\Delta x)^4} \right)$

lies within the ellipse shown in Fig. 3.3.1, Eqn. (3.3.11) “sandwiches” the curve of Fig. 3.3.2 so that it fits within the stability region of Fig. 3.3.1. Geometrically, the result might have the appearance of Fig. 3.3.3. The conditions given in Eqns. (3.3.10) and (3.3.11) in no way presuppose that optimal values of Δx and Δt may be obtained. On the contrary, this is in fact a rather conservative choice of constraints. Indeed, in the case of $\mu = 0$, this choice is in line with commonly used constraints; Eqn. (3.3.10) reduces to a “CFL-type” bound, while Eqn. (3.3.11) places a restriction on the cell Péclet number. Nonetheless, this example illustrates the role of artificial dissipation in the stability of numerical schemes. Moreover, in practice the optimal values of Δx and Δt are usually chosen through experimentation, and such conditions are meant to show how relative values of Δx and Δt behave as each is varied. Empirically we find a value of Δt for a chosen Δx , and then using the stability condition(s), we may find other acceptable values of Δt . Clearly the interplay between Δx and Δt is the underlying factor.

Figure 3.3.3 – A hypothetical stability region for the discretization of Eqn. (3.3.7) using a two-stage Runge-Kutta time integration scheme

As mentioned above, in addition to guiding the choice of appropriate grid spacing and time-step values, the numerical stability condition is also connected with the selection of U' and μ , as we shall see in the following section.

3.4 APPROPRIATE VALUES FOR THE BUFFER ZONE PARAMETERS U' AND μ

In Chapter 2 we presented Ta'asan and Nark's construction of the buffer zone parameters U' and μ , shown below:

$$U' = \beta(x - x_e)^m \text{ and } \mu = 1 - \eta(x - x_e)^m \quad (3.4.1)$$

In the study of Ta'asan *et al.*, U' is chosen to increase from 0 to β throughout the buffer zone, while μ is decreased from 1 to $(1-\eta)$. These definitions are appropriate when one keeps in mind that for their problem, the streamwise flow direction coincides with the x spatial coordinate direction. Furthermore, Ta'asan and Nark were aware of the final steady-state solution at the outset of their problem. As our test problem possesses these same features, the construction shown in Eqn. (3.4.1) is also carried out in *EE_Solver*. Chapter 5 explores the generalization of this definition.

Within *EE_Solver*, m is given a value of 8 in accordance with the choice made by Ta'asan and Nark (1995). The objective is to choose the value of m so that U' increases gradually throughout the buffer zone. Due to time constraints, no parametric optimization studies were conducted with regards to this parameter. In spite of this fact, future research examining the connection between this parameter and convergence rates might prove to be valuable.

Moreover, the choice of U' (or β) and μ (or η) are coupled according to stability condition (3.3.4) (actually according to the modified version of this stability condition, which follows from Eqn. (3.3.7) in the same manner). U' is first chosen to guarantee supersonic flow at the outer edge of buffer zone. According to the buffer zone theory, no modification should be made to the $(\Delta x, \Delta t)$ combination that was used on the same domain with no buffer zone. The inclusion of μ is aimed at counteracting increased speeds in the buffer zone which would otherwise require a decrease in Δt in order to maintain stability.

In this context, recall the stability condition derived in Chapter 2 for the set of equations considered in Ta'asan and Nark's (1995) work, namely:

$$\left| \cos(k\Delta x) - (U' + U) \frac{\Delta t}{\Delta x} (i \sin(k\Delta x)) - (V' + V) \frac{\Delta t}{\Delta y} (i \sin(l\Delta y)) + i\mu c\Delta t \left[\frac{\sin^2(k\Delta x) + \sin^2(l\Delta y)}{\Delta x \Delta y} \right]^{\frac{1}{2}} \right| \leq 1 \quad (3.4.2)$$

where, U' and V' are the x and y components of $\mathbf{U}'$, respectively, and similarly, U and V are the x and y components of $\mathbf{U}$, respectively. Eqn. (3.4.2) gives theoretical bounds on the choice of $\mathbf{U}'$, μ , and Δt . As described in Section 2.3.1, in the case of transient flow simulations where time-steps are traditionally small, one may achieve optimal solution convergence without the necessity for μ . Alternatively, for steady flow simulations, where often the largest possible time-step is sought, the increased flow velocity within the buffer zone causes stability problems. Thus, the introduction of μ is required to counteract this phenomenon.

Once U' is chosen to ensure only outgoing waves at the outflow boundary, a certain “freedom” exists in the choice of μ , as dictated by the numerical stability of the problem. In Chapter 4, specific values for these two parameters, U' and μ are presented in the context of our *EE_Solver* numerical simulations.

As a final note, we elaborate on one of the chief advantages of writing the Euler equations in characteristic form. Specifically, Eqns. (3.1.5) – (3.1.7) can now be manipulated with distinct U' and μ terms, as shown below:

$$\frac{\partial}{\partial t} \int \left(d\rho - \frac{dp}{a^2} \right) + [\mu_1 u + U'_1] \frac{\partial}{\partial x} \int \left(d\rho - \frac{dp}{a^2} \right) = 0 \quad (3.4.3)$$

$$\frac{\partial}{\partial t} \left(u + \frac{2a}{\gamma-1} \right) + [\mu_2 (u+a) + U'_2] \frac{\partial}{\partial x} \left(u + \frac{2a}{\gamma-1} \right) = 0 \quad (3.4.4)$$

$$\frac{\partial}{\partial t} \left(u - \frac{2a}{\gamma-1} \right) + [\mu_2 (u-a) + U'_2] \frac{\partial}{\partial x} \left(u - \frac{2a}{\gamma-1} \right) = 0 \quad (3.4.5)$$

where,

$$\begin{aligned} U'_1 &= \beta_1 (x - x_e)^m & \mu_1 &= 1 - \eta_1 (x - x_e)^m \\ U'_2 &= \beta_2 (x - x_e)^m & \mu_2 &= 1 - \eta_2 (x - x_e)^m \\ U'_3 &= \beta_3 (x - x_e)^m & \mu_3 &= 1 - \eta_3 (x - x_e)^m \end{aligned} \quad (3.4.6)$$

What is more, physical justification can be given for the overall treatment, as it is clear that any modification to an individual equation alters the corresponding wave speed within the computational domain. The introduction of individual μ and U' parameters for each equation (as opposed to the more restrictive method of using $U' = U'_1 = U'_2 = U'_3$ and $\mu = \mu_1 = \mu_2 = \mu_3$) shows the potential of the buffer zone technique for redirect incoming waves out, thus alleviating the necessity for an outflow boundary condition. Unfortunately, the characteristic system shown above is solved by *Char_Solver* in a non-coupled fashion, as explained in Section 3.1. As a result, we could not take full advantage of the characteristic form. Why is this so? The full characteristic form is composed of the three advection equations (3.1.5) – (3.1.7). At the outflow boundary of our domain, Eqn. (3.1.7) represents an incoming wave, while Eqns. (3.1.5) and (3.1.6) define outgoing waves. Thus, one boundary condition is required to account for the incoming Riemann invariant. As we have seen though, the specification of an outflow boundary condition causes reflection of the outgoing waves. Since Eqn. (3.1.5) is solved independently of Eqn. (3.1.6) and

(3.1.7), a boundary condition for pressure (or density) will reflect only the outgoing wave defined by Eqn. (3.1.5), and not that of Eqn. (3.1.6). We thus hypothesize that the application of the buffer zone modification to Eqn. (3.1.7) (i.e. incoming Riemann invariant) should produce the same results as an application to all the equations. In Chapter 4 we present the results of our *EE_Solver* simulations.

With a greater understanding of the equations being solved by *EE_Solver*, and the manner in which this is done, we are now in a position to analyse our simulation results, hence delving further into the fundamental aspects of buffer zone techniques.

4.0 SIMULATION RESULTS AND DISCUSSION

In order to obtain a preliminary insight into the difficulties associated with outflow boundary treatments – an insight complementing the theoretical findings presented thus far – fluid flow simulations were carried out with the full Navier-Stokes solver *FS*. Appendix D contains the computational domains that were considered in these simulations, along with other relevant information such as optimal algorithm-parameter settings and residual profiles. The pervasive nature of reflection demonstrated in these simulations was then studied more thoroughly in numerical experiments based on the inviscid flow solutions obtained with *EE_Solver*.

4.1 REFLECTIVE BOUNDARY DIFFICULTIES IN NAVIER-STOKES SIMULATIONS

In the *FS* algorithm, the residual component of each of the discretized governing equations is calculated and used to drive the solution to convergence (Appendix B). Reflection of outgoing characteristics from the outflow boundary can be viewed, in a practical way, by studying the spatial distribution and time rate of change of the residuals associated with the governing equations as the solution procedure is iterated in time. Encountering unusual “spikes” in the residual time profiles during simulations is usually an indication of significant reflection of outgoing characteristics at the boundaries. Of course, this is somewhat of an over-simplification, since the convergence rate of the solution also depends on the appropriate specification of several iteration parameters. It will be shown in the next few sections that the lack of efficient convergence demonstrated in this chapter is indeed primarily due to reflection from the outflow boundary, and thus, not an artifact of incorrect parameter specification.

The *FS* algorithm solves steady flows by marching in artificial time to a steady-state. In conjunction with this, the L_2 – norm of the residual for each of the governing equations is computed at every time step (Appendix D). Results presented in this section are based on the L_2 – norm of the residual.

As a first example, Fig. 4.1.1 shows the x -momentum equation residual profile for the first 500 iterations of a two-dimensional laminar flow simulation. The computational domain is enclosed by one inflow boundary at $x = 0\text{ m}$, one outflow boundary at $x = 0.60\text{ m}$, and lower and upper wall boundaries at $y = 0\text{ m}$ and $y = 0.12\text{ m}$, respectively. A velocity of 10 m/s in the x -direction is specified at the inflow boundary, a static pressure of 1 Pa is fixed at one node of the outflow boundary, and all remaining nodes in the domain are initialized with zero velocity and 1 Pa static pressure. To ensure an accurate spatial representation of the flow, 101 uniformly distributed spatial grid nodes are used in the streamwise (x) direction, and 51 uniformly distributed nodes in the cross-streamwise (y) direction.

Static pressure was fixed at one node of the outflow boundary according to Rudy *et al.*'s Eqn. (2.2.3.7) with $\alpha = 0.1$. As we remarked in Chapter 2, specification of static pressure can have severe ramifications on the solution convergence. This becomes evident when we compare the x -momentum residual profile shown in Fig. 4.1.1 with the absolute change in pressure occurring during this laminar flow simulation, depicted in Fig. 4.1.2. From Fig. 4.1.2, one can clearly correlate the residual spikes to the pressure fluctuations as a result of the fixed static pressure boundary condition.

Figure 4.1.1 – X-momentum equation residual profile for a two-dimensional laminar flow simulation, $Re_L = 40000$, $M_\infty = 0.03$

Figure 4.1.2 – The maximum absolute change in pressure during the above laminar flow simulation

Generally, residual profiles vary from one simulation to the next depending on the boundary and initial conditions, as well as other factors such as domain shape.

Similar trends were noted for both the y-momentum and continuity equation residual profiles. In these simulations, no effort was made to correlate residual trends with the detailed dynamics of waves in the vicinity of the outflow boundary. However, we shall revisit this idea later in the chapter in the context of our *EE_Solver* simulations.

As another example, the y-momentum equation residual profile shown in Fig. 4.1.3 is produced from a two-dimensional steady pipe flow simulation. The initial and boundary conditions are chosen as in the preceding example. Spikes created at the instant the outgoing characteristics reach the outflow boundary for the first time, at approximately the 120 iteration mark, are apparent.

Figure 4.1.3 – Y-momentum equation residual profile for a two-dimensional steady pipe flow simulation, $Re_L = 40000$, $M_\infty = 0.03$

Figs. 4.1.4 and 4.1.6 are based on three-dimensional (in space) simulations involving flow through a rectangular duct and flow through a pipe, respectively. The overall residual profiles are noted to be somewhat different from those given for the two-dimensional flows. Nevertheless, undesirable “stalling” or spikes in residual profiles that adversely affect the convergence rate of the solution, show themselves to be present in each case. Fig. 4.1.5 shows the absolute change in pressure for this three-dimensional steady rectangular duct flow simulation. This is the analogous figure to 4.1.2. In addition, Fig. 4.1.7 shows the converged artificial diffusion distribution for the x -momentum equation along the center streamline of our three-dimensional pipe computational domain. Artificial diffusion, which was described in Chapter 3, is created by the pressure difference between the fixed pressure boundary node and the adjacent interior nodes. Consequently, this value remains comparably higher at the last three nodes of the computational domain, even after a steady state has been reached.

Figure 4.1.4 – X-momentum equation residual profile for a three-dimensional steady rectangular duct flow simulation, $Re_L = 40000$, $M_\infty = 0.03$

Figure 4.1.5 – The maximum absolute change in pressure during the above rectangular duct flow simulation

Figure 4.1.6 – X-momentum equation residual profile for a three-dimensional steady pipe flow simulation, $Re_L = 40000$, $M_\infty = 0.03$

Figure 4.1.7 – X-momentum equation artificial diffusion profile (at steady state) for a three-dimensional steady pipe flow simulation

The few results presented thus far do not give a full indication of the problems incurred throughout the simulations using *FS*, nor are they sufficiently indicative of the effort spent converging each solution. However, in the intention of presenting outflow boundary reflection as a numerical roadblock to convergence, they serve the purpose adequately.

In the next few sections, the outflow boundary reflection problem is examined more quantitatively based on the solution of very simple flows for which the Euler equations in one-dimensional space are applicable.

4.2 BOUNDARY REFLECTION STUDIES WITH *EE_SOLVER*

The test case being considered in this section is based on the inviscid, one-dimensional flow in a straight duct that is marched in time from stationary initial conditions to a steady state. The reflection problem at the outflow boundary is studied as the solution is marched in time to the final state of a uniform flow field.

The computational domain to be used for the present test case is shown in Fig. 4.2.1. For aesthetic purposes, it is depicted here with an arbitrary height in the y -direction. However, given the one-dimensional nature of the problem, only the x -direction is considered in the numerical implementation. 100 spatial nodes are uniformly situated along the streamwise direction with $\Delta x = 0.01$. This conservative choice of node count is made in order to accurately capture the transient development of the flow in the streamwise direction.

The inlet Mach number is chosen to be subsonic ($M = 0.7$) in order to ensure that the flow is subsonic at the outflow boundary. This condition is preferred at the outlet since under supersonic conditions no incoming characteristics exist at this boundary. The underlying assumption throughout has been that the presence of an incoming characteristic is a prerequisite to the reflection problem for the outgoing characteristics. More specifically, it is the manner in which flow information is prescribed at an outflow boundary to account for the incoming characteristics that can potentially yield reflection of the outgoing characteristics.

An inlet Mach number of 0.7 is achieved by specifying a velocity of 241 m/s and a static temperature of 683°C at the inflow boundary node. All other nodes (i.e. interior nodes and outflow boundary node) are prescribed an initial velocity of zero. A summary of the initial and boundary conditions for Test Case 1 (described below) is presented in Fig. 4.2.1. Although variable quantities are presented here in their dimensional form, the non-dimensional versions of the equations are solved in *EE_Solver*.

Figure 4.2.1 – Summary of initial and boundary conditions for Test Case 1

Fixing of the static density fulfils the required outflow boundary condition, a technique previously discussed and known for its consequences. It should be noted that traditionally among CFD practitioners, the predominant choice for the variable to be fixed at subsonic outflow boundaries is the static pressure. The selection of pressure as opposed to another thermodynamic variable is due to its availability in most fluid flow problems of interest.

As described earlier, an advantage of the research presented herein is that the governing equations are examined in both their characteristic and non-characteristic form. Yet, as explained in Chapter 3, when the characteristic form of the equations is used, Eqns. (3.1.6) and (3.1.7) are solved initially for u and a , followed by the solution of density from Eqn. (3.1.10). One may have alternatively formulated Eqn. (3.1.10) in order to solve for pressure, with the overall results being similar. The selection of density is made with the purpose of avoiding additional (and unnecessary) computations. The use of the characteristic form will be considered in Section 4.5. Nonetheless, fixing of the static density adversely affects the static pressure calculation, both in non-characteristic and characteristic form, as the two variables are related according to Eqn. (3.1.9). For the present test case the static density is set to 1.1575 kg/m^3 . This value is intentionally chosen to be different from that used to initialize ρ in the interior of the domain. The motivation for this decision stems from the fact that in practice, poor initial guesses for fixed outflow variables, due to a lack of a priori knowledge, often cause severe reflection problems. Of course, treating the boundary in this way introduces a small discontinuity between the value specified at the outflow node and the calculated value at the adjacent node. This in turn affects the convergence rate of the solution, since the finite-differencing operation carried out

at the last three nodes of the domain will yield non-zero values for p . Nonetheless, the fixed boundary condition still induces a great amount of reflection, as we shall see shortly. For convenience, from this point on the abovementioned problem, with computational domain depicted in Fig. 4.2.2, shall be referred to as Test Case 1.

Figure 4.2.2 – Test Case 1 computational domain. A finite thickness in the y direction is shown for aesthetic purposes

Unlike our *FS* simulations presented in Section 4.1, trials conducted with Test Case 1 quantify reflection so as to obtain some insight into the effectiveness of the buffer zone technique. Several factors lead to our present choice of quantification method, the first among them being the desire to isolate the phenomenon of reflection in our simulations. Consequently, quantification and comparison of reflection was facilitated by matching numerical solutions against results obtained from a concurrent simulation carried out on an “infinite domain”. In this way, the presence of other factors that could potentially hinder solution convergence is accounted for in the simultaneous (infinite domain) simulation, and thus, outflow boundary reflection emerges as the focal point. Reflection errors are given a specific value according to

Eqn. (4.2.1), which can be thought of as the L_2 – norm of the pressure residual (denoted by p_{error}). This calculation is carried out at each node excluding those within the buffer zone. Furthermore, a new p_{error} value is calculated at each iteration. Lastly, the fact that other valuable studies in this field of research, including those of Ta'asan *et al.* and Hayder *et al.*, quantify reflection in a similar manner (Chapter 2), our choice here allows for a comparison, at least qualitatively, with these contributions.

$$p_{error} = \left\| \frac{p_{numeric} - p_{\infty}}{p_{\infty}} \right\|_2 \quad (4.2.1)$$

It might appear that Test Case 1 has been concocted for the purpose of this research. However, physically realistic situations can be described by this process, such as the purging of a pipe or duct. It is for this reason that Test Case 1 can be simulated with the intention of studying the transient development of the flow, or with the interest of efficiently converging to a steady-state. We begin here with the former assumption and revisit the steady-state situation in Section 4.4. Studying the transient development of this flow is beneficial, as our aim is to reduce reflection from the outflow boundary; an observable fact that takes place prior to the reaching of a steady state. For this reason, we arbitrarily choose the value of 5% or 0.05 for p_{error} as the point of “convergence”. Although p_{error} values are reduced to orders of magnitude lower than 0.05, this choice provides sufficient time for reflection to transpire. Unless otherwise stated, all results are presented for the equations in their non-characteristic form. Furthermore, as the *EE_Solver* calculations are carried out explicitly, and p_{error}

is computed in a transient fashion (that is, at every time step of the process), each iteration of the Runge-Kutta time-integration used in the *EE_Solver* algorithm is equivalent to one time step. Thus, all figures resulting from *EE_Solver* simulations are presented with the x -axis labeled “iterations.”

Fig. 4.2.3 shows the result of a simulation where the non-characteristic form of the equations is used. Twelve pressure probes are taken at various iteration counts, and the corresponding pressure distributions throughout the computational domain are plotted. As the simulation progresses, one can observe the pressure wave moving from inflow to outflow boundary, and the subsequent reflection caused by our rigid boundary condition (fixed density at the outflow boundary). Also evident in these probes, namely ① – ③, is the non-zero finite-differencing at the outflow boundary resulting from a mismatch between the density value at the outflow node and its adjacent neighbour (shown in the first two probes of Fig. 4.2.4). As the pressure wave proceeds from left to right, one would expect complications to occur when it reaches the outflow boundary. While this is true, the finite difference calculation detects this disparity in density between the two nodes mentioned above, and subsequently causes a change in pressure to propagate inward from the outlet. In this way, the p_{error} curve shown in Fig. 4.2.3 includes both reflection and this pressure calculation propagating inward from the outflow boundary. The first eight pressure probes are depicted in detail in Fig. 4.2.4.

Figure 4.2.3 – p_{error} profile for a Test Case 1 *NonCharSolver* simulation showing pressure probes at various iteration counts – fixed density outflow boundary condition

Figure 4.2.4 – The first eight pressure probes from a *NonCharSolver* Test Case 1 simulation - fixed density outflow boundary condition

Upstream of the wave-front, the pressure continues to be modified as there is coupling between pressure and velocity through the x -momentum equation. Moreover, the disparity between the newly modified velocity field and the Dirichlet boundary conditions for u and a at the inlet cause the pressure spikes commencing in Figure ④.

As the simulation proceeds in time, the pressure stabilizes at the appropriate level (Fig. 4.2.5).

Using the speed of sound, a , and grid spacing for the current problem, a quick calculation shows that the outgoing characteristics reach the outflow boundary in approximately 5 iterations. Thus, reflection from the outflow boundary takes place between snapshots ④ and ⑤. What is more, Nitta (1962) and Matsuno (1966) demonstrate that the magnitude of the reflection coefficient (Eqn. (1.7.1.13)) generated by a fixed boundary condition, such as the one employed here, is unity.

Fig. 4.2.5 shows the remaining probes ⑨ – ⑫ as the solution converges to its uniform steady-state value.

Along with reflection from the outflow boundary, a standing wave is also produced due to the fixed boundary condition. The existence of a standing wave at the outlet is verified by plotting the pressure and artificial dissipation profiles for a converged *NonCharSolver* simulation of Test Case 1 with a fixed density outflow boundary condition (Figs. 4.2.6 and 4.2.7).

Figure 4.2.5 – The four pressure probes ⑨ – ⑫ taken from Fig. 4.2.3 for a *NonCharSolver* Test Case 1 simulation - fixed density boundary condition

Figure 4.2.6 – Normalized pressure distribution at steady state for a *NonCharSolver* simulation with fixed density outflow boundary condition

Figure 4.2.7 – X-momentum artificial diffusion distribution at steady state for a *NonCharSolver* simulation with fixed density outflow boundary condition

Before implementing the buffer zone technique as a possible remedy for the reflection observed above, let us take a quick look at the outcome involved in fixing outflow density equal to that of the initialized computational domain value (i.e. $\rho_{Npts} = 0.5$ for the present case). Fig. 4.2.8 shows 8 pressure probes from the first eight iterations of a *NonCharSolver* simulation conducted with this new outflow boundary condition. Clearly, the finite-differencing calculation near the outlet does not have the same effects as before. Moreover, reflection (according to Eqn. (4.2.1)) occurs mainly in the form of a standing wave, which in this case is evident as early as iteration ⑧.

Given the complexity of many fluid flow problems, it is highly unlikely that the specification of an outflow boundary condition will be as accurate as this latter case. In this way, we have actually demonstrated that reflection problems can definitely be sensitive to initial and boundary conditions.

Figure 4.2.8 – The first eight pressure probes from a *NonCharSolver* Test Case 1 simulation - fixed density boundary condition

The small disturbances encountered in the first few iterations, as shown in Fig. 4.2.8, are not indicative of the same difficulty encountered above when we set $\rho_{Npts} = 1.1575$. It is well known that centered differences lead to spatial oscillations (“wiggles”) due to numerical dispersion. These small wiggles can be seen in the second and third pressure probe diagrams of Fig. 4.2.8.

We now proceed to examine the prospect of Ta’asan and Nark’s buffer zone technique, introduced by Streett (1986) for practical fluid flow simulations, as a feasible remedy for the difficulties encountered thus far. In executing the buffer zone modification, we progress in the same manner as Ta’asan and Nark (1995), namely by extending the outflow boundary in the positive x -direction, as shown in Fig. 4.2.9.

Figure 4.2.9 – Graphical representation of the buffer zone modification used in Test Case 1 simulations

It should be noted that reflection of outgoing characteristics at the inflow boundary exerts little influence on the overall convergence of the solution. Consequently, no buffer zone treatment is implemented at this boundary. Details pertaining to how the

buffer zone modification could be implemented in the context of more elaborate computational domains shall be discussed in Chapter 5.

4.3 TEST CASE 1 SIMULATION RESULTS WITH AN OUTFLOW BUFFER ZONE

Section 3.4 discusses the underlying principles involved in choosing appropriate values of U' and μ in regards to Ta'asan *et al.*'s buffer zone technique. Systematic studies were carried out to determine the effects of these parameters on simulation convergence in relationship with varying buffer zone sizes. The abbreviation BW denotes the buffer zone width (or length) used, with the value indicating the length of the buffer zone as a fraction of the computational domain length. The use of a zero-streamwise gradient for all flow variables (velocity, temperature, and pressure) will suffice at the outer edge of the buffer zone, since by virtue of the buffer zone technique, the flow is artificially supersonic at this boundary.

Any conclusions regarding the effectiveness, or lack thereof, of the buffer zone technique, presupposes that the buffer zone parameters have been optimized for Test Case 1. Thus, we pause briefly in the next section to indicate how these optimized values were obtained.

4.3.1 DEFINITION OF U' AND ITS CONNECTION WITH β AND μ

The a priori knowledge of the steady-state solution for Test Case 1 is known, facilitates the selection of a minimum value of U' . In particular, when steady-state for Test Case 1 is reached, the speed of sound, a , obtains the value of 344 m/s throughout the computational domain. Therefore, we require that the artificially

introduced speed, U' , have as a minimum, the value of 1 (non-dimensional) at the outer edge of the buffer zone (i.e. one times the speed of sound). Moreover, since $U' = \beta(x - x_e)^m$ ($m = 8$), the selection of β must be correspondingly chosen. If x_{Npts} refers to the x -coordinate at the outer edge of the buffer zone and x_e refers to the point at the interface between the original outflow boundary and the buffer zone (see Fig. 2.3.1.1), then an obvious formula for β is as follows:

$$\beta = \frac{\text{desired value of } U' \text{ at the outer edge of the buffer zone}}{(x_{Npts} - x_e)^m} \quad (4.3.1.1)$$

For matters of brevity and ease of reading, further references, either in text or in figures, shall present the value of U' only at the outer edge of the buffer zone, denoted by U'_{Npts} . This makes easy the comparison of simulations with different buffer widths. A similar treatment holds for the parameter μ , where the outer edge value shall be given by μ_{Npts} .

Let us recall that our interest for the time being is to study the transient flow paths. Thus, we are more concerned with the optimization of U'_{Npts} than with maximizing our simulation time-step, as would be the case if the transient solutions were of no interest to us. For this reason, the selection of μ_{Npts} (or η) need not be considered for the moment, provided its value is adequate for stability. In this regard, a conservative value of 0.1 is chosen for μ_{Npts} . Having made this choice of μ_{Npts} , we may now turn to the issue of optimizing U'_{Npts} .

4.3.2 OPTIMIZING U'_{Npts} , BW, AND BUFFER ZONE NODE COUNT

Once the selection of μ_{Npts} has been made, there remain three competing factors: the choice of U'_{Npts} , the selection of a maximum buffer zone width, and the optimal node count within the buffer zone. Clearly, the three parameters are interrelated, and hence, a logical method is indispensable for arriving at appropriate values for each. A small example will help to illustrate the interconnectivity of these three terms.

Suppose a particular choice of U'_{Npts} is made, and with this value numerous simulations are conducted with varying buffer zone widths and node counts. Subsequently, assume that from these simulations an optimal buffer zone width and node count is found for our choice of U'_{Npts} . Despite having obtained these latter two values, there is no guarantee at this point that the value of U'_{Npts} is optimal. We may only conclude that for this particular choice of U'_{Npts} , we have found an optimal buffer zone width and node count. One would then be required to parametrically iterate on U'_{Npts} in order to conclude confidently that no other value gives way to improvements. If on the other hand, a different value of U'_{Npts} produces superior results, the process must repeat itself for this new value of U'_{Npts} . Buffer zone widths less than 10% of the computational domain width (i.e. $BW < 0.1$) were not considered.

Although trivial, the above example identifies precisely the manner in which we determined our optimal values for U'_{Npts} , buffer zone width, and node count. The appropriate selection of these three parameters indeed has important ramifications. Specifically, for more complex geometries, the addition of large buffer zones leads to

increased simulation times, particularly if the node count within the buffer zone is not chosen appropriately.

One cycle of this optimization procedure outlined in the previous paragraph, with $U'_{Npts} = 1$, yielded an optimal buffer width value of $BW = 0.3$ and 5 nodes within this zone. The next step of the iteration process is summarized in Fig. 4.3.2.1. Namely, for $BW = 0.3$ and 5 nodes within this buffer zone, simulations were conducted with gradually increasing values of U'_{Npts} to elucidate whether or not a more preferable value could be used. As is clear from Fig. 4.3.2.1, the value of $U'_{Npts} = 1$ is certainly not the optimal choice. In contrast, the value of $U'_{Npts} = 10$ was found to be a fairly strict upper bound. Divergence occurred at roughly $U'_{Npts} = 11$, and no benefit was derived from intermediate values (i.e. $10 < U'_{Npts} < 11$). Consequently, as a first guess, the above result suggests that the largest allowable value of U' , within stability requirements, is also the optimal value. As we shall see, Section 4.4 will give further support to this assertion.

Figure 4.3.2.1 – *NonCharSolver* Test Case 1 simulations for various U'_{Npts} values, with BW = 0.3 and 5 nodes in the buffer zone

Replicating the studies from above with $U'_{Npts} = 10$ gave way to an optimal trio of parameters. Fig. 4.3.2.2 summarizes the results of these studies for BW = 0.1 and 0.2, and buffer zone node counts ranging from 2 to 5. Alternative values for these two parameters, specifically larger buffer zone widths and increased node counts, were omitted from this figure as they provided no additional benefit.

Figure 4.3.2.2 – *NonCharSolver* Test Case 1 simulations for BW values of 0.1 and 0.2 and buffer zone node counts ranging from 2 to 5, with $U'_{Npts} = 10$

The two lower overlapping lines of Fig. 4.3.2.2 represent p_{error} curves for simulations having $BW = 0.1$ and 0.2 with 5 nodes within the buffer zone, respectively. One curve with $BW = 0.7$ and 10 nodes in the buffer zone is inserted merely to show that no computational advantage was gained with buffer zone widths greater than $BW = 0.1$, even when larger node counts were used within these buffer zones. In particular, simulations with $BW = 0.3 - 0.7$, $U'_{Npts} = 10$, and varying node counts produced nearly identical results. Thus, we have the following optimal buffer zone parameters:

Table 4.3.2.1 – Optimized Test Case 1 buffer zone parameters

U'_{Npts}	BW	Number of Nodes in the Buffer Zone
10	0.1	5

As mentioned in Chapter 2, the values given in Table 4.3.2.1 stand, to the best of our knowledge, as the first published account of optimal values for these three buffer zone parameters. The results obtained also reinforce our earlier comments of Chapter 2, pointing to the advantage that the buffer zone technique offers over absorbing techniques, particularly in terms of node counts within the buffer layer. It is important to note that, in presenting optimal node counts for buffer zones, the given values should not be considered as absolutes, which may be applied arbitrarily to any fluid flow problem. Rather, one must consider the choice in the light of the spatial discretization scheme. For the current work, the fourth-order dissipation term requires five nodes for its calculation; precisely the number of nodes used in our buffer zone. If on the other hand, a higher-order scheme were used requiring more nodes for its calculation, it is expected the efficiency of the buffer zone technique will diminish. We actually get a hint of this when we decrease the number of nodes to less than 5 in the buffer zone. No studies were carried out with alternative computational stencils. In this new light then, we examine a particular result of Hayder *et al.* (1997) more closely. In particular for their simplest two-dimensional shock-vorticity wave interaction problem (2-D Euler equations), despite having used a compact sixth-order scheme to discretize the spatial derivatives, 25 nodes were used within their absorbing layer. Keeping in mind that such compact schemes are designed to reduce the number of nodes involved in spatial discretization, we again conclude that absorbing techniques are less sensitive to computational stencil and more affected by the nature of the problem. The above results would suggest that the buffer width is of secondary concern, provided there are a minimum number of nodes within the buffer zone. Clearly the present research can benefit from higher dimensional trials in order to ascertain whether the same trends hold there as well.

Table 4.3.2.2 – CPU comparisons for Test Case 1 optimization studies

U'_{Npts}	BW	Buffer Zone Nodes	Total CPU Time	Iterations	CPU time per Iteration
10	0.1	2	36.5 s	20	1.82 s
10	0.1	3	23.0 s	17	1.35 s
10	0.1	4	5.8 s	5	1.16 s
10	0.1	5	5.8 s	5	1.16 s
10	0.2	2	36.5 s	20	1.82 s
10	0.2	3	31.8 s	17	1.86 s
10	0.2	4	7.4 s	5	1.47 s
10	0.2	5	7.5 s	5	1.49 s

With the selection of the optimal parameters in place, we may now proceed to study the potential benefits derived from the buffer zone implementation, with a focus on reduction in reflection and hence CPU time.

4.3.3 TEST CASE 1 SIMULATIONS USING OPTIMAL BUFFER ZONE PARAMETERS

Intuitively, as a starting point we ask ourselves if any significant improvement is obtained by the use of the buffer zone technique with our optimal parameters over the traditional boundary condition outlined in Section 4.2. Fig. 4.3.3.1 is an extension of Fig. 4.2.3 which includes a *NonCharSolver* Test Case 1 simulation with optimal buffer zone parameters. A summary of this experiment is given in Table 4.3.3.1 and highlights the major gains afforded by this modification. The CPU time per iteration does increase with buffer zone implementation for the *NonCharSolver* simulations as expected, however is more than offset by the reduced iteration count to convergence.

Figure 4.3.3.1 – Test Case 1 p_{error} trends for two *NonCharSolver* simulations conducted with no buffer zone and optimal buffer zone, respectively

Table 4.3.3.1 – Summary of the gains provided by the optimal implementation of the buffer zone technique

TEST CASE 1 SIMULATION	Max p_{error}	Reduction in p_{error}	Total CPU Time	Reduction in Total CPU Time	CPU Time per Iteration
<i>Non-characteristic Form</i>					
No buffer zone	4.43		210 s		1.47
$BW = 0.1, U'_{Npts} = 10$	0.45	90 %	7.5 s	96.4%	1.49

Lastly, we can plot the “ $\frac{p}{p_{ref}}$ vs. streamwise node position” curves for the first eight

iterations of this buffer zone simulation, and subsequently compare them to those found in Fig. 4.2.4. As one would expect, we no longer experience reflection at the outflow boundary, nor produce a standing wave, as no boundary condition is needed at the outlet.

Figure 4.3.3.2 – The first eight pressure probes from a *NonCharSolver* Test Case 1 simulation – optimal buffer zone parameters

For completeness, we now introduce a short section outlining how similar benefits to those observed above may also be obtained within simulations whose aim is the final steady-state rather than the transient flow development.

4.4 STEADY-STATE SIMULATIONS

Until now we have been considering transient flow simulations. For the researcher carrying out such experiments, particularly on very small time-scales such as those encountered in the study of turbulence, the results thus far are beneficial. However, in steady-state simulations where the transient path is not of interest, one wishes to maximize the time-step in order to reach a converged state as quickly as possible. Unfortunately, as we have observed on a number of occasions, not only is the numerical stability of a given algorithm related to the choice of time-step, but also to the various “wave speeds” within the domain. As a result, we expect that the implementation of the buffer zone technique will encounter problems when carried out at the maximum Δt and in the absence of the parameter μ . In fact, the current demonstrates the necessity for μ in steady-state simulations using maximum time-steps. When *NonCharSolver* simulations were conducted with the maximum allowable time-step and $\mu_{Npts} = 0.1$, a new optimal of $U'_{Npts} = 7$ was required for stability, as depicted in Fig. 4.4.1. With the maximum time-step in place, values of U'_{Npts} greater than 7 resulted in divergence. These results actually mimic those of Section 4.3.2. In particular, as we mentioned there, we are again led to conclude that the optimal value of U' is at the same time, the maximum value. Although small differences in convergence patterns were observed between our *NonCharSolver* simulation carried out at the maximum time-step and that of the previous section

(i.e. $U'_{Npts} = 10$, $\mu_{Npts} = 0.1$, and $\Delta t < \Delta t_{max}$), as shown in Fig. 4.4.1, two more pressing issues must be pointed out here. In particular, of central importance here is that the simulation using Δt_{max} cannot be converged without the inclusion of μ , as otherwise instabilities cause solution divergence. Also of significant consequence, the second of these issues points to the fact that once the minimum value of μ needed for stability is identified ($\mu_{Npts} = 0.4$ in this case), larger values have no effect on the convergence rates. Thus, the manner in which one goes about finding the maximum value of U'_{Npts} as outlined in Section 4.3.2, is identical when Δt_{max} is used. μ is then selected merely for stability purposes, but as is now clear, its inclusion is a necessity.

Figure 4.4.1 – Four *NonCharSolver* simulations indicating the necessity for a new optimal value of U'_{Npts} when Δt_{max} is used with various value of μ_{Npts}

As a final investigation, we have recourse to the characteristic form of the Euler equations in order to explore potential improvements to the current buffer zone technique.

4.5 THE BUFFER ZONE MODIFICATION WITH THE EULER EQUATIONS IN CHARACTERISTIC FORM

The majority of CFD algorithms implement their particular equations of interest in a non-characteristic form, and hence the application of a buffer zone technique results in a modification similar to that shown in Eqn. (3.1.1). The use of distinct parameters U' and μ for each equation (continuity, momentum, and energy) when the non-characteristic form is employed is of course possible, however, the justification for this methodology is unclear. Consequently, researchers are forced to use a uniform set of parameters, or otherwise, carry out a tedious parametric study. Until now, no research has addressed whether differences exist between the two set-ups.

It is important to note that our main objective is not to conduct numerous parametric studies in the hopes of obtaining trends whose a priori knowledge was never hypothesized. Rather, by carrying out specific simulations, the focus is to convincingly demonstrate that past applications of the buffer zone technique may have been done more efficiently, while providing future researchers with this valuable knowledge.

The fundamental aim of the buffer zone technique is to counteract numerical errors introduced through the reflection of outgoing Riemann invariants. If reflection of the outgoing Riemann invariants is caused by the specification of one boundary condition

at the outflow boundary, which accounts for the incoming Riemann invariant, then one would speculate that the buffer zone treatment of the incoming wave alone is sufficient for counteracting reflection. From the examples of Chapter 1 and the conclusions of this chapter, the above hypothesis seems quite intuitive. More specifically, we began by considering the one-dimensional advection equation, (1.6.1.a). From there, we saw an example of the interaction of two advection-type equations, namely Eqns. (1.7.1) and (1.7.2). In the first case, an outflow boundary condition was not needed, provided $c > 0$. In fact, in Section 1.7.1 we gave some idea of how detrimental such a boundary condition can be if employed. In the second case, the boundary condition required for the incoming Riemann invariant at the outflow boundary caused reflection of the outgoing Riemann invariant.

Theoretically, for the full characteristic form given by Eqns. (3.1.5) – (3.1.7) we expect that the boundary condition required for Eqn. (3.1.7) should cause reflection of the waves described by Eqns. (3.1.5) and (3.1.6) at the outflow boundary.

Unfortunately, the method by which this system is solved in *CharSolver* decouples Eqn. (3.1.5) from the other two (see Section 3.4). Regardless of this fact, application of the buffer zone treatment to Eqn. (3.1.7) gives the wave which it describes a supersonic speed at the outer edge of the buffer zone. Thus, no outflow boundary condition is required, and as Fig. 4.5.1 confirms, this eliminates the reflection problem. No attempt was made to quantify this reduction in reflection, as the solution method for this set of equations is somewhat contrived, and the results of Fig. 4.5.1 seem intuitive.

Figure 4.5.1 – Three *CharSolver* simulations modifying different combinations of incoming and outgoing Riemann invariants

Further studies will likely show whether or not these trends are consistent for a more coupled system. From our theoretical investigations and the experimental results presented earlier in the chapter, it appears that the uniform application of one set of parameters (U', μ) to all the equations would reduce reflection to the same level as a simulation that modifies only the incoming waves. If this statement turns out to be true, the application of the buffer zone to equations in non-characteristic form is further justified.

With these results in place, we establish that the buffer zone technique may indeed be a useful tool in the field of computational fluid dynamics, particularly when the

suggested modifications to the governing equations are well understood, both mathematically and physically.

In order to complete this analysis of the buffer zone method as a viable alternative to traditional outflow boundary conditions, we make the buffer zone technique completely accessible by describing its implementation into full Navier-Stokes solvers. This description, given in Chapter 5, uses *FS* as the prototypical full Navier-Stokes algorithm.

5.0 IMPLEMENTATION OF THE BUFFER ZONE TECHNIQUE INTO FULL NAVIER-STOKES ALGORITHMS

The results presented in Chapter 4 were obtained using a simplified set of equations derived from more general Navier-Stokes relationships. Although these experiments provide greater insight into the nature of the buffer zone technique, we endeavor to complete this analysis by describing how one would implement this method in the setting of a full Navier-Stokes algorithm. With this mind, two issues immediately emerge as potential roadblocks to this implementation. In the first place, we must explain why the use of the buffer zone technique is valid in conjunction with the procedure known as artificial compressibility, an issue that has remained unexplored until this point. This technique of artificial compressibility is presented below, as well as in Appendix B. Secondly, we would like to generalize the definitions of U' and μ , thus expanding on Eqn. (3.4.6), which was implemented for a simple rectangular two-dimensional computational domain. Unfortunately, for many complex three-dimensional flows, a simple, orthogonal set-up such as the one employed in this earlier example is rare.

The discussion of these two important issues shall be considered in the context of the full Navier-Stokes algorithm, *FS*, regarding which, a thorough description may be found in Appendix B. To facilitate a detailed examination of the abovementioned problems it will be the author's assumption that the reader has some background knowledge in fundamental CFD issues. However, prior to doing so, we shall briefly describe the particular components of the *FS* algorithm that will be needed to complete this analysis.

5.1 NUMERICALLY SOLVING THE GOVERNING EQUATIONS OF FLUID DYNAMICS

Implementation of the governing equations of fluid dynamics, with appropriate initial and boundary conditions, into the framework of a numerical algorithm is indeed a daunting task.

The equations of fluid motion consisting of conservation of mass, momentum, and energy, can be written in a variety of forms. In general, we say that an equation can be placed in a conservative form provided it can be written as:

$$\vec{U}_t + \nabla \cdot \vec{F} = P \quad (5.1.1)$$

where $\vec{U}$ is the conserved quantity, $\vec{F} = \vec{F}(\vec{U})$ is an associated flux, and P corresponds to any source terms. $\vec{U}$ and P may be scalars, vectors, or tensors, and $\vec{F}$ is a quantity of higher order than $\vec{U}$ and P .

When the equations of fluid motion are written in a conservative form, it becomes possible to develop numerical schemes that are also in conservative form and obtain the correct jump conditions at discontinuities and shocks. These two properties are desirable in a numerical algorithm since they allow for an accurate representation of the physical reality given by the analytic equations. As we observed in Chapter 1 (Eqn. (1.2.11)), the Navier-Stokes equations can be written in the following compact form:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} - \left(\frac{\partial \mathbf{E}_v}{\partial x} + \frac{\partial \mathbf{F}_v}{\partial y} + \frac{\partial \mathbf{G}_v}{\partial z} \right) = \mathbf{H} \quad (5.1.2)$$

where, $\mathbf{Q}, \mathbf{E}, \mathbf{F}, \mathbf{G}, \mathbf{E}_v, \mathbf{F}_v$, and $\mathbf{G}_v$ are given in Eqns. (1.2.12) – (1.2.15) and $\mathbf{H}$ is the source term.

Amongst the general classes of discretization schemes used for solving PDEs numerically, three of the most common are finite-differencing, finite-element, and finite-volume methods, of which, the finite-volume technique is used in *FS*. Each of these schemes receives further examination in Appendix B.

As a technique for solving partial differential equations, the finite-volume method calculates the values of the conserved variables over finite control volumes. To explain how the technique is carried out, consider Eqn. (5.1.2). Integrating this expression over our control volume $V_{c.v.}$, we have:

$$\iiint_{V_{c.v.}} \left[\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} - \left(\frac{\partial \mathbf{E}_v}{\partial x} + \frac{\partial \mathbf{F}_v}{\partial y} + \frac{\partial \mathbf{G}_v}{\partial z} \right) \right] dV_{c.v.} = \iiint_{V_{c.v.}} \mathbf{H} dV_{c.v.} \quad (5.1.3)$$

In Eqn. (5.1.3) we apply Gauss' Divergence Theorem to the spatial derivatives to obtain:

$$\iiint_{V_{c.v.}} \frac{\partial \mathbf{Q}}{\partial t} dV_{c.v.} + \iint_{A_{c.v.}} ((\mathbf{E} - \mathbf{E}_v) \cdot \mathbf{n}_x + (\mathbf{F} - \mathbf{F}_v) \cdot \mathbf{n}_y + (\mathbf{G} - \mathbf{G}_v) \cdot \mathbf{n}_z) dA_{c.v.} = \iiint_{V_{c.v.}} \mathbf{H} dV_{c.v.} \quad (5.1.4)$$

The term $A_{c.v.}$ represents the surface area of the control volume. Assuming we have a spatially uniform distribution of $\mathbf{Q}$ and $\mathbf{H}$ within the control volume, and piecewise uniform distribution of fluxes over the surface faces A_x , A_y , and A_z of this control volume, Eqn. (5.1.4) yields:

$$\left(\frac{d\mathbf{Q}_{c.v.}}{dt} \right) V_{c.v.} + \sum_{c.v. \text{ faces}} (\mathbf{E} - \mathbf{E}_v) A_x + (\mathbf{F} - \mathbf{F}_v) A_y + (\mathbf{G} - \mathbf{G}_v) A_z = \mathbf{H}_{c.v.} V_{c.v.} \quad (5.1.5)$$

Here the summation is taken over the control volume faces. What remains in Eqn. (5.1.5) is a set of ordinary differential equations in time, and thus can be integrated by one of the many available methods. Following the addition of artificial diffusion, which was discussed in Chapter 3, Eqn. (5.1.5) becomes:

$$\left(\frac{d\mathbf{Q}_{c.v.}}{dt} \right) V_{c.v.} + \sum_{c.v. \text{ faces}} (\mathbf{E} - \mathbf{E}_v) A_x + (\mathbf{F} - \mathbf{F}_v) A_y + (\mathbf{G} - \mathbf{G}_v) A_z = \mathbf{H}_{c.v.} V_{c.v.} + D_{a.c.v.} \quad (5.1.6)$$

In this case the three-dimensional form of the artificial diffusion term, $D_{a.c.v.}$, consisting of second- and fourth-order differences of the solution vector in all three grid directions, is given by:

$$D_{a.c.v.} = \left(d_{i+\frac{1}{2},j,k} - d_{i-\frac{1}{2},j,k} \right) + \left(d_{i,j+\frac{1}{2},k} - d_{i,j-\frac{1}{2},k} \right) + \left(d_{i,j,k+\frac{1}{2}} - d_{i,j,k-\frac{1}{2}} \right) \quad (5.1.7)$$

As pointed out in Chapter 3, the artificial diffusion term is added to counteract spurious oscillations that may develop in numerical solutions, and to eliminate

decoupling of the pressure field over odd- and even-numbered nodes on the spatial grid. With this explicit introduction of diffusion terms, *FS* is able to damp oscillations in regions of high spatial gradients in flow variables, such as shock waves in high-speed flows. Once the various terms to be solved are formulated, the process of preconditioning, the subject of the next section, is implemented.

5.2 PSEUDO-TIME STEPPING AND PRECONDITIONING (ARTIFICIAL COMPRESSIBILITY)

Adopting the technique used in Chapter 3, we aid our discussion here by leaving the time derivative alone on the left hand side of Eqn. (5.1.6) and denoting the remaining right hand side terms by $-\mathbf{R}$. In doing so we have:

$$V_{c.v.} \left(\frac{d\mathbf{Q}_{c.v.}}{dt} \right) = -\mathbf{R} \quad (5.2.1)$$

Eqn. (5.2.1) can be discretized implicitly at time level $(n + 1)$ as follows:

$$\frac{d}{dt} \left((\mathbf{Q}_{c.v.})_{i,j,k}^{n+1} (V_{c.v.})_{i,j,k}^{n+1} \right) = -\mathbf{R}_{i,j,k}^{n+1} \quad (5.2.2)$$

Discretization of the time derivative term is through an implicit backward difference formula of second order accuracy, which yields (dropping the *c.v.* notation):

$$\frac{3}{2\Delta t} (\mathbf{Q}_{i,j,k}^{n+1} \mathbf{V}_{i,j,k}^{n+1}) - \frac{2}{\Delta t} (\mathbf{Q}_{i,j,k}^n \mathbf{V}_{i,j,k}^n) + \frac{1}{2\Delta t} (\mathbf{Q}_{i,j,k}^{n-1} \mathbf{V}_{i,j,k}^{n-1}) = -\mathbf{R}_{i,j,k}^{n+1} \quad (5.2.3)$$

This discretization is applicable at every time-step except for the first, where the following two-point first order scheme is used:

$$\frac{(\mathbf{Q}_{i,j,k}^{n+1} \mathbf{V}_{i,j,k}^{n+1}) - (\mathbf{Q}_{i,j,k}^n \mathbf{V}_{i,j,k}^n)}{\Delta t} = -\mathbf{R}_{i,j,k}^{n+1} \quad (5.2.4)$$

Eqn. (5.2.3) (as well as Eqn. (5.2.4) for the first time step) is non-linear and thus cannot be solved directly. To evade this problem, the time integration of these equations can be understood as a modified “pseudo-time marching” problem with the following altered residual term:

$$\mathbf{R}^*(\mathbf{Q}) = \mathbf{R}(\mathbf{Q}) - \left[\frac{3}{2\Delta t} (\mathbf{Q}\mathbf{V}) - \frac{2}{\Delta t} (\mathbf{Q}_{i,j,k}^n \mathbf{V}_{i,j,k}^n) + \frac{1}{2\Delta t} (\mathbf{Q}_{i,j,k}^{n-1} \mathbf{V}_{i,j,k}^{n-1}) \right] \quad (5.2.5)$$

In this case, the solution vector $\mathbf{Q}$ which satisfies the Equation $\mathbf{R}^*(\mathbf{Q}) = 0$ is the $\mathbf{Q}^{n+1}$ vector we are looking for. Thus, in order to obtain this solution we can reformulate the problem at each real time step as the modified steady state problem shown in Eqn. (5.2.6), solved in pseudo-time t^* :

$$\frac{d}{dt^*} ((\mathbf{Q}_{c,v})_{i,j,k}^{n+1} (\mathbf{V}_{c,v})_{i,j,k}^{n+1}) = -(\mathbf{R}^*)_{i,j,k}^{n+1} \quad (5.2.6)$$

To Eqn. (5.2.6) one can apply the several convergence techniques used for steady state calculations. Consequently, the solution of Eqn. (5.2.2) is equivalent to marching Eqn. (5.2.6) to a steady state in pseudo-time. As explained in Chapter 3, the *FS* and *EE_Solver* algorithms implement multistage Runge-Kutta time-stepping to integrate the discrete equations. Furthermore, in the *FS* algorithm local time-stepping is used so that the solution at each grid point is advanced at the maximum time step

allowed by stability. Thus, for each real time step, any given number of pseudo-time steps can be taken, a value that is specified by the user.

The pseudo-time derivative is preconditioned to extend the range of applicability of the algorithm to low Mach-number flows. The exact nature of this preconditioning is described in the following subsection.

5.2.1 ARTIFICIAL COMPRESSIBILITY

The standard hyperbolic methods used to solve the discretized governing equations (5.1.6) are not effective in the limit of incompressible flow. At this limit, the sound waves dominate the system and the governing equations become poorly conditioned for numerical solution. A possible way around this difficulty is the coupling of the mass and momentum conservation equations through the concept of artificial compressibility (e.g. Chorin, 1967; Rizzi and Eriksson, 1985). In this approach, the original continuity equation for incompressible flow,

$$\nabla \cdot \mathbf{u} = 0$$

is modified by adding an artificial compressibility term:

$$\left(\frac{1}{\rho_d} \right) \left(\frac{\partial p}{\partial t} \right) + c_a^2 \nabla \cdot \mathbf{u} = 0 \quad (5.2.1.1)$$

where c_a acts as a compressibility parameter and ρ_d is the constant density of the flow. Eqn. (5.2.1.1) can be deduced from the continuity equation for compressible flows by using the following equation of state:

$$p = c_a^2 \rho_d \quad (5.2.1.2)$$

The modified inviscid system of partial differential equations consisting of mass and momentum conservation principles now becomes:

$$\frac{\partial \mathbf{Q}}{\partial t} + \boldsymbol{\Omega} \cdot \left[\frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} \right] = \mathbf{0} \quad (5.2.1.3)$$

where,

$$\mathbf{Q} = \begin{bmatrix} \frac{p}{\rho_d} \\ u \\ v \\ w \end{bmatrix} \quad \boldsymbol{\Omega} = \begin{bmatrix} c_a^2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.2.1.4)$$

$$\mathbf{E} = \begin{bmatrix} u \\ u^2 + \frac{p}{\rho_d} \\ uv \\ uw \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} v \\ uv \\ v^2 + \frac{p}{\rho_d} \\ vw \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} w \\ uw \\ vw \\ w^2 + \frac{p}{\rho_d} \end{bmatrix}$$

In this way, the pressure acts like a relaxation parameter to satisfy the continuity equation. As a result of this preconditioning, the Navier-Stokes equations remain hyperbolic in time regardless of whether they are elliptic or hyperbolic in space. The solution is then marched in this pseudo-time dimension to a steady state, hence artificial compressibility is only present during the pseudo-transient state.

A preconditioning similar to that proposed by Turkel (1987) is used in *FS*. In this method, the pseudo-time derivative of $\mathbf{Q}$ is written as:

$$\frac{\partial \mathbf{Q}}{\partial t_p} = \begin{bmatrix} \left(\frac{a_1}{c_a}\right)^2 \frac{\partial p}{\partial t_p} \\ \left(\frac{a_2}{c_a}\right)^2 u \frac{\partial p}{\partial t_p} + \frac{\partial(\rho u)}{\partial t_p} \\ \left(\frac{a_3}{c_a}\right)^2 v \frac{\partial p}{\partial t_p} + \frac{\partial(\rho v)}{\partial t_p} \\ \left(\frac{a_4}{c_a}\right)^2 w \frac{\partial p}{\partial t_p} + \frac{\partial(\rho w)}{\partial t_p} \\ \left(\frac{a_5}{c_a}\right)^2 e_t \frac{\partial p}{\partial t_p} + \frac{\partial(\rho e_t)}{\partial t_p} \end{bmatrix} \quad (5.2.1.5)$$

c_a may be thought of as the artificial speed of pressure wave propagation and is typically scaled on the local flow velocity or to a reference velocity representative of the overall fluid flow in question. Moreover, the $a_1, \dots, a_5$ terms are scaling coefficients. The choice of c_a and the scaling coefficients vary depending on the nature of the problem and many have attempted to place particular bounds on these choices. For example, Bet *et al.* (1995) examined the sensitivity of solution convergence of a two-dimensional laminar boundary layer flow for various artificial compressibility settings. The results are shown in Fig. 5.2.1.1:

Figure 5.2.1.1 – Convergence sensitivity of a laminar boundary-layer simulation to artificial compressibility. Here $\beta = a_1 = a_2 = a_3 = a_4 = a_5$, reproduced from Bet *et al.* (1995)

Simulation parametric studies involving the parameter c_a are given in Appendix D.

Finally, the time-step size is adjusted locally to its maximum value, limited by the numerical stability as per the following relation:

$$\frac{1}{(\Delta t_p)_{i,j,k}} \geq \left(\frac{1}{\Delta t_c} + \frac{1}{\Delta t_d} \right)_{i,j,k} = \left(\frac{\sum \lambda}{K_c V_{c.v.}} + \frac{1}{K_d V_{c.v.}^2} \sum \frac{\mu + \mu_t}{\rho} A_{c.v.}^2 \right)_{i,j,k} \quad (5.2.1.6)$$

where K_c and K_d are coefficients that dictate the inviscid (convective) and viscous (diffusive) limits of the artificial time step, respectively. Local sizing of the time step warps the transient flow development in the pseudo-time dimension. This however does not pose a concern since the steady solution is independent of the transient path.

The preceding sections have provided a quick overview of some of the fundamental aspects of CFD, and are by no means all-inclusive. Moreover, the topics covered

from the beginning of Chapter 5 until now, and in greater detail in Appendix B, are not unique to the *FS* algorithm, as it is one among an assortment of numerical solvers. The commonality among fluid flow solvers can be found precisely in their collective dependency on various techniques (e.g. multigrid methods, preconditioning, artificial diffusion, etc.) in order to obtain accurate and efficient results, as opposed to the manner in which these techniques are carried out. As a consequence, researchers may exercise their freedom to devise “unique” algorithms in as far as they are capable of appropriately solving their problem(s) of interest (as in the case of *EE_Solver*), while these same scientists retain the capability of fully understanding a colleague’s distinct, albeit in the same field, algorithm.

As we have seen, the standard hyperbolic methods implemented within numerical algorithms designed to solve the compressible equations of motion are not effective in the limit of incompressible flow. Consequently, many have incorporated the notion of artificial compressibility, or preconditioning, to circumvent the problems caused by the domination of sound waves at the incompressible limit. In the next section, we examine how the notion of artificial compressibility is also compatible with the buffer zone method.

5.3 THE BUFFER ZONE METHOD IN THE CONTEXT OF ARTIFICIAL COMPRESSIBILITY

In the previous section we observed that the preconditioning used in *FS* yields the

following expression for $\frac{\partial \mathbf{Q}}{\partial t}$:

$$\frac{\partial \mathbf{Q}}{\partial t_p} = \begin{bmatrix} \left(\frac{a_1}{c_a}\right)^2 \frac{\partial p}{\partial t_p} \\ \left(\frac{a_2}{c_a}\right)^2 u \frac{\partial p}{\partial t_p} + \frac{\partial(\rho u)}{\partial t_p} \\ \left(\frac{a_3}{c_a}\right)^2 v \frac{\partial p}{\partial t_p} + \frac{\partial(\rho v)}{\partial t_p} \\ \left(\frac{a_4}{c_a}\right)^2 w \frac{\partial p}{\partial t_p} + \frac{\partial(\rho w)}{\partial t_p} \\ \left(\frac{a_5}{c_a}\right)^2 e_i \frac{\partial p}{\partial t_p} + \frac{\partial(\rho e_i)}{\partial t_p} \end{bmatrix} \quad (5.3.1)$$

where c_a is the artificial speed of pressure wave propagation and $a_1, \dots, a_5$ are scaling coefficients. In order to study the integration of the buffer zone technique into full Navier-Stokes algorithms such as *FS*, we need only consider the modified solution vector shown in Eqn. (5.3.1). This follows from the fact that the viscous terms are unaltered in the preconditioned system.

In order to justify the use of the buffer zone technique on the preconditioned form of the equations, we must show that the method of artificial compressibility maintains the hyperbolic nature of problems, and secondly, that this artificial component is only present during the transient state (that is, the final results do not depend on the artificial compressibility). The latter is addressed in the ensuing subsection.

To show the former, consider again Eqn. (5.2.1.3):

$$\frac{\partial \mathbf{Q}}{\partial t} + \Omega \cdot \left[\frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} \right] = \mathbf{0} \quad (5.3.2)$$

where,

$$\mathbf{Q} = \begin{bmatrix} \frac{p}{\rho_d} \\ u \\ v \\ w \end{bmatrix}$$

$$\mathbf{\Omega} = \begin{bmatrix} c_a^2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.3.3)$$

$$\mathbf{E} = \begin{bmatrix} u \\ u^2 + \frac{p}{\rho_d} \\ uv \\ uw \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} v \\ uv \\ v^2 + \frac{p}{\rho_d} \\ vw \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} w \\ uw \\ vw \\ w^2 + \frac{p}{\rho_d} \end{bmatrix}$$

The primitive variable form of Eqn. (5.3.2) is:

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{K}_1 \frac{\partial \mathbf{Q}}{\partial x} + \mathbf{K}_2 \frac{\partial \mathbf{Q}}{\partial y} + \mathbf{K}_3 \frac{\partial \mathbf{Q}}{\partial z} = 0 \quad (5.3.4)$$

with,

$$\mathbf{K}_1 = \mathbf{\Omega} \frac{\partial \mathbf{E}}{\partial \mathbf{Q}} = \begin{bmatrix} 0 & c_a^2 & 0 & 0 \\ 1 & 2u & 0 & 0 \\ 0 & v & u & 0 \\ 0 & w & 0 & u \end{bmatrix} \quad \mathbf{K}_2 = \mathbf{\Omega} \frac{\partial \mathbf{F}}{\partial \mathbf{Q}} = \begin{bmatrix} 0 & 0 & c_a^2 & 0 \\ 0 & v & u & 0 \\ 1 & 0 & 2v & 0 \\ 0 & 0 & w & v \end{bmatrix}$$

(5.3.5)

$$\mathbf{K}_3 = \mathbf{\Omega} \frac{\partial \mathbf{G}}{\partial \mathbf{Q}} = \begin{bmatrix} 0 & 0 & 0 & c_a^2 \\ 0 & w & 0 & u \\ 0 & 0 & w & v \\ 1 & 0 & 0 & 2w \end{bmatrix}$$

The primitive variable system is called hyperbolic at the point $(x, y, z, t, \mathbf{Q})$ provided there exists a non-singular matrix $\mathbf{T}_1(\varsigma, \chi, \tau)$ that diagonalizes the linear combination $\mathbf{D}_1 = \varsigma \mathbf{K}_1 + \chi \mathbf{K}_2 + \tau \mathbf{K}_3$ (Chorin, 1967). That is:

$$\mathbf{T}_1^{-1} \mathbf{D}_1 \mathbf{T}_1 = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix} \quad (5.3.6)$$

where the eigenvalues, λ of $\mathbf{D}_1$ are real and the norms of $\mathbf{T}_1$ and $\mathbf{T}_1^{-1}$ are uniformly bounded for arbitrary real numbers ς, χ , and τ . The eigenvalues of $\mathbf{D}_1$ are found, using the definitions $U = \varsigma u + \chi v + \tau w$ and $\phi = U^2 + c_a^2(\varsigma^2 + \chi^2 + \tau^2)$, to be $\lambda_1 = \lambda_2 = U$, $\lambda_3 = U + \phi$, $\lambda_4 = U - \phi$. Furthermore, these eigenvalues are always real. It is important to note that the range between the smallest and largest eigenvalues can be adjusted according to the value of the parameter c_a , an important fact that regulates the efficiency with which simulations using the method of artificial compressibility perform.

The set of linearly independent right eigenvectors of $\mathbf{D}_1$ are the columns of $\mathbf{T}_1$.

Furthermore, the matrix $\mathbf{T}_1$ has the following form:

$$\mathbf{T}_1 = \begin{bmatrix} 0 & 0 & c_a^2 & -c_a^2 \\ -\tau & -\chi & u(U + \phi) + \varsigma c_a^2 & u(U - \phi) + \varsigma c_a^2 \\ 0 & \varsigma & u(U + \phi) + \chi c_a^2 & u(U - \phi) + \chi c_a^2 \\ \varsigma & 0 & u(U + \phi) + \tau c_a^2 & u(U - \phi) + \tau c_a^2 \end{bmatrix} \quad (5.3.7)$$

Its inverse is formed from the left eigenvectors of $\mathbf{D}_1$:

$$\mathbf{T}_1^{-1} = \begin{bmatrix} \frac{\tau U - (\zeta^2 + \chi^2 + \tau^2)w}{\zeta \phi^2} & -\frac{wU + \tau c_a^2}{\phi^2} & -\frac{\chi(wU + \tau c_a^2)}{\zeta \phi^2} & \frac{(\zeta u + \chi v)U(\zeta^2 + \chi^2)c_a^2}{\zeta \phi^2} \\ \frac{\chi U - (\zeta^2 + \chi^2 + \tau^2)v}{\zeta \phi^2} & -\frac{vU + \chi c_a^2}{\phi^2} & \frac{(\zeta u + \tau w)U + (\zeta^2 + \tau^2)c_a^2}{\zeta \phi^2} & -\frac{\tau(vU + \chi c_a^2)}{\zeta \phi^2} \\ -\frac{U - \phi}{2\phi^2 c_a^2} & \frac{\zeta}{2\phi^2} & \frac{\chi}{2\phi^2} & \frac{\tau}{2\phi^2} \\ -\frac{U + a}{2\phi^2 c_a^2} & \frac{\zeta}{2\phi^2} & \frac{\chi}{2\phi^2} & \frac{\tau}{2\phi^2} \end{bmatrix} \quad (5.3.8)$$

From this we conclude that $\mathbf{T}_1$ and $\mathbf{T}_1^{-1}$ are uniformly bounded provided $|u|$, $|v|$, and $|w|$ are bounded, from which we may establish that the system is hyperbolic.

5.3.1 ARTIFICIAL COMPRESSIBILITY IN THE STEADY STATE LIMIT

Chorin (1967) explains that it is not essential for the solution of the differential system (5.1.2) to tend to a steady limit, as long as the solution of the discretized system does. It is however believed that the solution of Eqn. (5.3.1) does tend to a steady limit, at least in the absence of external forces, under quite general conditions. In particular, Chorin shows that the velocity components within the artificial terms tend to zero, while the pressure tends to a constant value in the steady state limit. Nonetheless, many algorithms over the years have confirmed what theory has proposed; in the steady state limit the equations become independent of the artificial compressibility (Chorin (1967), Rizzi and Eriksson (1985), Yaras (2002)).

Consequently, the addition of artificial compressibility and the implementation of the buffer zone technique to the full Navier-Stokes equations in compact flux vector form (Eqn. (5.1.2)) results in the following transformed set of equations:

$$\left[\frac{\partial \mathbf{Q}}{\partial t_p} \right]_{U'} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} = \mu \left[\frac{\partial \mathbf{E}_v}{\partial x} + \frac{\partial \mathbf{F}_v}{\partial y} + \frac{\partial \mathbf{G}_v}{\partial z} + \mathbf{H} \right] \quad (5.3.1.1)$$

where,

$$\left[\frac{\partial \mathbf{Q}}{\partial t_p} \right]_{U'} = \begin{bmatrix} \left(\frac{a_1}{c_a} \right)^2 \left[\frac{\partial p}{\partial t_p} + U'_1 \frac{\partial p}{\partial x} \right] \\ \left(\frac{a_2}{c_a} \right)^2 u \frac{\partial p}{\partial t_p} + \frac{\partial(\rho u)}{\partial t_p} + U'_2 \frac{\partial(\rho u)}{\partial x} \\ \left(\frac{a_3}{c_a} \right)^2 v \frac{\partial p}{\partial t_p} + \frac{\partial(\rho v)}{\partial t_p} + U'_3 \frac{\partial(\rho v)}{\partial x} \\ \left(\frac{a_4}{c_a} \right)^2 w \frac{\partial p}{\partial t_p} + \frac{\partial(\rho w)}{\partial t_p} + U'_4 \frac{\partial(\rho w)}{\partial x} \\ \left(\frac{a_5}{c_a} \right)^2 e_i \frac{\partial p}{\partial t_p} + \frac{\partial(\rho e_i)}{\partial t_p} + U'_5 \frac{\partial(\rho e_i)}{\partial x} \end{bmatrix} \quad (5.3.1.2)$$

$\mathbf{E}_v, \mathbf{F}_v$, and $\mathbf{G}_v$ are given in Eqns. (1.2.13) – (1.2.15). It is important to note that μ now multiplies the viscous terms in keeping with original formulation of Streett (1986). Eqn. (5.3.1.1) has included in its time derivative term, both elements of the artificial compressibility technique and the spatial derivatives from the buffer zone technique. For brevity, the original temporal derivative is not included in this formulation and does not affect the conclusions presented herein.

So as to complete this derivation, the components $U'_1 - U'_5$ of $\mathbf{U}'$ and μ must be selected to ensure the desired conditions at the outer edge of the buffer zone are met.

In the next two sections, we attempt to generalize this selection process, hence completing the current work and making the buffer zone technique completely accessible.

5.4 GENERALIZING THE DEFINITIONS OF U' AND μ FOR ARBITRARY THREE-DIMENSIONAL DOMAINS

Among the most notable omissions found in the literature regarding buffer zone techniques thus far, has been the lack of generalization of the method itself, including the definitions of U' and μ . This task is indeed beneficial, for as yet, the majority of applications have focused on fairly simplified domains. In order to investigate further this issue, consider once again the computational domain employed in the study of Ta'asan and Nark (1995):

Figure 5.4.1 – A schematic of the buffer zone computational domain used in the work of Ta'asan and Nark (1995)

In regards to both Ta'asan *et al*'s choice of computational domain, as well as that considered in the current research (Chapter 4), a buffer zone was implemented strictly in the streamwise x -direction. This choice was guided by the fact that each flow exhibited velocity solely in the x coordinate direction. Nonetheless, buffer zones may be appended to any pertinent boundary, and in any required proportion. Fig. 5.4.2 depicts three of an infinite number of possibilities for affixing buffer zones to an arbitrary two-dimensional domain: $x_{\min} \leq x \leq x_{\max}$, $y_{\min} \leq y \leq y_{\max}$

Figure 5.4.2 – Theoretical buffer zone additions to an arbitrary computational domain,
 $x_{\min} \leq x \leq x_{\max}$, $y_{\min} \leq y \leq y_{\max}$

As both Ta'asan *et al.*'s study and the current work involve flows whose velocity components are strictly in the streamwise direction, it seems only intuitive to define U' and μ as in Eqn. (3.4.5).

We then ask ourselves, what if the flow in question is not in the streamwise direction? Consider Fig. 5.4.3 showing an arbitrary computational domain similar to the one used by Ta'asan *et al.* Let us assume that for some arbitrary fluid flow problem defined in this region, a velocity with non-zero x and y components is encountered at the outflow boundary. With this assumption, it is obvious then that the velocity at the outlet is not in the streamwise direction (the x coordinate direction in this case).

Figure 5.4.3 – An arbitrary computational domain where, for some non-specified problem, the fluid velocity at the point (x_e, y_e) lying on the interface of the computational outflow boundary and the buffer zone has both an x and y component

Of course, the ideal and most fundamental setting to carry out our extension of the buffer zone technique to three-dimensions is within an algorithm that writes the momentum equations in the streamwise direction. In this way, modifications of the flow field with the parameters U' and μ are natural extensions of the one-

dimensional cases previously discussed. Unfortunately, in many cases it is not feasible to write the momentum equations in the streamwise direction, especially when dealing with intricate computational domains. Even when one performs a coordinate transformation, as shown in Fig. 5.4.4, in order to work with a simpler computational domain, the problem is not evaded. Such transformations relocate the fluid flow problem of interest from a physical (x, y) domain to a less cumbersome computational domain, by way of an appropriate coordinate transformation:

$x = x(\xi, \eta)$, $y = y(\xi, \eta)$ (one will often find the (ξ, η) notation used within textbooks dedicated to this topic). Irrespective of this fact, a coordinate transformation does not guarantee that the momentum equations can be written in the streamwise direction.

Figure 5.4.4 – A schematic of a coordinate transformation from a physical domain (x, y) to a computational domain (ξ, η)

Since the goal of the buffer zone technique is, generally speaking, to accelerate waves at the outflow boundary to supersonic speeds, in three-dimensions we may define U' as follows:

$$U' = \beta_x(x - x_e)^m + \beta_y(y - y_e)^m + \beta_z(z - z_e)^m \quad (5.4.1)$$

Firstly, from the results of Chapter 4, we shall assume that we are application a uniform U' is sufficient. Of course, our discussion here allows for individual U' definitions for each equation shown in (5.3.1.2). Although Eqn. (5.4.1) appears to be a straightforward extension of the original definition (Eqn. (2.3.1.33)), the choice of the β terms require some discussion.

In order to choose β_x, β_y , and β_z correctly at some point (x, y, z) , we consider the velocity component with the largest magnitude. Let us assume that at this point, all three velocity components are non-zero. We would like the magnitude of the velocity to be accelerated to supersonic speeds in the correct “artificial” direction. However, rather than arbitrarily choosing β terms with the risk of insufficiently treating the problem, we identify the velocity component having the largest magnitude. Without a loss of generality, let us assume that this component is u . Our next step is to choose β_x large enough so that U' is supersonic at the outflow boundary. Even when $\beta_y = \beta_z = 0$, this choice of β_x guarantees that Eqn. (5.4.1) gives supersonic conditions at the outflow boundary. Yet, in order to maintain the direction of the flow, we may then scale the remaining β terms (β_y and β_z in this case) up to the same β_x value. Consequently, the correct flow direction is maintained and uniformity in the coefficient matrix is not lost. Of course situations may arise where a particular flow direction might not require modification. Indeed, further numerical simulations will be needed to test the validity of definition (5.4.1).

The full extension to three-dimensions is thus:

$$\left[\frac{\partial Q}{\partial t_p} \right]_{U'} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = \mu \left[\frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y} + \frac{\partial G_v}{\partial z} + H \right] \quad (5.4.2)$$

with,

$$\left[\frac{\partial Q}{\partial t_p} \right]_{U'} = \left[\begin{aligned} & \left(\frac{a_1}{c_a} \right)^2 \left[\frac{\partial p}{\partial t_p} + \left\{ \beta_x (x - x_e)^m \frac{\partial p}{\partial x} + \beta_y (y - y_e)^m \frac{\partial p}{\partial y} + \beta_z (z - z_e)^m \frac{\partial p}{\partial z} \right\} \right] \\ & \left(\frac{a_2}{c_a} \right)^2 u \frac{\partial p}{\partial t_p} + \frac{\partial(\rho u)}{\partial t_p} + \left\{ \beta_x (x - x_e)^m \frac{\partial(\rho u)}{\partial x} + \beta_y (y - y_e)^m \frac{\partial(\rho u)}{\partial y} + \beta_z (z - z_e)^m \frac{\partial(\rho u)}{\partial z} \right\} \\ & \left(\frac{a_3}{c_a} \right)^2 v \frac{\partial p}{\partial t_p} + \frac{\partial(\rho v)}{\partial t_p} + \left\{ \beta_x (x - x_e)^m \frac{\partial(\rho v)}{\partial x} + \beta_y (y - y_e)^m \frac{\partial(\rho v)}{\partial y} + \beta_z (z - z_e)^m \frac{\partial(\rho v)}{\partial z} \right\} \\ & \left(\frac{a_4}{c_a} \right)^2 w \frac{\partial p}{\partial t_p} + \frac{\partial(\rho w)}{\partial t_p} + \left\{ \beta_x (x - x_e)^m \frac{\partial(\rho w)}{\partial x} + \beta_y (y - y_e)^m \frac{\partial(\rho w)}{\partial y} + \beta_z (z - z_e)^m \frac{\partial(\rho w)}{\partial z} \right\} \\ & \left(\frac{a_5}{c_a} \right)^2 e_i \frac{\partial p}{\partial t_p} + \frac{\partial(\rho e_i)}{\partial t_p} + \left\{ \beta_x (x - x_e)^m \frac{\partial(\rho e_i)}{\partial x} + \beta_y (y - y_e)^m \frac{\partial(\rho e_i)}{\partial y} + \beta_z (z - z_e)^m \frac{\partial(\rho e_i)}{\partial z} \right\} \end{aligned} \right] \quad (5.4.4)$$

and,

$$\mu = (1 - \eta_x (x - x_e)^m) + (1 - \eta_y (y - y_e)^m) + (1 - \eta_z (z - z_e)^m) \quad (5.4.5)$$

Once again we see the fusing of the artificial compressibility technique and the buffer zone method.

The buffer zone addition can perhaps best be seen from a slight alteration to Eqn.

(5.4.2), shown below:

$$\frac{\partial Q}{\partial t_p} + \mathbf{U}' \frac{\partial Q}{\partial \mathbf{x}} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = \mu \left[\frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y} + \frac{\partial G_v}{\partial z} + \mathbf{H} \right] \quad (5.4.6)$$

where $\frac{\partial Q}{\partial t_p}$ is as originally defined in Eqn. (5.2.1.5) and,

$$\mathbf{U}' = \begin{bmatrix} U'_{buffer} & 0 & 0 & 0 & 0 \\ 0 & U'_{buffer} & 0 & 0 & 0 \\ 0 & 0 & U'_{buffer} & 0 & 0 \\ 0 & 0 & 0 & U'_{buffer} & 0 \\ 0 & 0 & 0 & 0 & U'_{buffer} \end{bmatrix} \quad (5.4.7)$$

$$\mu = \begin{bmatrix} \mu_{buffer} & 0 & 0 & 0 & 0 \\ 0 & \mu_{buffer} & 0 & 0 & 0 \\ 0 & 0 & \mu_{buffer} & 0 & 0 \\ 0 & 0 & 0 & \mu_{buffer} & 0 \\ 0 & 0 & 0 & 0 & \mu_{buffer} \end{bmatrix} \quad (5.4.8)$$

with,

$$U'_{buffer} = (\beta_x (x - x_e)^m, \beta_y (y - y_e)^m, \beta_z (z - z_e)^m) \quad (5.4.9)$$

$$\mu_{buffer} = ((1 - \eta_x (x - x_e)^m), (1 - \eta_y (y - y_e)^m), (1 - \eta_z (z - z_e)^m)) \quad (5.4.10)$$

As before, stability is maintained through the contribution of μ , whose selection is in the same vein as that of $\mathbf{U}'$.

Therefore, the above derivation reinforces the opinion of advocates who applaud the buffer zone technique for its simplicity in regards to numerical implementation. This fact, when combined with the results presented in Chapter 4, places this technique among the more promising within the CFD community. In the final chapter of this work, we reiterate on the significant results obtained thus far and indicate where

future research might be beneficial, in the hopes of further improving the buffer zone technique's position as a viable resource for CFD researchers.

6.0 CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE STUDIES

Clearly a researcher endeavouring to follow the efforts of mathematicians and engineers in the field of computational fluid dynamics must be attentive to the many competing schools of thought present today. These difficulties are especially prevalent when designing an algorithm intended to solve a wide range of fluid dynamics problems, both efficiently and accurately. CFD algorithms, or CFD codes, must not only carefully integrate the various numerical “weapons” used to carry out this solution process, but also require constant revision in order to maximize the balance between computational time and applicability.

The struggle to attain equilibrium in regards to issues such as computation time and applicability is widespread. We received some indication of this effort in Chapter 2 when we outlined the current state-of-the-art in outflow boundary conditions, with a particular focus on reduction of reflection of outgoing Riemann invariants. From this body of knowledge it is not difficult to see that the very nature of computational fluid dynamics demands that researchers approach novel ideas with a fair amount of reservation, for what might be applicable in one context, could certainly fall short in another. In the particular area of outflow boundary reflection, the following general issues were explored:

- Demonstration of reflection problems in three-dimensional Navier-Stokes Solver (Section 4.1).
- Demonstration of similar reflection problems in one-dimensional Euler equation solver (Section 4.2).

- Demonstration of the sensitivity of the reflection problem to the accuracy of initial and boundary conditions (Section 4.2 i.e. $\rho_{Npts} = 1.1575$ vs. $\rho_{Npts} = 0.5$).

In response to these reflection issues, the following developments were made:

- The first-time application of the buffer zone technique of Ta'asan and Nark (1995) to the non-linear Euler equations, both in characteristic and non-characteristic form (Section 3.1).
- For the first time we present optimal values for buffer zone parameters such as U' , buffer width, and buffer zone node counts. We also provided convincing evidence showing the close connection between the buffer zone node count and the choice of computational stencil. Consequently, the values obtained herein, particularly 5 nodes in the buffer zone and $U'_{Npts} = 10$ (transient simulations), should not be considered universal. Rather they give an indication as to how such parameters are obtained. On the other hand, the result which might have broader use is our conclusion that the largest U' was also the optimal value of this parameter (Section 4.3.2). These results place a distinct separation between buffer zone techniques such as Ta'asan *et al.*'s and absorbing techniques, such as that of Hayder *et al.*
- We demonstrated the applicability of the buffer zone technique to both transient and steady-state simulations (Chapter 4).
- Finally, we outlined how one would apply the buffer zone technique in the context of a full three-dimensional Navier-Stokes solver (Chapter 5).

6.1 FUTURE RESEARCH

As insinuated earlier, the aspects of the buffer zone technique explored in the simulations of Chapter 4 can certainly benefit from similar higher-dimensional studies. In this way, three-dimensional simulations that vary node counts within the buffer zone and seek optimal buffer zone widths either fortify the results found in the current work or shed light on more pertinent issues. Some specific areas of research that might serve to improve the application of the buffer zone technique include:

- Test Case 1 or other more elaborate computational domains with fluids of varying Mach number. In this way, one can further elucidate whether the buffer zone technique is sensitive to Mach number. From the work herein, it appears that there will be less effects than in the absorbing techniques discussed earlier. However, buffer zone parameters will have to be chosen differently in order to account for the different velocities.
- Another potential area of research deals with the use of different computational stencils. Such studies will certainly shed more light on the issue of optimal node count in the buffer zone.
- In regards to our three-dimensional extension of the buffer zone presented in Chapter 5, numerical simulations can be conducted in order to determine assess its efficiency. Cases such as that described in Chapter 5, in which particular velocity components need no modification and yet receive it, will definitely help researchers along the path towards a more optimal implementation of this technique.

A final point of interest, which has received potentially no documentation, deals with the extension of computational domains to create physical buffer zones. As opposed to the one and two-dimensional problems studied herein, in which the buffer zones were natural extensions of the domains (i.e. extension in the x direction), it appears that no strict protocol exists, nor is it expected to. In fact, given the infinite number of computational domain possibilities, the generalization of buffer zone selection borders on the level of unfeasible. What can be said is that the most "natural" extension in these higher dimensional problems appears to be a continuation of the computational domain in the streamwise flow direction. Considering this statement briefly will lead one to see that for most practical problems, such an extension is quite instinctive. Exceptions, such as swirling flow problems where there is possibly little streamwise fluid flow, also present themselves as areas of interesting research.

ACKNOWLEDGMENTS

It would only be fitting to begin with the words of the apostle uttered nearly two thousand years ago, “*in omnibus gratias agite haec enim voluntas Dei est in Christo Iesu in omnibus vobis*” (1 Thessalonians 5:18).

For my co-supervisors who patiently endured two years of “sinusoidal-like” dealings, Dr. D. Amundsen and Dr. M.I. Yaras, words of appreciation do not suffice. I walk away from this endeavour, not with regrets, but with a wealth of knowledge and experience. Though some battles were lost early on, the outcome was far better than I could have imagined: I gained two admirable friends.

I extend a special thank you to Dr. O. Chandna who introduced me to the subject of fluid dynamics.

In *gladly* keeping my promise, I thank Ashley and Megan for their “Ashley Surprises” and chicken breast – this food had the contradictory effect of causing me to “stumble”, while at the same time giving me the necessary fuel to go on.

I thank those five special people (Fr. Paul Cormier, Dr. Miguel Rodriguez, Salvador Rego, Dr. Jean Rheaume, and Dr. Ernest Capparos) who for two years helped me find what was missing in my life.

Mathieu Lemire, many days went by much smoother because of the “Backstreet Boys” performances and walks along the canal. Thank you my good friend.

Kevin George, Ryan George, Jason Laba, Mike Raffoul, Mariusz Buchic, and Jessica Laba – perhaps your understanding of boundary conditions in CFD problems is minimal, however, I wouldn't have it any other way. My many years of gratitude fall quite short of what you deserve.

To the Prof. Yaras "CFD Family" – Steve, Craig, Michel, Samira. Each of you have had your own place in this work, and I assure you, it goes far beyond CFD, intermittency, vortex generators, blood pressure, or neural networks. I hope this is only the beginning.

I thank Mr. and Mrs. Richard and Louise Bastien for having chosen to "adopt" this struggling master's student - "Seeing the end of the road" is one thing, but having the love and support of family is another. Pascal, my good friend, you and I both know that it wasn't luck that kept me in Ottawa to work on this thesis. Thank you (*C'est dans le besoin qu'on reconnaît ses vrais amis*).

Wissam, my brother, I thank you for your silent, but influential support. Just in case you weren't sure about it, I look up to you (not because you're taller than I) and I am proud of you.

My parents, how do I go about separating the two of you in order to thank you? Dad, it wasn't just driving and swimming that you taught me, but more importantly, what really matters in life. I hope that I can be half the man that you are. Mom, your fortitude, self-sacrifice, and loving heart have marked me throughout my life, including the last two years where I felt I've needed them the most.

REFERENCES

- Abu-Ghannam, B.J. and Shaw, R., 1980, "Natural Transition of Boundary Layers-The Effects of Turbulence, Pressure Gradient, and Flow History," *Journal Mechanical Engineering Science*, Vol.22, No.5, pp. 213 – 228.
- American Institute of Aeronautics and Astronautics, 1998, "Guide for the Verification and Validation of Computational Fluid Dynamics Simulations," AIAA Paper G-077-1998.
- Anderson, D.A., Tannehill, J.C., and Pletcher, R.H., 1984, *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation.
- Armud, K. K., 1985, "Tip Leakage in a Planar Cascade of Turbine Blades," M. Eng. Thesis, Department of Mechanical and Aerospace Engineering, Carleton University, Ottawa, Canada.
- Arnone, A., Meng-Sing, L., and Povinelli, L.A., 1993, "Multigrid Time-Accurate Integration of Navier-Stokes Equations," AIAA Paper 93-3361.
- Berenger, J., 1994, "A Perfectly Matched Layer for the Absorption of Electromagnetic Waves," *Journal of Computational Physics*, Vol.114, pp. 185 – 200.
- Bittencourt, M., Nogueira Jr., A., and Feijóo, R.A., 2000, "Multigrid Methods Applied to the H and P Versions of the FEM," European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2000, Barcelona.
- Bluestein, D., Rambod, E., and Gharib, M., 2000, "Vortex Shedding as a Mechanism for Free Emboli Formation in Mechanical Heart Valves," *Journal of Biomechanical Engineering*, Vol. 122, pp. 125 – 134.
- Brandt, A., 1977, "Multi-Level Adaptive Solutions to Boundary-Value Problems," *Mathematics of Computation*, Vol.31, No.138, pp. 333 – 390.

- Briggs, W.L., 1987A *Multigrid Tutorial*, Lancaster Press, Lancaster, Pennsylvania, SIAM.
- Buelow, P.E.O., Venkateswaran, S., and Merkle, C.L., 1994, "Effect of Grid Aspect Ratio on Convergence," *AIAA Journal*, Vol.32, No.12, pp. 2401 – 2408.
- Cerjan, C., Kosloff, D., Kosloff, R., and Reshef, M., 1985, "A Nonreflecting Boundary Condition for Discrete Acoustic and Elastic Wave Equations," *Geophysics*, Vol. 50, pp. 705 – 708.
- Chima, R., 1984, "Inviscid and Viscous Flows in Cascades with an Explicit Multiple-Grid Algorithm," *AIAA Journal*, Vol.23, No.10, pp. 1556 – 1563.
- Chorin, A.J., 1967, "A Numerical Method for Solving Incompressible Viscous Flow Problems," *Journal of Computational Physics*, Vol.2, pp. 12 – 26.
- Colonius, T., Lele, S.K., and Moin, P., 1993, "Boundary Conditions for Direct Computation of Aerodynamic Sound Generation," *AIAA Journal*, Vol. 31, No. 9, pp. 1574 – 1582.
- Darmofal, D.L., 1998, "Eigenmode Analysis of Boundary Conditions for the One-Dimensional Preconditioned Euler Equations," *ICASE Report No.98 – 51*, NASA/CR-1998-208741.
- Darmofal, D.L., Moinier, P., and Giles, M.B., 2000, "Eigenmode Analysis of Boundary Conditions for the One-Dimensional Preconditioned Euler Equations," *Journal of Computational Physics*, Vol.160, pp. 369 – 384.
- Darrigol, O., 2002, "Between Hydrodynamics and Elasticity Theory: The First Five Births of the Navier-Stokes Equations," *Archive for History of Exact Sciences*, Vol.56, pp. 95 – 120.
- Durran, D.R., 1999, *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics*, Springer-Verlag, New York, Inc.

Engquist, B. and Majda, A., 1977, "Absorbing Boundary Conditions for the Numerical Simulation of Waves," *Mathematics of Computation*, Vol.31, No.139, pp. 629 – 651.

Erlebacher, G., Hussaini, M.Y., and Shu, C.W., 1996, "Interaction of a Shock With Longitudinal Vortex," *Journal of Fluid Mechanics*, Vol.337, pp. 129 – 153.

Fletcher, C.A.J. 1991, *Computational Techniques for Fluid Dynamics Volume II*, Springer-Verlag.

Fletcher, C.A.J., 1991, *Computational Techniques for Fluid Dynamics Volume I*, Springer-Verlag.

Flow Science: Excellence in Flow Modeling Software, Copyright © 2003, Flow Science Inc, <http://www.flow3d.com/Cfd-101/artvis.htm>.

Geiger, S., Matthäi, S.K., Roberts, S., and Zoppou, C., 2003, "Combining Finite Volume and Finite Element Methods to Simulate Fluid Flow in Geologic Media," *ANZIAM J.*, Vol.44 (E), pp. C180 – C201.

Giles, M.B., 1990, "Nonreflecting Boundary Conditions for Euler Equation Calculations," *AIAA Journal*, Vol.28, No.12, pp. 2050 – 2058.

Givoli, D., 1991, "Non-Reflecting Boundary Conditions," *Journal of Computational Physics*, Vol. 94, pp. 1 – 29.

Gourgoulhon, Eric, 2002, "Introduction to Spectral Methods," *Laboratoire de l'Univers et des Théories (LUTH), CNRS/Observatoire de Paris, 4th EU Network Meeting*, Palma de Mallorca.

Gustafsson, B. and Sundström, A., 1978, "Incompletely Parabolic Problems in Fluid Dynamics," *SIAM Journal of Applied Mathematics*, Vol.35, pp. 343 – 357.

Hanson, M.E. and Petschek, A.G., 1976, "A Boundary Condition for Sufficiently Reducing Boundary Reflection With a Lagrangian Mesh," *Journal of Computational Physics*, Vol.21, pp. 333 – 339.

Hayder, M.E., Hu, Q., and Hussaini, M.Y., 2001, "Towards Perfectly Absorbing Boundary Conditions for Euler Equations," National Aeronautics and Space Administration, contract No. NAS1-19480.

Hedstrom, G.W., 1979, "Nonreflecting Boundary Conditions for Nonlinear Hyperbolic Systems," *Journal of Computational Physics*, Vol.30, pp. 222 – 237.

Hesthaven, J.S., 1998, "On the Analysis and Construction of Perfectly Matched Layers for the Linearized Euler Equations," *Journal of Computational Physics*, Vol.142, pp. 129 – 147.

Hopp, T.H., 1996, "Representation of Axes for Geometric Fitting," U.S. Department of Commerce, NISTIR 5897, Manufacturing Systems Integration Division.

Hoffmann, K.A. and Chiang, S.T., 2000, *Computational Fluid Dynamics*, 4th Edition, Vol. 2, Engineering Education System.

Hoffmann, K.A. and Chiang, S.T., 2000, *Computational Fluid Dynamics*, 4th Edition, Vol. 3, Engineering Education System.

Hu, F.Q., 1996, "On Absorbing Boundary Conditions for Linearized Euler Equations by a Perfectly Matched Layer," *Journal of Computational Physics*, Vol.129, pp. 201 – 219.

Hu, F.Q., 2001, "A Stable, Perfectly Matched Layer for Linearized Euler Equations in Unsplit Physical Variables," *Journal of Computational Physics*, Vol.173, pp. 455 – 480.

- Israeli, M. and Orszag, S., 1981, "Approximation of Radiation Boundary Conditions," *Journal of Computational Physics*, Vol. 41, pp. 115 – 135.
- Jameson, A., Schmidt, W., and Turkel, E., 1981, "Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," *AIAA Paper 81-1259*.
- Kishoni, D. and Ta'asan, S., 1994, "Non-Reflecting Regions for Finite Difference Methods in Modeling of Elastic Wave Propagation in Plates," *Review of Progress in Quantitative Nondestructive Evaluation*, Vol.13, pp. 195 – 202.
- Klemp, J.B. and Lilly, D., 1978, "Numerical Simulation of Hydrostatic Mountain Waves," *J. Atmos. Sci.*, Vol. 35, pp. 78 – 107.
- Klien, A., 1981, "REVIEW: Effects of Inlet Conditions on Conical-Diffuser Performance," *Transactions of the ASME*, Vol.103, pp. 250 – 257.
- Kreiss, H.O., 1970, "Initial Boundary Value Problems for Hyperbolic Systems," *Communications on Pure and Applied Mathematics*, Vol.23, pp. 241 – 259.
- Lappas, T., Leonard, A., and Dimotakis, P.E., 1997, "Riemann Invariant Manifolds for the Multidimensional Euler Equations," *SIAM J. Sci. Comput.*, Vol.20, No.4, pp. 1481 – 1512.
- Lax, P.D., 1954, "Weak Solutions of Nonlinear Hyperbolic Equations and Their Numerical Computation," *Communications on Pure and Applied Mathematics*, Vol.7, pp. 159 – 193.
- Lin, F.B. and Sotiropoulos, F., 1997, "Assessment of Artificial Dissipation Models for Three-Dimensional Incompressible Flow Solutions," *Journal of Fluids Engineering*, Vol.119, pp. 331 – 340.
- Macaraeg, M.G. and Streett, C.L., 1986, "Improvements in Spectral Collocation Discretization Through a Multiple Domain Technique," *Applied Numerical Mathematics*, Vol.2, pp. 95 – 108.

Macaraeg, M.G., 1992, "Stability of Non-Similar Shear Layers," *Int. J. Num. Meth. Heat Fluid Flow*, Vol.2, pp. 79 – 86.

Makino, H., 1991, "An Adaptive Space Subdivision Method for Urban Area," Department of Information and System Engineering, Chuo University, makino@makino.ise.chuo-u.ac.jp.

Matsuno, T., 1966, "False Reflection Waves at the Boundary due to the use of Finite Differences," *J. Meteor. Soc. Japan*, Ser. 244, pp. 145 – 157.

Mazaheri, K. and Roe, P.L., 1997, "Numerical Wave Propagation and Steady state Solutions: Soft Wall and Outer Boundary Conditions," *AIAA Journal*, Vol. 35, No. 6, pp. 965 – 975.

Nicoud, F. and Poinso, T., 2000, "Boundary Conditions for Compressible Unsteady Flows," CERFACS-42, Toulouse, France, nicoud@cerfacs.fr and poinso@cerfacs.fr.

Nitta, T., "The Outflow Boundary Condition in Numerical Time Integration of Advection Equations," *J. Meteor. Soc. Japan*, Vol. 40, pp. 13 – 24.

NUMECA International, http://www.numeca.be/fturbo_home.html, 2003.

Oliger, J. and Sundström, A., 1978, "Theoretical and Practical Aspects of Some Initial Boundary Value Problems in Fluid Dynamics," *SIAM Journal on Applied Mathematics*, Vol.25, No.3, pp. 419 – 446.

Peterson, N.A., 2001, "Stability of Pressure Boundary Conditions for Stokes and Navier-Stokes Equations," *Journal of Computational Physics*, Vol.172, pp. 40 – 70.

Poinso, T.J. and Lele, S.K., 1992, "Boundary Conditions for Direct Simulations of Compressible Viscous Flows," *Journal of Computational Physics*, Vol. 101, pp. 104 – 129.

- Pulliam, T.H., 1993, "Time Accuracy and the Use of Implicit Methods," AIAA Paper 93-3360-CP, pp. 685 – 693.
- Reggio, M. and Camarero, R., 1983, "A Multigrid Scheme for Three-Dimensional Body-Fitted Coordinates in Turbomachine Applications," Transactions of the ASME, Vol.105, pp. 76 – 82.
- Reneau, L.R., Johnston, J.P., and Kline, S.J., 1996, "Performance and Design of Straight, Two-Dimensional Diffusers," Journal of Basic Engineering, Paper No. 66, pp. 141 – 150.
- Rhie, C.M. and Chow, W.L., 1983, "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation," AIAA Journal, Vol.11, pp. 1525 – 1533.
- Rizzi, A. and Eriksson, L., 1985, "Computation of Inviscid Incompressible Flow With Rotation," Journal of Fluid Mechanics, Vol.153, pp. 275 – 312.
- Roache, P.J., 1994, "Perspective: A Method for Uniform Reporting of Grid Refinement Studies," Journal of Fluids Engineering, Vol.116, pp. 405 – 413.
- Roe, P.L., 1986, ICASE Report No. 86-75, Nasa Langley, Hampton, NA, (unpublished).
- Rudy, D.H. and Strikwerda, J.C., 1980, "A Nonreflecting Outflow Boundary Condition for Subsonic Navier-Stokes Calculations," Journal of Computational Physics, Vol.36, pp. 55 – 70.
- Schneider, G.E. and Raw, M.J., 1987, "Control Volume Finite-Element Method for Heat Transfer and Fluid Flow Using Colocated Variables – 1. Computational Procedure," Numerical Heat Transfer, Vol.11, pp. 363 – 390.

Schneider, G.E. and Zedan, M., 1981, "A Modified Strongly Implicit Procedure for the Numerical Solution of Field Problems," Numerical Heat Transfer, Vol.4, pp. 1 – 19.

Schneider, G.E. and Zedan, M., 1982, "Control-Volume-Based Finite-Element Formulation of the Heat Conduction Equation," AIAA/ASME Joint Thermophysics, Fluids, Plasma and Heat Transfer Conference, Paper 82 – 0909.

Sesterhenn, J., 2001, "A Characteristic-type Formulation of the Navier-Stokes Equations for High Order Upwind Schemes," Computers and Fluids, Vol.30, pp. 37 – 67.

Sochacki, J., Kubichek, R., George, J., Fletcher, W.R., and Smithson, S., 1987, "Absorbing Boundary Conditions and Surface Waves," Geophysics, 52(1), pp. 60 – 71.

Sotiropoulos, F. and Abdallah, S., 1991, "The Discrete Continuity Equation in Primitive Variable Solution of Incompressible Flow," Journal of Computational Physics, Vol.95, pp. 212 – 227.

Steger, J.L. and Kutler, P., 1976, "Implicit Finite-Difference Procedures for the Computation of Vortex Wakes," AIAA Paper 76-385, San Diego, California.

Stone, H.L., 1968, "Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations," SIAM J. Numer. Anal., Vol.5, No.3, pp. 530 – 558.

Strang, G. and Fix, G.J., 1973, *An Analysis of the Finite Element Method*, Prentice-Hall, Inc.

Streett, C. and Macaraeg, M., 1986, "Preconditioning for First-Order Spectral Discretizations With Applications to Advection-Dominated Flows," Appl. Num. Math., Vol.2, pp. 123 – 139.

Swanson, R.C. and Radespiel, R., 1991, "Cell Centered and Cell Vertex Multigrid Schemes for the Navier-Stokes Equations," AIAA Journal, Vol.29, pp. 697 – 703.

Swanson, R.C. and Turkel, E., 1987, "Artificial Dissipation and Central Difference Schemes for the Euler and Navier-Stokes Equations," AIAA Paper 87-1107, pp. 55 – 62.

Ta'asan, S. and Nark, D.M., 1995, "An Absorbing Buffer Zone Technique of Acoustic Wave Propagation," AIAA Paper 95-0164.

Thompson, K., 1987, "Time Dependent Boundary Conditions for Hyperbolic Systems," Journal of Computational Physics, Vol.68, pp. 1 – 24.

Trefethen, L.N., *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations*, Cornell University,
<http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/pdetext.html>

Turkel, E. and Vatsa, V.N., "Effect of Artificial Viscosity on Three-Dimensional Flow Solutions," AIAA Journal, Vol.32, No.1, pp. 39 – 45.

Turkel, E., 1987, "Preconditioned Methods for Solving the Incompressible and Low Speed Compressible Equations," Journal of Computational Physics, Vol.72, pp. 277 – 298.

Turkel, E., 2002, "Robust Low Speed Preconditioning for Viscous High Lift Flows," AIAA Paper 2002-0962.

Vay, J., 2000, "A New Absorbing Layer Boundary Condition for the Wave Equation," Journal of Computational Physics, Vol.165, pp. 511 – 521.

Venkateswaran, S. and Merkle, C.L., 2000, "Evaluation of Artificial Dissipation Models and Their Relationship to the Accuracy of Euler and Navier-Stokes Computations," University of Tennessee Space Institute, pp. 427 – 432.

White, F.M., 1991, *Viscous Fluid Flow*, Second Edition, McGraw-Hill.

Whitham, G.B., 1974, *Linear and Nonlinear Waves*, John Wiley & Sons, Inc.

Yaras, M.I., 2002, "Numerical Simulations of Aircraft Wake-Vortex Dynamics in Nonuniform Windshear and Ground Proximity," AIAA Paper 2002-0940.

Yaras, M.I. and Grosvenor, A.D., 2002, "An Evaluation of Several Low-Re Turbulence Models Part 1 – Flat-Plate Boundary Layer and Axisymmetric Separating Flows," ICAS 2002 Congress, pp. 192.1 – 192.15.

Young, R. and MacPhedran, I., *Internet Finite Element Resources*,
http://www.engr.usask.ca/~macphed/finite/fe_resources/fe_resources.html.

Zauderer, E., 1989, *Partial Differential Equations of Applied Mathematics*, Second Edition, John Wiley & Sons, Inc.

APPENDIX A – THE DERIVATION OF THE GOVERNING EQUATIONS OF FLUID DYNAMICS

In order to incorporate an element of completeness, a brief derivation of the governing equations of fluid motion is included here. Similar approaches can be found in most introductory fluid mechanics textbooks.

A.1 CONSERVATION OF MASS

To begin, let us consider an arbitrary region R of fixed volume $V_{c.v.}$ within a flow, having boundary $A_{c.v.}$ and outward normal $\mathbf{n}$, as depicted in the figure below.

Figure A.1.1 – Arbitrary flow region R

Let us first assume that the density, ρ , is constant.

$$\Rightarrow \text{mass inside } R \text{ is constant}$$

$$\Rightarrow \text{Net mass flux} = \frac{\text{mass}}{\text{time}} \text{ across } S = 0 \quad (\text{A.1.1})$$

$$\Rightarrow \int_{A_{c.v.}} (\mathbf{u} \cdot \mathbf{n}) dA_{c.v.} = 0 \quad (\text{A.1.2})$$

$$\Rightarrow \int_R (\nabla \cdot \mathbf{u}) dV_{c.v.} = 0 \quad (\text{Divergence Theorem}) \quad (\text{A.1.3})$$

$$\Rightarrow \nabla \cdot \mathbf{u} = 0 \quad (\text{Incompressibility Condition}) \quad (\text{A.1.4})$$

Conversely, suppose that ρ is not constant, that is, that the fluid density can vary in time and space. In the Eulerian frame of reference, the mass inside $A_{c.v.}$ at any given time is given by:

$$A_{c.v.} = \int_R \rho dV_{c.v.} \quad (A.1.5)$$

The fact that the increase of mass inside $A_{c.v.}$ must be equal to the net influx into R gives the following mathematical expression:

$$\frac{\partial}{\partial t} \left(\int_R \rho dV_{c.v.} \right) = - \int_{A_{c.v.}} \rho (\mathbf{u} \cdot \mathbf{n}) dA_{c.v.} \quad (A.1.6)$$

$$\Rightarrow \int_R \left(\frac{\partial \rho}{\partial t} \right) dV_{c.v.} = - \int_R \nabla \cdot (\rho \mathbf{u}) dV_{c.v.} \quad (A.1.7)$$

$$\Rightarrow \int_R \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right) dV_{c.v.} = 0 \quad (A.1.8)$$

$$\Rightarrow \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad (A.1.9)$$

Eqn. (A.1.9) is the resulting expression for the conservation of mass. As an alternative approach, let us now consider the Lagrangian frame of reference, which implies that the density is a function of the three spatial coordinates in time:

$$\rho = \rho(x(t), y(t), z(t), t)$$

Proceeding as above, one can arrive at the following equation for the conservation of mass in the Lagrangian frame of reference:

$$\frac{1}{\rho} \frac{D\rho}{Dt} + \nabla \cdot \mathbf{u} = 0 \quad (A.1.10)$$

where, $\frac{D\rho}{Dt}$ is the material derivative given by:

$$\frac{D\rho}{Dt} \equiv \frac{\partial \rho}{\partial t} + (\mathbf{u} \cdot \nabla) \rho \quad (\text{A.1.11})$$

Finally, the two derivations are mathematically equivalent, that is, conservation of mass in the Eulerian frame of reference is equal to that in the Lagrangian frame of reference. This follows easily since:

$$\begin{aligned} \frac{1}{\rho} \frac{D\rho}{Dt} + \nabla \cdot \mathbf{u} &= 0 \\ \Rightarrow \frac{\partial \rho}{\partial t} + (\mathbf{u} \cdot \nabla) \rho + \rho (\nabla \cdot \mathbf{u}) &= 0 \\ \Rightarrow \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \end{aligned}$$

A.2 CONSERVATION OF MOMENTUM

Continuing in the Lagrangian frame of reference and applying Newton's Second Law to the fluid region R , yields:

$$\int_R \frac{D\mathbf{u}}{Dt} \rho dV_{c.v.} = \int_R \rho \mathbf{g} dV_{c.v.} - \int_R \nabla p dV_{c.v.} + \int_R \mu \nabla^2 \mathbf{u} dV_{c.v.} \quad (\text{A.2.1})$$

$$\Rightarrow \int_R \left(\frac{D\mathbf{u}}{Dt} \rho - \rho \mathbf{g} + \nabla p - \mu \nabla^2 \mathbf{u} \right) dV = 0 \quad (\text{A.2.2})$$

$$\Rightarrow \frac{D\mathbf{u}}{Dt} \rho = \rho \mathbf{g} - \nabla p + \mu \nabla^2 \mathbf{u} \quad (\text{A.2.3})$$

$$\Rightarrow \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \mathbf{g} - \frac{\nabla p}{\rho} + \mu \nabla^2 \cdot \mathbf{u} \quad (\text{A.2.4})$$

Eqn. (A.2.4) is the familiar expression for the conservation of momentum. In using Newton's Second Law above, the forces accounted for are gravity, viscosity, and the

pressure due to neighbouring fluid particles. The result is a system of four equations (conservation of mass and three scalar equations representing conservation of momentum) in five unknowns (u, v, w, ρ , and p). To close the system, the most common method of resolution is to introduce an equation of state relating two of the above variables as the necessary fifth equation. For example, when dealing with a perfect gas, one can use the following equation of state:

$$p = \rho RT \quad (\text{A.2.5})$$

This introduces temperature as yet another unknown, hence an additional equation is needed to close the system of equations.

A.3 CONSERVATION OF ENERGY

To obtain this needed equation, we consider the energy balance over our control volume $V_{c.v.}$, according to the First Law of Thermodynamics. The fluid has internal energy due to molecular motion, which shall be denoted by e . Moreover, there is also heat flow, q , across the boundary. The total energy density per unit volume consists of the kinetic energy, $\frac{1}{2}\rho u^2$, at the macroscopic level, plus the internal energy, ρe , of the molecular motion. Our integral energy balance is:

$$\begin{aligned} \frac{\partial}{\partial t} \int_{V_{c.v.}} \left(\frac{1}{2} \rho u^2 + \rho e \right) dV_{c.v.} + \int_{A_{c.v.}} \{ (\rho u^2 + \rho e)(\mathbf{u} \cdot \mathbf{n}) - (\mathbf{u} \cdot \mathbf{p}) + (\mathbf{q} \cdot \mathbf{n}) \} dA_{c.v.} \\ = \int_{V_{c.v.}} \rho \mathbf{F}_B \cdot \mathbf{u} dV_{c.v.} \end{aligned} \quad (\text{A.3.1})$$

The first term in the surface integral is again the contribution from the bodily transport across the boundary, the second term is the rate of work being done by the stresses $\mathbf{p}$ at the boundary, and the third term is the loss or gain of heat by conduction across the boundary. The right hand side represents the rate of work being done by the body forces $\mathbf{F}_B$. Applying Gauss' Divergence Theorem to Eqn. (A.3.1) yields:

$$\begin{aligned} \int_{V_{c.v.}} \frac{\partial}{\partial t} \left(\frac{1}{2} \rho \mathbf{u}^2 + \rho \mathbf{e} \right) dV_{c.v.} + \int_{V_{c.v.}} \left(\nabla \cdot [(\rho \mathbf{u}^2 + \rho \mathbf{e})\mathbf{u}] - (\mathbf{u} \cdot \mathbf{p}) + (\nabla \cdot \mathbf{q}) \right) dV_{c.v.} \\ = \int_{V_{c.v.}} \rho \mathbf{F}_B \cdot \mathbf{u} dV_{c.v.} \end{aligned} \quad (\text{A.3.2})$$

Reasoning as before, we are left with a version of what is commonly known as the energy equation:

$$\frac{\partial}{\partial t} \left(\frac{1}{2} \rho \mathbf{u}^2 + \rho \mathbf{e} \right) + \nabla \cdot [(\rho \mathbf{u}^2 + \rho \mathbf{e})\mathbf{u}] - (\mathbf{u} \cdot \mathbf{p}) + (\nabla \cdot \mathbf{q}) = \rho \mathbf{F}_B \cdot \mathbf{u} \quad (\text{A.3.3})$$

Using $\mathbf{e} = e_i - \frac{1}{2} \mathbf{u}^2$, the energy equation may be expressed in a conservative form as:

$$\begin{aligned} \frac{\partial}{\partial t} (\rho e_i) + \frac{\partial}{\partial x} (\rho u e_i + p u) + \frac{\partial}{\partial y} (\rho v e_i + p v) + \frac{\partial}{\partial z} (\rho w e_i + p w) = \frac{\partial}{\partial x} (u \tau_{xx} + v \tau_{xy} + w \tau_{xz} - q_x) \\ + \frac{\partial}{\partial y} (u \tau_{yx} + v \tau_{yy} + w \tau_{yz} - q_y) + \frac{\partial}{\partial z} (u \tau_{zx} + v \tau_{zy} + w \tau_{zz} - q_z) \end{aligned} \quad (\text{A.3.4})$$

Here e_i is the total energy, q_{x-z} represent components of the heat flux $\mathbf{q}$, and for a Newtonian fluid the viscous stresses are given by:

$$\tau_{xx} = 2\mu \frac{\partial u}{\partial x} + \lambda \nabla \cdot \mathbf{u} \quad \tau_{xy} = \tau_{yx} = \mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \quad (\text{A.3.5})$$

$$\tau_{yy} = 2\mu \frac{\partial v}{\partial y} + \lambda \nabla \cdot \mathbf{u} \quad \tau_{xz} = \tau_{zx} = \mu \left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) \quad (\text{A.3.6})$$

$$\tau_{zz} = 2\mu \frac{\partial w}{\partial z} + \lambda \nabla \cdot \mathbf{u} \quad \tau_{yz} = \tau_{zy} = \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \quad (\text{A.3.7})$$

The constant $k = \lambda + \frac{2}{3}\mu$ is referred to as the bulk viscosity of the fluid. This

parameter is often assumed negligible, implying that $\lambda = -\frac{2}{3}\mu$.

Thus, the inclusion of the viscous term in the momentum conservation equations and the derivation of the energy equation result in a system referred to as the Navier-Stokes equations. In vector form these are given as:

Compressible: (A.3.8)

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{\nabla p}{\rho} + \mathbf{g} + \mu \nabla^2 \cdot \mathbf{u}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

$$\frac{\partial}{\partial t} \left(\frac{1}{2} \rho \mathbf{u}^2 + \rho e \right) + \nabla \cdot [(\rho \mathbf{u}^2 + \rho e) \mathbf{u}] - (\mathbf{u} \cdot \mathbf{p}) + (\nabla \cdot \mathbf{q}) = \rho \mathbf{F}_B \cdot \mathbf{u}$$

Incompressible: (A.3.9)

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{\nabla p}{\rho} + \mathbf{g} + \mu \nabla^2 \cdot \mathbf{u}$$

$$\nabla \cdot \mathbf{u} = 0$$

APPENDIX B – COMMON CFD TECHNIQUES AND THE *FS* ALGORITHM

Implementation of the governing equations of fluid dynamics, with appropriate initial and boundary conditions, into the framework of a numerical algorithm is indeed a daunting task. As mentioned within the main portion of the text, often researchers, in designing an appropriate algorithm, have a multitude of options in regards to available techniques. In Appendix B, not only do we present the methods employed within the *FS* code, but attempt to give a general overview of various techniques so as to encompass as much of the CFD picture as possible. Thus, the appendix given here might read as an introductory textbook to the field of computational fluid dynamics, and so for that reason, it may read in its entirety or used merely as a tool to address a particular topic.

The equations of fluid motion of conservation of mass, momentum, and energy, can be written in a variety forms. In general, we say that an equation can be placed in a conservative form provided it can be written as:

$$\vec{U}_t + \nabla \cdot \vec{F} = P \quad (\text{B.0.1})$$

where $\vec{U}$ is the conserved quantity, $\vec{F} = \vec{F}(\vec{U})$ is an associated flux, and P corresponds to any external sources or sinks. $\vec{U}$ and P may be scalars, vectors, or tensors, and $\vec{F}$ is a quantity of higher order than $\vec{U}$ and P .

When the equations of fluid motion are written in a conservative form, it becomes easier to develop numerical schemes that are also in conservative form. The Navier-Stokes equations can be written in the following compact form:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} = \frac{\partial \mathbf{E}_v}{\partial x} + \frac{\partial \mathbf{F}_v}{\partial y} + \frac{\partial \mathbf{G}_v}{\partial z} + \mathbf{H} \quad (\text{B.0.2})$$

where, $\mathbf{Q}, \mathbf{E}, \mathbf{F}, \mathbf{G}, \mathbf{E}_v, \mathbf{F}_v$, and $\mathbf{G}_v$ are given in Eqns. (1.2.9) – (1.2.12), and the viscous stresses τ_{ij} are defined in Eqns. (1.2.4) – (1.2.6). Here $\mathbf{H}$ is the source term. In addition, Fourier's Law of Heat Conduction gives us the form of the heat fluxes, q_{x-z} . For example, q_x is given by:

$$q_x = -k \frac{\partial T}{\partial x} \quad (\text{B.0.3})$$

Here k is referred to as the thermal conductivity and T is the temperature. The expressions for q_y and q_z are analogous.

Although these equations are presented in their dimensional form, *FS* uses the non-dimensional form of the governing equations in order to minimize round-off error during computation.

Amongst the general classes of discretization schemes used for solving PDEs numerically, three of the most common are finite-differencing, finite-element, and finite-volume methods. Each of these schemes receives further examination below.

B.1 FINITE-DIFFERENCE METHOD

Historically, the application of finite-differencing methods to the governing equations of fluid mechanics has been the most common among the aforementioned discretization methods. This is primarily due to the simplicity of the schemes and associated structured grid system. In finite-differencing, the discrete representations of the partial derivatives are obtained from Taylor series expansions. The finite-difference equation that approximates the PDE is algebraic in nature, and one such equation is written for each grid point within the domain. In an alternative approach, derivatives are approximated by representing the function as a polynomial. The coefficients of the polynomial are computed by substitution of data (dependent variable) from a series of (usually) equally “spaced” values of the independent variable. Approximate values of the derivatives are then computed from the polynomial.

Although classical finite-difference methods have virtually disappeared from practical use in solid and structural mechanics, these discretization techniques are still important in fluid mechanics applications that allow the domain to be discretized into a smooth structured grid.

On the other hand, when the integral form of the equations is utilized (as opposed to the differential form), the domain of solution is divided into small volumes (or areas in the two-dimensional case). Such integral methods include finite-volume and finite-element schemes. For linear and certain non-linear problems, finite-element methods seem to be the dominant choice, however, for the highly non-linear problems of fluid mechanics, finite-volume techniques stand out as the first choice.

B.2 FINITE-ELEMENT METHOD

As Huyakorn *et al.* (1983) explains, the finite-element method is a numerical technique for solving the differential equations by means of “piecewise approximations”. Unlike the finite-difference method, which carries out the computation on an array of grid points, the finite-element method treats the region as being made up of many small, interconnected subregions called “finite elements”. These elements generally take simple shapes such as triangles and quadrilaterals in 2-D space, and are assembled in such a way so that they represent the solution domain of interest.

In order to introduce the notion of a “finite element”, many educators have used the ancient problem of finding the perimeter, P , of a circle of diameter d . Since $P = \pi d$, this is equivalent to obtaining a numerical value for π , and is thus sometimes referred to as the “ π problem”. For the circle shown in Fig. B.2.1 (a) we inscribe a regular polygon of n sides, where $n = 8$, as shown in Fig. B.2.1 (b). To make the parallel with finite-element methods, the sides of the polygon are referred to as *elements* and the vertices as *nodal points* or *nodes*. Extracting one of the elements, for example joining nodes 4 and 5 as shown in Fig. B.2.1 (c), we have one of the *generic elements*, whose general structure is shown in Fig. B.2.1 (d). The length of this element is

$P_{ij} = 2r \sin\left(\frac{\pi}{n}\right)$. Since all elements have the same length, the polygon perimeter is

$P_n = n P_{ij}$, hence the approximation to π is $\pi_n = P_n / d = n \sin\left(\frac{\pi}{n}\right)$.

Figure B.2.1 – Introduction to finite element methods in the context of the “ π problem”

(a) continuum object, (b) a discrete approximation, (c) disconnected element, (d) generic element, reproduced from Young and MacPhedran (2002)

Values of π_n obtained for $n = 1, 2, 4, \dots, 256$ are listed in the second column of Table

B.2.1:

Table B.2.1 – An approximation for π using $\pi_n = n \sin(\frac{\pi}{n})$, reproduced from Roger and MacPhedran (2002)

n	$\pi_n = n \sin(\frac{\pi}{n})$	Extrapolated by Wynn- ε	Exact π to 16 places
1	0.0000000000000000		
2	2.0000000000000000		
4	2.828427124746190	3.414213562373096	
8	3.061467458920718		
16	3.121445152258052	3.141418327933211	
32	3.136548490545939		
64	3.140331156954753	3.141592658918053	
128	3.141277250932773		
256	3.141513801144301	3.141592653589786	3.141592653589793

Clearly, convergence to the true value of π , using only the above approximation is noticeably slow. Results obtained through the use of Wynn’s extrapolation ε algorithm, a widely used method for speeding up the convergence of many sequences, are given column three. The fourth column gives π , exact to 16 decimal places.

Although this example is trivial in nature, it demonstrates some of the key ideas behind the finite-element method. The circle, which can be interpreted as the physical domain of interest, is replaced by polygons, and hence, is *discretely approximated* by elements (the sides of the triangles). Moreover, disconnecting the nodes, a process referred to as *disassembly*, can separate elements. Once the elements are disassembled, a generic element can be defined, independently of the original domain. In general, a matrix expression is developed to relate the nodal variables of each element, which appropriately, is often called an *element matrix*. For a discrete problem, the element matrix relation can usually be established by direct physical reasoning. Alternatively, for problems of a continuous nature, a more general mathematical formulation making use of either a variational or weighted residual method is required. These element matrices are then combined to form a set of algebraic equations that describe the entire global system. The assembly is done in such a way that compatibility conditions are satisfied at each node shared by different elements. Prescribed boundary conditions are incorporated into the global matrix equation and the resulting set of simultaneous algebraic equations can be solved using any one of a variety of algorithms. This method is carried out for each unknown quantity that we are interested in solving for.

B.3 FINITE-VOLUME METHOD

As a technique for solving partial differential equations, the finite-volume method calculates the values of the conserved variables averaged across a control volume. To explain how the technique is carried out, consider Eqn. (B.0.2). If we denote the right hand side of this equation by $\mathbf{H}$, the source term, we can integrate over our control volume $V_{c.v.}$:

$$\iiint_{V_{c.v.}} \left[\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} - \left(\frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y} + \frac{\partial G_v}{\partial z} \right) \right] dV_{c.v.} = \iiint_{V_{c.v.}} H dV_{c.v.} \quad (B.3.1)$$

In Eqn. (B.3.1) we apply Gauss' Divergence Theorem to the spatial derivatives to obtain:

$$\iiint_{V_{c.v.}} \frac{\partial Q}{\partial t} dV_{c.v.} + \iint_{A_{c.v.}} ((E - E_v) \cdot \mathbf{n}_x + (F - F_v) \cdot \mathbf{n}_y + (G - G_v) \cdot \mathbf{n}_z) dA_{c.v.} = \iiint_{V_{c.v.}} H dV_{c.v.} \quad (B.3.2)$$

Assuming we have a spatially uniform distribution of Q and H within the control volume, and piecewise uniform distribution of fluxes over the surface faces of this control volume, Eqn. (B.3.2) yields:

$$\left(\frac{dQ_{c.v.}}{dt} \right) V_{c.v.} + \sum_{c.v. \text{ faces}} ((E - E_v) A_x + (F - F_v) A_y + (G - G_v) A_z) = H_{c.v.} V_{c.v.} \quad (B.3.3)$$

Here the summation is taken over the control volume faces. What remains in Eqn. (B.3.3) is a set of ordinary differential equations in time, and thus can be integrated by one of the many available methods.

B.4 SPATIAL GRIDS

There are two main types of spatial grids used in numerical algorithms, structured and unstructured grids. Classical examples of structured and unstructured grids are given in Fig. B.4.1:

Figure B.4.1 – Example of (a) a structured grid and (b) an unstructured grid

B.4.1 STRUCTURED GRIDS

Structured grids have a regular topology where the neighbourhood relationship between all nodes can be written in a two- or three-dimensional array. This geometric property makes the computer storage of flow variables relatively easy. Moreover, the multigrid method to be discussed in Section B.4.8 works rather efficiently on structured grids. Unfortunately, Fig. B.4.1 (a) provides a very simple representation of this type of grid.

Generating a high-quality structured grid for more intricate computational domains is often quite a challenge. The orthogonality of grid lines is crucial for the convergence of the corresponding numerical algorithm. For example, in turbomachinery problems, the resolution of a flow near the leading and trailing edges of a blade often requires additional node clustering, which in turn translates into increased computational time. Fig. B.4.1.1 shows a portion of a grid surrounding the nozzle of NASA/Langley space shuttle.

Figure B.4.1.1 – Structured grid surrounding the front portion of a NASA/Langley space shuttle

B.4.2 UNSTRUCTURED GRIDS

Unstructured grids are a collection of vertices, edges, faces, and cells whose connectivity information must be specified. Most grids of this type are well equipped to handle complex geometries; however, at times grid quality becomes a concern.

Unstructured grids can focus on particular regions of a flow more easily than structured grids. This quality allows for a greater concentration of cells in desired areas of the computational domain. In Fig. B.4.2.1 we once again have the NASA/Langley space shuttle nozzle, however, this time in the context of an unstructured grid. Since there is no need to preserve uniformity in the grid, clustering near the surface of the leading edge is not as difficult. This lack of “order” in the unstructured mesh translates into a need for a connectivity matrix, which outlines node connectivity within a given computational domain.

Figure B.4.2.1 – Unstructured grid surrounding the front portion of a NASA/Langley space shuttle

B.4.3 MULTI BLOCK STRUCTURED GRIDS

In order to address the need for greater resolution in certain regions of the computational domain, some numerical algorithms can accommodate multi block or zonal structured grids. In such a case, several simple blocks are combined to form more complex shapes. Fig. B.4.3.1 shows a few examples of how this multi block approach might be carried out. Note that the Chimera approach allows for overlapping blocks, yet once again, with an associated increase in computational time. Each structured grid component can be generated independently provided additional information specifies their interaction.

Figure B.4.3.1 – Different grid distribution approaches, reproduced from Lin *et al.* (1991)

The solution of one grid is communicated with other grids through these overlapped regions. Unfortunately, the numerical implementation of the zonal grid approach can be cumbersome if one is to develop a general-purpose algorithm for any arbitrary configuration.

In terms of numerical integration, finite-differencing schemes are more readily implemented with a structured grid, and thus possess an inherent weakness for highly complicated domains, whereas finite-volume and finite-element schemes may be used with either structured or unstructured grids. For these latter two methods, the independent variables are integrated directly on the physical domain. As a result, the

governing equations can be solved so long as the domain can be successfully discretized into elements, whether structured or unstructured. Clearly finite-element and finite-volume schemes provide greater flexibility in that a wide range of choices are available for the selection of discrete elements. However, it should be noted that if the domain can be discretized into a smooth structured grid, the finite-differencing method would be a better choice due to its efficiency over that of the finite-volume or finite-element methods. *FS* uses a finite-volume scheme that can accommodate single or multi block structured grids.

B.4.4 CONTROL VOLUME

Swanson and Radespiel (1991) describe two basic finite-volume techniques for the discretization of the compressible Navier-Stokes equations represented in an integral form: a cell-centered scheme and a vertex-centered scheme. The control volume is the basic unit over which computations are carried out and flow variables are stored. Their shape, number of nodes used, and location of variable storage can vary widely from problem to problem. However, in general their definition depends on whether a cell-centered or a vertex-centered scheme is employed.

B.4.4.1 CELL-CENTERED CONTROL VOLUME SCHEME

In the cell-centered scheme, the flow variables are located at the “center” of a grid cell (represented by an x in Fig. B.4.4.1.1), which as we shall see shortly, need not be the geometrical center. One can deduce from Fig. B.4.4.1.1 that the same combination of grid nodes define both a grid cell and a control volume (i.e. $A = (i+1, j)$, $B = (i+1, j+1)$, $C = (i, j+1)$, and $D = (i, j)$). For example, the grid cell

composed of the four points A, B, C, and D represents a computational control volume, where the information is stored at its center (geometrical center in this case).

Figure B.4.4.1.1 – Cell-centered control volume (two-dimensional)

An averaging process evaluates the flux vectors at the midpoint of a cell. In order to see this, consider Fig. B.4.4.1.2, displaying a single grid cell with a local (s,t) coordinate system used to label the positioning of the sides. Suppose that Θ is some flow quantity (i.e. pressure, velocity, temperature, etc.). Then this quantity and the global coordinates (x, y) are expressed in the form:

$$\Theta = \sum_{i=1}^4 S_i \Theta_i \quad (\text{B.4.4.1.1})$$

$$x = \sum_{i=1}^4 S_i x_i \quad (\text{B.4.4.1.2})$$

$$y = \sum_{i=1}^4 S_i y_i \quad (\text{B.4.4.1.3})$$

where the shape functions S_i are given by:

$$S_1 = \frac{1}{4}(1-s)(1-t) \quad (\text{B.4.4.1.4})$$

$$S_2 = \frac{1}{4}(1+s)(1-t) \quad (\text{B.4.4.1.5})$$

$$S_3 = \frac{1}{4}(1+s)(1+t) \quad (\text{B.4.4.1.6})$$

$$S_4 = \frac{1}{4}(1-s)(1+t) \quad (\text{B.4.4.1.7})$$

Figure B.4.4.1.2 – A typical single grid cell

Having these definitions allows the x and y derivatives of a flow variable to be computed as follows:

$$\frac{\partial \Theta}{\partial x} = \sum_{i=1}^4 \frac{\partial S_i}{\partial x} \Theta_i \quad (\text{B.4.4.1.8})$$

$$\frac{\partial \Theta}{\partial y} = \sum_{i=1}^4 \frac{\partial S_i}{\partial y} \Theta_i \quad (\text{B.4.4.1.9})$$

Finally, the x and y derivatives of the shape functions are given by the following relationship:

$$\begin{bmatrix} \frac{\partial S_i}{\partial x} \\ \frac{\partial S_i}{\partial y} \end{bmatrix} = \frac{1}{\text{Det}(H)} \begin{bmatrix} \frac{\partial y}{\partial t} & -\frac{\partial y}{\partial s} \\ -\frac{\partial x}{\partial t} & \frac{\partial x}{\partial s} \end{bmatrix} \begin{bmatrix} \frac{\partial S_i}{\partial s} \\ \frac{\partial S_i}{\partial t} \end{bmatrix} \quad (\text{B.4.4.1.10})$$

where,

$$\text{Det}(H) = \frac{\partial x}{\partial s} \frac{\partial y}{\partial t} - \frac{\partial y}{\partial s} \frac{\partial x}{\partial t} \quad (\text{B.4.4.1.11})$$

The contributions of u_x and u_y to the viscous fluxes across cell face BC are determined using:

$$\iint_{\Omega'} u_x d\Omega' = \int_{\partial\Omega'} u dy \quad (\text{B.4.4.1.12})$$

$$\iint_{\Omega'} u_y d\Omega' = - \int_{\partial\Omega'} u dx \quad (\text{B.4.4.1.13})$$

where Ω' is the auxiliary cell A' B' C' D', and $\partial\Omega'$ is its boundary. The values of u at B and C required for the above integrals are obtained by a simple average of the values of u at the four surrounding cells. v_x and v_y are computed in the same fashion.

B.4.4.2 VERTEX-CENTERED CONTROL VOLUME SCHEME

In the case of a vertex-centered scheme, the flow variables are located at the vertices of the grid cells, denoted by $I_{gc} = (i+1, j)$ for cell ABCD in Fig. B.4.4.2.1.

Figure B.4.4.2.1 – Vertex-centered control volume (two-dimensional)

As with cell-centered schemes, a grid cell is defined by the nodes of the computational grid. However, in this case a control volume, again denoted by ABCD, differs from a grid cell. A control volume is composed of four points whose locations are defined by the centers of four adjacent grid cells, as shown in Fig. B.4.4.2.1.

The dependent variables at the midpoints of the faces of adjacent control volume cells are obtained by averaging the values at adjacent vertices. From this, the resultant convective inflow of mass, momentum, and energy associated with point I_{gc} is computed by summing the contributions of the adjacent cells. The viscous fluxes for each control volume face are obtained in a similar fashion as the cell-centered scheme described in the previous section.

Some of these aspects are discussed within Chapter 5 of the main portion of this work, where we describe the implementation of the buffer zone technique into full Navier-Stokes solvers.

B.4.4.3 FS CONTROL VOLUME

The shape of a control volume depends on the capabilities of the solver in which it is implemented. Structured-grid codes, such as *FS*, use quadrilaterals in two dimensions and hexahedra for three-dimensional flows. Unstructured grid solvers often use triangles (2D) or tetrahedra (3D). *FS* employs a vertex-centered control volume with the advantage that, as outlined in Swanson *et al.* (1991), special interpolation methods at computational-domain boundaries are not required. Furthermore, the option of selecting between one of two computational volumes is provided. More specifically, one has the choice of either a 7-node (5 nodes in two dimensions) or a 27-node (9 nodes in two dimensions) control volume, which are used to determine the fluxes at the faces. The larger computational molecule allows for the non-uniform description of fluxes over the faces of the control volume. Intuitively this would imply that fewer grid nodes are required for similar levels of convergence as compared to the smaller control volume. However, a greater computational effort is needed when utilizing the larger molecule. Appendix D presents some numerical simulations comparing the two computational stencils.

B.4.5 NUMERICAL DISCRETIZATION

In principle, the Navier-Stokes equations discretized in space, as in Eqn. (B.3.3), are sufficient to produce unique solutions given complete specification of initial and boundary conditions. However, since the continuum that exists in space is only represented in a discrete manner, the damping that exists in real life through the viscosity of fluids is insufficient to reduce the spurious oscillations that may develop in numerical solutions. Many existing algorithms rely on the artificial diffusion that is

produced by virtue of the specific discretization used for each derivative in the governing equations, particularly the spatial convection term. This, however, does not allow for any control over the extent of artificial diffusion introduced into the solution. An alternative approach is to minimize the artificial diffusion inherent in the discretized derivatives, and then introduce the desired amount of artificial diffusion through terms explicitly added to the governing equations being solved. The extent of this diffusion would then be chosen to eliminate decoupling of the pressure field over odd- and even-numbered nodes on the spatial grid, and to damp oscillations in regions of high spatial gradients in flow variables, such as shock waves in high-speed flows.

The governing equations discretized in space would thus become:

$$\left(\frac{dQ_{c.v.}}{dt} \right) V_{c.v.} + \sum_{c.v. \text{ faces}} (\mathbf{E}A_x + \mathbf{F}A_y + \mathbf{G}A_z) = \mathbf{H}_{c.v.} V_{c.v.} + D_{a_{c.v.}} \quad (\text{B.4.5.1})$$

where $D_{a_{c.v.}}$ is the artificial diffusion term consisting of second- and fourth-order differences of the solution vector in all three grid directions:

$$D_{a_{c.v.}} = \left(d_{i+\frac{1}{2},j,k} - d_{i-\frac{1}{2},j,k} \right) + \left(d_{i,j+\frac{1}{2},k} - d_{i,j-\frac{1}{2},k} \right) + \left(d_{i,j,k+\frac{1}{2}} - d_{i,j,k-\frac{1}{2}} \right) \quad (\text{B.4.5.2})$$

The terms in each bracket on the right-hand side denote artificial diffusion along one particular grid direction. For example:

$$d_{i+\frac{1}{2},j,k} = \lambda_{i+\frac{1}{2},j,k} \varepsilon_{2,i+\frac{1}{2},j,k} (Q_{i+1,j,k} - Q_{i,j,k}) \quad (\text{B.4.5.3})$$

$$- \varepsilon_{4,i+\frac{1}{2},j,k} (Q_{i+2,j,k} - 3Q_{i+1,j,k} + 3Q_{i,j,k} - Q_{i-1,j,k})$$

Here $\lambda_{i+\frac{1}{2},j,k}$ is the spectral radius of the inviscid flux Jacobian matrix and is a common scaling parameter for the artificial dissipation term. This spectral radius is obtained as follows:

$$\lambda_{i+\frac{1}{2},j,k} = \left(\left| \mathbf{V} A_{c.v.} \right| + \left(\left(\frac{c}{a_1} \right)^2 + \mathbf{V}^2 \right)^{\frac{1}{2}} \left| A_{c.v.} \right| \right)_{i+\frac{1}{2},j,k} \quad (\text{B.4.5.4})$$

$A_{c.v.}$ is the control volume surface taken in the appropriate direction and $\frac{c}{a_1}$ is a

preconditioning parameter to be described shortly.

The second- and fourth-order nature of the finite differencing is clearly evident. The coefficients ε_2 and ε_4 are given by:

$$\varepsilon_{2,i+\frac{1}{2},j,k} = \kappa_2 \max(v_{i+1,j,k}; v_{i,j,k})$$

$$\varepsilon_{4,i+\frac{1}{2},j,k} = \max \left(0; \left(\kappa_4 - \varepsilon_{2,i+\frac{1}{2},j,k} \right) \right) \quad (\text{B.4.5.5})$$

where,

$$v_{i,j,k} = \frac{|p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}|}{p_{i+1,j,k} + 2p_{i,j,k} + p_{i-1,j,k}} \quad (\text{B.4.5.6})$$

κ_2 and κ_4 are scaling coefficients chosen so that the solution is free of decoupling oscillations in space without having the artificial diffusion mask the physical process. The second- and fourth-order differences are first- and third-order accurate on a uniform mesh. As per the pressure switch given in Eqn. (B.4.5.6), the second-order difference term is activated only in regions of high-pressure gradients such as shocks in compressible flows. In these regions the solution accuracy therefore reduces to first-order in space. Fortunately, this usually accounts for a very small portion of the domain, hence minimally affecting solution convergence. Elsewhere, only the fourth-order difference term remains active, in which case the formal spatial accuracy of the solution is driven by the order of accuracy of the convective and diffusive spatial derivatives, which are chosen to be formally second-order accurate; that is, one order less accurate than the artificial diffusion term.

B.4.5.1 PSEUDO-TIME STEPPING AND PRECONDITIONING (ARTIFICIAL COMPRESSIBILITY)

Section 5.2 of the main portion of this work is an excellent compliment to the next two sections presented here regarding artificial compressibility and pseudo-time stepping. It is advised to those wishing to have a greater understanding of the sections to follow, to consult Section 5.2 first. Discretization of the temporal derivative in Eqn. (B.4.5.1) is based on a three-point backward differencing, yielding second order accuracy in time (with the exception of the of the first time increment which is based on on a two-point scheme). At each real-time increment, the discretized governing equations are solved iteratively using pseudo-time stepping for relaxation:

$$\left(\frac{d\mathbf{Q}_{c,v}}{dt_p}\right)V_{c,v} + \left(\frac{d\mathbf{Q}_{c,v}}{dt}\right)V_{c,v} + \sum_{c.v. \text{ faces}} (\mathbf{E}A_x + \mathbf{F}A_y + \mathbf{G}A_z) - \mathbf{H}_{c,v} V_{c,v} - D_{a_{c,v}} = 0 \quad (\text{B.4.5.1.1})$$

For steady-flow simulations, the temporal derivative $\frac{d\mathbf{Q}_{c,v}}{dt}$ is not included in computations. The pseudo-time derivative replaces the vector of conservative variables $\mathbf{Q}$, and is preconditioned to extend the range of applicability of the algorithm to low Mach-number flows. The exact nature of this preconditioning is described in the following subsection.

B.4.5.1.1 ARTIFICIAL COMPRESSIBILITY

The standard hyperbolic methods used to solve the discretized governing equations described in the previous sections are not effective in the limit of incompressible flow. At this limit, the sound waves dominate the system and the governing equations become poorly conditioned for numerical solution. A possible way around this difficulty is the coupling of the mass and momentum conservation equations through the concept of artificial compressibility (e.g. Chorin, 1967; Rizzi and Eriksson, 1985). In this approach, the original continuity equation for incompressible flow,

$$\nabla \cdot \mathbf{u} = 0$$

is modified by adding an artificial compressibility term:

$$\left(\frac{1}{\rho_d}\right)\left(\frac{\partial p}{\partial t}\right) + c_a^2 \nabla \cdot \mathbf{u} = 0 \quad (\text{B.4.5.1.1.1})$$

where c_a acts as a compressibility parameter and ρ_d is the constant density of the flow. Eqn. (B.4.5.1.1.1) can be deduced from the continuity equation for compressible flows by using the following equation of state:

$$p = c_a^2 \rho_d \quad (\text{B.4.5.1.1.2})$$

The modified inviscid system of partial differential equations consisting of mass and momentum conservation principles now becomes:

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{\Omega} \cdot \left[\frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} \right] = \mathbf{0} \quad (\text{B.4.5.1.1.3})$$

where,

$$\mathbf{Q} = \begin{bmatrix} \frac{p}{\rho_d} \\ u \\ v \\ w \end{bmatrix}$$

$$\mathbf{\Omega} = \begin{bmatrix} c_a^2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{B.4.5.1.1.4})$$

$$\mathbf{E} = \begin{bmatrix} u \\ u^2 + \frac{p}{\rho_d} \\ uv \\ uw \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} v \\ uv \\ v^2 + \frac{p}{\rho_d} \\ vw \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} w \\ uw \\ vw \\ w^2 + \frac{p}{\rho_d} \end{bmatrix}$$

As a result of this preconditioning, the Navier-Stokes equations remain hyperbolic in time regardless of whether they are elliptic or hyperbolic in space. The solution is then marched in this pseudo-time dimension to a steady state, hence artificial compressibility is only present during the transient state.

A preconditioning similar to that proposed by Turkel (1987) is used in *FS*. The standard time derivative of the solution vector that should appear in the governing equations for compressible flow,

$$\frac{\partial \mathbf{Q}}{\partial t} = \begin{bmatrix} \frac{\partial \rho}{\partial t} \\ \frac{\partial(\rho u)}{\partial t} \\ \frac{\partial(\rho v)}{\partial t} \\ \frac{\partial(\rho w)}{\partial t} \\ \frac{\partial(\rho e_t)}{\partial t} \end{bmatrix}$$

is replaced by,

$$\frac{\partial \mathbf{Q}}{\partial t_p} = \begin{bmatrix} \left(\frac{a_1}{c_a}\right)^2 \frac{\partial p}{\partial t_p} \\ \left(\frac{a_2}{c_a}\right)^2 u \frac{\partial p}{\partial t_p} + \frac{\partial(\rho u)}{\partial t_p} \\ \left(\frac{a_3}{c_a}\right)^2 v \frac{\partial p}{\partial t_p} + \frac{\partial(\rho v)}{\partial t_p} \\ \left(\frac{a_4}{c_a}\right)^2 w \frac{\partial p}{\partial t_p} + \frac{\partial(\rho w)}{\partial t_p} \\ \left(\frac{a_5}{c_a}\right)^2 e_t \frac{\partial p}{\partial t_p} + \frac{\partial(\rho e_t)}{\partial t_p} \end{bmatrix} \quad (\text{B.4.5.1.1.5})$$

c_a may be thought of as the artificial speed of pressure wave propagation and is typically scaled on the local flow velocity or to a reference velocity representative of the overall flow motion in question. Moreover, the $a_1, \dots, a_5$ terms are scaling coefficients. The choice of c_a and the scaling coefficients vary depending on the nature of the problem and many have attempted to place particular bounds on these choices. For example, Bet *et al.* (1995) examined the sensitivity of solution

convergence of a two-dimensional laminar boundary layer flow for various artificial compressibility settings. The results are shown in Fig. B.4.5.1.1.1:

Figure B.4.5.1.1.1 – Convergence sensitivity of a laminar boundary-layer simulation to artificial compressibility. Here $\beta = a_1 = a_2 = a_3 = a_4 = a_5$, reproduced from Bet *et al.* (1995)

Simulation parametric studies involving these parameters are given in Appendix D.

Finally, the time-step size is adjusted locally to its maximum value, limited by the numerical stability as per the following relation:

$$\frac{1}{(\Delta t_p)_{i,j,k}} \geq \left(\frac{1}{\Delta t_c} + \frac{1}{\Delta t_d} \right)_{i,j,k} = \left(\frac{\sum \lambda}{K_c V_{c.v.}} + \frac{1}{K_d V_{c.v.}^2} \sum \frac{\mu + \mu_t}{\rho} A_{c.v.}^2 \right)_{i,j,k} \quad (\text{B.4.5.1.1.6})$$

where K_c and K_d are coefficients that dictate the inviscid (convective) and viscous (diffusive) limits of the artificial time step, respectively. Local sizing of the time step warps the transient flow development in the pseudo-time dimension. This however does not pose a concern since the steady solution is independent of the transient path.

B.4.6 RELAXATION SCHEME

In the multigrid scheme, described in Section B.4.8, the discretized governing equations written for each grid node must be solved on each of the multigrid levels.

This solution process, also known as relaxation, can be performed using either explicit or implicit schemes.

Explicit schemes involve the update of fluid flow parameters at each node sequentially. Although such schemes tend to be relatively simple to code and are computationally efficient, they suffer from rather restrictive stability bounds.

For implicit schemes, in contrast, the updating of flow variables at the grid nodes is a coupled process where the discretized governing equations associated with a particular node will also include the flow variables of neighbouring nodes as unknowns. The number of neighbouring nodes involved will depend on the choice of computational volume. The assembly of the discretized equations written at the grid nodes yields an equation set with a sparse coefficient matrix, for which numerous efficient solution methods exist. Some examples of these include Gauss-Seidel, SOR, and ADI methods (see Sections B.4.6.1 and B.4.6.2). Implicit schemes tend to be more expensive computationally, but are generally more stable, hence require less under-relaxation in their iterative solution than explicit schemes. Past experience has shown that the higher computational cost per iteration of implicit schemes is more than offset by the reduction in the number of iterations to convergence.

With the inclusion of the pseudo-time derivatives into the N-S equations discretized in space, we essentially end up with ordinary differential equations with pseudo-time as

the differentiation variable. An implicit or explicit relaxation scheme is then nothing more than a means to numerically integrate these ordinary differential equations. Therefore, marching in pseudo-time can essentially be considered as an under-relaxation to the iterative solution of the Navier-Stokes equations.

As mentioned in Chapter 1, *FS* uses the Runge-Kutta method to carry out this integration. To see this formulation in the current context, we first rearrange Eqn. (B.4.5.1.1) to leave the artificial time derivative alone on the left-hand side and denote the right hand side by $-\mathbf{R}$, the residual vector:

$$V_{c.v.} \frac{\partial \mathbf{Q}_{c.v.}}{\partial t_p} = -\mathbf{R} \quad (\text{B.4.6.1})$$

In *FS*, the number of stages for the Runge-Kutta scheme is user specified. The two-stage version proceeds as follows:

$$\begin{aligned} (\mathbf{Q}_{c.v.})_{i,j,k}^* &= (\mathbf{Q}_{c.v.})_{i,j,k}^n \\ (\mathbf{Q}_{c.v.})_{i,j,k}^{**} &= (\mathbf{Q}_{c.v.})_{i,j,k}^* - \alpha_1 \frac{\Delta t_{i,j,k}}{V_{c.v.}} \mathbf{R}_{i,j,k}^* \\ (\mathbf{Q}_{c.v.})_{i,j,k}^{***} &= (\mathbf{Q}_{c.v.})_{i,j,k}^* - \alpha_2 \frac{\Delta t_{i,j,k}}{V_{c.v.}} \mathbf{R}_{i,j,k}^{**} \\ (\mathbf{Q}_{c.v.})_{i,j,k}^{n+1} &= (\mathbf{Q}_{c.v.})_{i,j,k}^{***} \end{aligned} \quad (\text{B.4.6.2})$$

where, as in Chapter 1, the superscript $*$ denotes an intermediate vector. This formulation, consistent with the approach of other CFD algorithms (Jameson *et al.*, 1981), is implemented to reduce storage requirements, and consequently reduces the formal accuracy of the integration process as compared to classical two-stage Runge-Kutta scheme. However, in this instance, the difference is not significant since the

accuracy of the transient path in the artificial time dimension is inconsequential as far as the steady end state is concerned.

The Runge-Kutta integration of the N-S equations as formulated in Eqn. (B.4.6.2) (and Eqn. (1.5.10)) is an explicit process. As such, the stability limits imposed on the artificial time step through the selection of K_c and K_d are quite restrictive. To circumvent this problem without discarding the Runge-Kutta integration option, the FS algorithm executes the even-numbered stages in an implicit manner. This is referred to as implicit residual smoothing in other, similar algorithms (Jameson *et al.*, 1981). In order to turn the explicit stage of Eqn. (B.4.6.2) into an implicit one, an implicit operator is introduced as follows:

$$\begin{aligned} & \left(\mathbf{I} - \varphi_i \frac{\Delta t_{i,j,k}^2}{V_{c,v,i,j,k}^2} \lambda_i^2 \delta_i^2 \right) \left(\mathbf{I} - \varphi_j \frac{\Delta t_{i,j,k}^2}{V_{c,v,i,j,k}^2} \lambda_j^2 \delta_j^2 \right) \left(\mathbf{I} - \varphi_k \frac{\Delta t_{i,j,k}^2}{V_{c,v,i,j,k}^2} \lambda_k^2 \delta_k^2 \right) \Delta \mathbf{Q} \\ &= \frac{-\alpha_2 \Delta t_{i,j,k}}{V_{c,v,i,j,k}} \mathbf{R}_{i,j,k}^{(1)} \end{aligned} \quad (\text{B.4.6.3})$$

where,

$$\Delta \mathbf{Q} = \mathbf{Q}_{i,j,k}^{(2)} - \mathbf{Q}_{i,j,k}^{(0)} \quad (\text{B.4.6.4})$$

In Eqn. (B.4.6.3), $\mathbf{I}$ is the identity matrix, λ is the spectral radius of the inviscid flux Jacobian, $\varphi_i, \varphi_j, \varphi_k$ are scaling coefficients, and δ is the second-order central difference operator given as:

$$\delta_i^2 \Delta \mathbf{Q} = (\Delta \mathbf{Q}_{i+1} - 2\Delta \mathbf{Q}_i + \Delta \mathbf{Q}_{i-1}) \quad (\text{B.4.6.5})$$

The δ operator is replaced by one-sided differencing at the computational-domain boundaries. The exact mathematical form of the implicit operator does not impact the final steady state solution, but may influence significantly the path the iterative solution follows in pseudo-time.

It is crucial to formulate this operator to yield a diagonally dominant coefficient matrix for the resultant equation set. Moreover, in each row of this matrix the relative signs of the coefficients must be consistent with the physics of fluid flow. The optimum choice for K_c, K_d, α_1 , and α_2 appearing in the Runge-Kutta integration stages, and the coefficients ϕ_i, ϕ_j , and ϕ_k used with the implicit operator, were determined through a series of benchmark simulations given in Appendix D. In the next section we explore two methods of solution of the resultant equation set during the implicit stages: the Alternating Direction Implicit method (ADI) or the Modified Strongly Implicit Procedure (MSIP).

B.4.7 ADI vs. MSIP

Implementation of a finite differencing scheme with a larger computational stencil, yields a system of equations that, unlike its one-dimensional counterpart, is not tridiagonal. The corresponding coefficient matrix is still a sparse one, however, since the matrix is not tridiagonal, the system is more difficult to solve. As a solution to this, Pulliam (1993) describes the procedure known as the Alternating Direction Implicit (ADI) method. This tactic for the discretization of differential equations considers the powerful concept of operator splitting or time splitting. The ADI method updates variables semi-implicitly, along each grid direction sequentially. For

example, in progressing from time step n to time step $(n + 1)$, the i -grid direction treatment for the above Runge-Kutta scheme is:

$$(\mathbf{Q}_{c.v.})_{i,j,k}^{n+\frac{1}{2}} = (\mathbf{Q}_{c.v.})_{i,j,k}^n - \alpha_{1,2} \frac{\Delta t}{V_{c.v.}} \left((R_{i+1,j,k}^{n+\frac{1}{2}} - 2R_{i,j,k}^{n+\frac{1}{2}} + R_{i-1,j,k}^{n+\frac{1}{2}}) + (R_{i,j+1,k}^n - 2R_{i,j,k}^n + R_{i,j-1,k}^n) + (R_{i,j,k+1}^n - 2R_{i,j,k}^n + R_{i,j,k-1}^n) \right) \quad (\text{B.4.7.1})$$

$$(\mathbf{Q}_{c.v.})_{i,j,k}^{n+1} = (\mathbf{Q}_{c.v.})_{i,j,k}^{n+\frac{1}{2}} - \alpha_{1,2} \frac{\Delta t}{V_{c.v.}} \left((R_{i+1,j,k}^{n+1} - 2R_{i,j,k}^{n+1} + R_{i-1,j,k}^{n+1}) + (R_{i,j+1,k}^{n+\frac{1}{2}} - 2R_{i,j,k}^{n+\frac{1}{2}} + R_{i,j-1,k}^{n+\frac{1}{2}}) + (R_{i,j,k+1}^{n+\frac{1}{2}} - 2R_{i,j,k}^{n+\frac{1}{2}} + R_{i,j,k-1}^{n+\frac{1}{2}}) \right) \quad (\text{B.4.7.2})$$

The j - and k -grid directions are treated similarly. The ADI method is not equivalent to a fully implicit or strongly implicit approach for since, as we “sweep” in a particular grid direction, the other two directions are handled explicitly. Consequently, the ADI stability bounds are not as wide as some other more implicit schemes.

Nonetheless, the use of ADI offers substantial improvements in numerical stability over explicit approaches. The advantage of the ADI method is that each sweep along a grid direction requires only the solution of a simple tridiagonal system.

Schneider and Zedan (1981) outline the second of these solution methods, the Modified Strongly Implicit Procedure (MSIP). As the name suggests, the MSIP is in fact a modification of the Strongly Implicit Procedure (SIP) first introduced by Stone (1968). Let us consider Eqn. (B.4.6.3), and for simplicity, we denote the terms

multiplying ΔQ on the left-hand side by A_1 and the right hand side of the equation by $-R$. Thus, we have:

$$A_1 \Delta Q = -R \quad (B.4.7.3)$$

To implement the MSIP, one works with what is called the modified matrix equation.

In the present case, the modified matrix equation is given by:

$$(A_1 + A_2) \Delta Q = -R' \quad (B.4.7.4)$$

where,

$$-R' \equiv -R + A_2 \Delta Q \quad (B.4.7.5)$$

The form of A_2 is such that $\|A_2\| \ll \|A_1\|$, for any choice of norm. By manipulating Eqn. (B.4.7.3) in this way, the LU decomposition of $(A_1 + A_2)$ involves much less computation than the direct decomposition of A_1 . Consequently, the procedure of Schneider and Zedan iterates on the following equivalent form of Eqn. (B.4.7.3):

$$(A_1 + A_2) \Delta Q = (A_1 + A_2) \Delta Q - (A_1 \Delta Q + R) \quad (B.4.7.6)$$

In *FS* the MSIP method is performed at even-numbered Runge-Kutta stages.

Particularly, after dropping the time notation, the k^{th} iterative step looks like:

$$(A_1 + A_2) \Delta Q^{k+1} = (A_1 + A_2) \Delta Q^k - (A_1 \Delta Q^k + R) \quad (B.4.7.8)$$

Next we define a difference vector d and a residual vector R_{res} , according to:

$$\mathbf{d}^{k+1} \equiv \Delta \mathbf{Q}^{k+1} - \Delta \mathbf{Q}^k \quad (\text{B.4.7.9})$$

$$\mathbf{R}_{res.} \equiv -\mathbf{R} - \mathbf{A}_1 \Delta \mathbf{Q}^k \quad (\text{B.4.7.10})$$

Using these two relations, we may now write the iteration Eqn (B.4.7.8) as,

$$(\mathbf{A}_1 + \mathbf{A}_2) \mathbf{d}^{k+1} = \mathbf{R}_{res.}^k \quad (\text{B.4.7.11})$$

Replacing $(\mathbf{A}_1 + \mathbf{A}_2)$ by its LU decomposition yields:

$$\mathbf{LU} \mathbf{d}^{k+1} = \mathbf{R}_{res.}^k \quad (\text{B.4.7.12})$$

At this point we introduce the idea of an intermediate vector $\mathbf{T}$, which is defined as,

$$\mathbf{T}^{k+1} \equiv \mathbf{U} \mathbf{d}^{k+1} \quad (\text{B.4.7.13})$$

Therefore, with this last definition the procedure takes on a two-step nature, as given by the following equations:

$$\mathbf{LT}^{k+1} = \mathbf{R}_{res.}^k \quad (\text{B.4.7.14})$$

$$\mathbf{U} \mathbf{d}^{k+1} = \mathbf{T}^{k+1} \quad (\text{B.4.7.15})$$

Eqns. (B.4.7.14) and (B.4.7.15) represent a forward and backward substitution respectively. Schneider and Zedan (1981) outline some differences in convergence rates between the two methods, namely the MSIP and the ADI method.

B.4.8 THE MULTIGRID METHOD

The Runge-Kutta integration scheme for a given spatial grid is very efficient in reducing errors in the solution that have wavelengths comparable to the spacing of the nodes. Unfortunately, this efficiency diminishes rapidly as the wavelength becomes larger. The multigrid technique is a very useful method for dealing with this problem.

Without loss of generality, let us consider again Eqn. (B.4.7.15) (the following derivation also holds for Eqn. (B.4.7.14)). For simplicity we shall drop the superscript notation:

$$\mathbf{U} \mathbf{d} = \mathbf{T}$$

Let us suppose that we discretize Eqn. (B.4.7.15) on a uniform grid with spacing h .

The resulting discretized equations may be written as:

$$\mathbf{U}_h \mathbf{d}_h = \mathbf{T}_h \quad (\text{B.4.8.1})$$

Let $\tilde{\mathbf{d}}_h$ denote some approximate solution to the exact solution $\mathbf{d}_h$. Then the error, or correction, in $\tilde{\mathbf{d}}_h$ is given by:

$$\mathbf{e}_h = \mathbf{d}_h - \tilde{\mathbf{d}}_h \quad (\text{B.4.8.2})$$

Defining the residual, $\mathbf{r}_h$, as the difference between this approximate solution and the exact solution gives:

$$\mathbf{r}_h = \mathbf{U}_h \tilde{\mathbf{d}}_h - \mathbf{T}_h \quad (\text{B.4.8.3})$$

Since $\mathbf{U}_h$ is linear, we have

$$\mathbf{U}_h \mathbf{e}_h = -\mathbf{r}_h \quad (\text{B.4.8.4})$$

In order to evaluate the error term we seek an approximation to $\mathbf{U}_h$. At this point the notion of multiple grid levels becomes useful. Consider the evaluation of Eqn.

(B.4.8.4) on a “coarser” grid, with mesh size $H > h$ (a common choice is to take $H = 2h$). In *FS*, the coarser grids are constructed by dropping every other node of the next finer grid in all or selective grid directions. Eqn. (B.4.8.4) is then approximated by:

$$\mathbf{U}_H \mathbf{e}_H = -\mathbf{r}_H \quad (\text{B.4.8.5})$$

Since the grid has been coarsened, we now have fewer equations in the discretized form compared to Eqn. (B.4.8.4), and hence, obtaining a solution to this set is less expensive computationally. However, we need some way to define $\mathbf{r}_H$ on the coarse grid in order to solve for the correction $\mathbf{e}_H$. The multigrid method makes use of what is known as a restriction operator Φ that “restricts” $\mathbf{r}_h$ to the coarse grid:

$$\mathbf{r}_H = \Phi \mathbf{r}_h \quad (\text{B.4.8.6})$$

There are several common restriction operators, of which, the simplest choice is a process known as injection. With injection, the corresponding nodes on the coarse grid will simply assume the values of the dependent variables of that same node on the fine grid.

More accurate choices for restriction include a process called full weighting. A very useful way to understand full weighting is to assume that the coarse node of interest is the (2,2) entry of a 3×3 matrix, as shown in Fig. B.4.8.1. The remaining entries of the matrix are the direct and indirectly adjacent nodes of the two-dimensional fine grid. The coarse-node values of interest are obtained by “averaging” the values in the 3×3 matrix using the following scaling factors:

$$\begin{array}{c}
 \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix} \quad \text{Fine Grid Nodes} \\
 \downarrow \\
 \begin{bmatrix} * & * & * \\ * & \text{Gives this node} & * \\ * & * & * \end{bmatrix} \quad \text{Coarse Grid Nodes}
 \end{array}$$

Figure B.4.8.1 – Scaling coefficients for two-dimensional full-weighting restriction

In three-dimensional space the full-weighted restriction to a coarser grid-node involves 27 nodes on the fine grid with the following scaling factors:

Figure B.4.8.2 – Scaling coefficients for three-dimensional full-weighting restriction

These scaling coefficients are applicable to internal nodes. At boundaries, they are adjusted to affect the reduced neighbouring node count such that their sum remains 1.0. In *FS*, one has the option of choosing between injection and full weighting for the restriction process. Although full weighting improves interpolation accuracy for internal nodes, injection is found to be preferable to avoid spurious oscillations near periodic computational-domain boundaries. Simulation results presented in Appendix D use only the injection procedure.

Once we have a solution $\mathbf{e}_H^{temp}$ to Eqn. (B.4.8.5), we use a prolongation operator Ψ that “prolongates” or interpolates the correction to the fine grid:

$$\mathbf{e}_h^{temp} = \Psi \mathbf{e}_H^{temp} \quad (\text{B.4.8.7})$$

The most popular prolongation operator, and the one used in *FS*, is linear interpolation (bi-linear interpolation in two-dimensions and tri-linear interpolation in three-dimensions). For fine-grid nodes that have a ‘match’ located at the same physical coordinates on the coarser grid, the prolongation through linear interpolation would amount to injection. Once this prolongation is complete, we may update our initial approximation as follows:

$$\tilde{\mathbf{d}}_h^{new} = \tilde{\mathbf{d}}_h + \mathbf{e}_h^{temp} \quad (\text{B.4.8.8})$$

This overall process is often referred to as a coarse-grid correction scheme. To summarize, the first step involves computing the residual on the fine grid according to Eqn. (B.4.8.3) and then restriction to the coarse grid by Eqn. (B.4.8.6). Eqn. (B.4.8.5) is then solved exactly on the coarse grid, which yields the correction. Finally we interpolate this correction back to the fine grid using Eqn. (B.4.8.7), and compute the next approximation using Eqn. (B.4.8.8). It is important to note that any high-frequency variations in space, such as $\mathbf{r}_h$ fluctuating rapidly with ‘small’ wavelength ($\leq 2H$), are not representable on the coarse grid. However, these high-frequency components are dealt with efficiently on the fine grid. An example of the multigrid process is shown in Fig. B.4.8.3 for one cycle:

Figure B.4.8.3 – An example of a V-cycle multigrid process with five grid levels

Due to the shape produced by the restriction and prolongation process, this particular cycle is known as the V-cycle. Only V-cycles are available in *FS*, however, alternatives such as W- and F-cycles also exist (Fig. B.4.8.4). F-cycles are more commonly referred to as Full Multigrid cycles. These schemes use either nested V- or W-cycles to compute improved initial guesses on coarser grids before relaxing on the finer grids. Hence, the terminology Full Multigrid V-cycle (FMV) and Full Multigrid W-cycle (FMW) arise. In these alternatives, the coarser grids are visited more often within one cycle. This is considered beneficial since the calculations on coarser grids are not as CPU-time intensive and the solution progresses more rapidly on the coarser grids, facilitated by larger time steps.

Figure B.4.8.4 – Various multigrid strategies, reproduced from Bittencourt *et al.* (2000)

The correction scheme employed in *FS* is known as the Full Approximation Storage (FAS) algorithm (Brandt, 1977), which varies somewhat from the multigrid scheme described above. In FAS, instead of simply storing a correction e_h^{temp} to our initial approximation, as given by Eqn. (B.4.8.8), we store a full current approximation $\tilde{d}_h^{new}$, which is the sum of the correction e_h^{temp} and its approximation on the next coarser grid. Using our notation above, this new storage term is given by:

$$\tilde{d}_h^{new} = \Psi \tilde{d}_H^{new} + e_h^{temp} \quad (\text{B.4.8.9})$$

For linear problems, the traditional multigrid process and the FAS method are identical. The advantage of the FAS method is that it applies equally well to both linear and non-linear problems. Moreover, the slight increase in computational time compared to that of the traditional multigrid method is far offset by its efficiency. For greater detail into this area, see Brandt (1977).

APPENDIX C –MATLAB CODE FOR THE *EE_SOLVER* ALGORITHM

The one-dimensional Euler equation simulation results of Chapter 4 were obtained with a numerical algorithm designed to solve both the characteristic and the non-characteristic form of the equations. The solver, named *EE_Solver*, was written by the author in Matlab and is presented here in order to facilitate future research in the area.

A full description of the various aspects of this code is given in Chapter 3. In as much as possible, the nomenclature for this algorithm, presented on the following pages, adheres to the notation outlined in Chapter 3. Note that the files generated by the simulation are given the generic name Outputfile. This, along with the use of question marks in those areas that are user dependent, is done for the sake of generality.

For matters of brevity, only the CharSolver algorithm is presented here, both with its nomenclature and Matlab code, while the NonCharSolver has been omitted.

The two algorithms are essentially the same, the main difference being the $q_1 - q_3$ terms that were described in Chapter 3 and shown below.

C.1 – *CHAR_SOLVER* NOMENCLATURE

a	Speed of sound (a)
analytic	Corresponds to the pressure value obtained from a numerical simulation conducted on the extended domain
b1, b2	Following a variable name, these refer to one and two point backward differencing, respectively
beta	Buffer zone parameter, β
bufferwidth	Refers to the buffer zone width as a percentage of the computational domain width
dp1m, dp2m, dp3m	The “minus” components of the artificial diffusion terms (Chapter 3)
dp1p, dp2p, dp3p	The “plus” components of the artificial diffusion terms (Chapter 3)
dt	Time step value, dt
dx	Grid spacing value, dx
eps2m, eps4m	The “minus” components of the second and fourth order dissipation terms, respectively (ε_2 and ε_4)
eps2p, eps4p	The “plus” components of the second and fourth order dissipation terms, respectively (ε_2 and ε_4)
eta	Buffer zone parameter, η
f1, f2	Following a variable name, these refer to one and two point forward differencing, respectively
L	Length of the computational domain

m	Exponent (constant) buffer zone parameter (set to 8 for the current study)
mu1, mu2, mu3	Buffer zone parameters, $\mu_1 = \mu_2 = \mu_3 = \mu$
new	Following a variable name this denotes a final value
Nleft	Buffer zone node on the left hand side of the computational domain (in this case Nleft = 1)
Npts	The total number of nodes (buffer zone included)
Nright	Buffer zone on the right hand side of domain having the smallest index (i.e. x_e)
nstep	Simulation counter
p	Pressure, p
q1, q2, q3	The three Riemann invariants for the current problem (note q1 is modified for the current problem, as described in Chapter 3)
Ra1, Ra2, Ra3	Residual terms for the three equations (note Ra1 is not used)
rho	Density, ρ
star	Following a variable name this denotes Intermediate values used in the Runge-Kutta integration scheme
t	Time, t
temp	Used to denote a temporary variable to be updated
tmax	The maximum simulation time
u	Velocity, u
Uprime1-3	Buffer zone parameter, U'_1, U'_2 and U'_3
V	Artificial dissipation pressure switch (Chapter 3)
x	Spatial coordinate, x
xl	The x coordinate value at Nleft

<code>xr</code>	The x coordinate value at Nright
<code>xmax</code>	Spatial coordinate of outer edge of buffer zone
<code>z</code>	Matlab loop counter

CHAR_SOLVER MATLAB CODE

**% INLET FLOW PROBLEM – CHARACTERISTIC FORM OF THE EULER
EQUATIONS WITH BUFFER LAYER IMPLEMENTATION
% 2 STAGE EXPLICIT RUNGE-KUTTA SCHEME WITH ARTIFICIAL
DIFFUSION**

clear

% TEMPORAL SPACING

dt = ?;

tmax = ?;

% SET UP GRID

L = 0.5;

dx = L/?;

bufferwidth = 0.*L;

xmax = L + bufferwidth;

x = -L:dx:xmax-dx;

% The next line is used to vary nodes in the buffer zone

%x = [(-0.5):(dx):(0.49) 0.49 + ((?).*dx):((?).*dx):xmax-dx];

Npts = size(x);

Npts = Npts(2);

% INDICES OF BOUNDARY/REAL DOMAIN INTERFACE

Nleft = 1;

Nright = 100;

xr = x(Nright); x1 = x(Nleft);

%SET UP U' AND μ FOR DOMAIN AND BUFFER ZONE

m = 8;

beta = ?/(x(Npts)-x(Nright)).^m;

eta = ?/(x(Npts)-x(Nright)).^m;

for i = Nleft+1:Nright-1

Uprime1(i) = 0;

mul1(i) = 1;

end

for i = 1:Nleft

Uprime1(i) = beta*(x1-x(i)).^m;

mul1(i) = 1 - eta*(x1-x(i)).^m;

end

for i = Nright:Npts

Uprime1(i) = beta*(x(i)-xr).^m;

mul1(i) = 1 - eta*(x(i)-xr).^m;

end

```

for i = Nleft+1:Nright-1
    Uprime2(i) = 0;
    mu2(i) = 1;
end
for i = 1:Nleft
    Uprime2(i) = beta*(x1-x(i)).^m;
    mu2(i)=1 - eta*(x1-x(i)).^m;
end
for i = Nright:Npts
    Uprime2(i) = beta*(x(i)-xr).^m;
    mu2(i) = 1 - eta*(x(i)-xr).^m;
end

```

```

for i = Nleft+1:Nright-1
    Uprime3(i) = 0;
    mu3(i) = 1;
end
for i = 1:Nleft
    Uprime3(i) = beta*(x1-x(i)).^m;
    mu3(i) = 1 - eta*(x1-x(i)).^m;
end
for i = Nright:Npts
    Uprime3(i) = beta*(x(i)-xr).^m;
    mu3(i) = 1 - eta*(x(i)-xr).^m;
end

```

% INITIAL CONDITIONS

% Velocity

```

for i = 1:1
    u(i) = 0.7;
end
for i = 2:Npts
    u(i) = 0;
end

```

% Pressure

```

for i = 1:Npts-1
    p(i) = 3.4;
end
for i = Npts:Npts
    p(i) = 3.4;
end

```

% Density

```

for i = 1:Npts-1
    rho(i) = 0.5;
end
for i = Npts:Npts
    rho(i) = 0.5;
end

```

```

for i = 1:Npts
a(i) = ((1.4).*((p(i))./(rho(i))))).^(0.5);
end

% Define Q vectors
for i = 1:Npts
q1(i) = rho(i);
q2(i) = (u(i) + (5).*(a(i)));
q3(i) = (u(i) - (5).*(a(i)));
end

% Carry out the Runge-Kutta Integration
% Here we construct the terms to be used in the finite differencing steps
nstep = 0
z = 0
for t = dt:dt:tmax
nstep = nstep+1;

q1f1 = [q1(2:Npts) q1(Npts)];
q1f2 = [q1(3:Npts) q1(Npts) q1(Npts)];
q1b1 = [q1(1) q1(1:Npts-1)];
q1b2 = [q1(1) q1(1) q1(1:Npts-2)];

q2f1 = [q2(2:Npts) q2(Npts)];
q2f2 = [q2(3:Npts) q2(Npts) q2(Npts)];
q2b1 = [q2(1) q2(1:Npts-1)];
q2b2 = [q2(1) q2(1) q2(1:Npts-2)];

q3f1 = [q3(2:Npts) q3(Npts)];
q3f2 = [q3(3:Npts) q3(Npts) q3(Npts)];
q3b1 = [q3(1) q3(1:Npts-1)];
q3b2 = [q3(1) q3(1) q3(1:Npts-2)];

af1 = [a(2:Npts) a(Npts)];
af2 = [a(3:Npts) a(Npts) a(Npts)];
ab1 = [a(1) a(1:Npts-1)];
ab2 = [a(1) a(1) a(1:Npts-2)];

pf1 = [p(2:Npts) p(Npts)];
pf2 = [p(3:Npts) p(Npts) p(Npts)];
pb1 = [p(1) p(1:Npts-1)];
pb2 = [p(1) p(1) p(1:Npts-2)];

uf1 = [u(2:Npts) u(Npts)];
uf2 = [u(3:Npts) u(Npts) u(Npts)];
ub1 = [u(1) u(1:Npts-1)];
ub2 = [u(1) u(1) u(1:Npts-2)];

% Pressure Switch

```

```

Vb1 = (abs(p - 2.*pb1 + pb2))./(p + 2.*pb1 + pb2);
V = (abs(pf1 - 2.*p + pb1))./(pf1 + 2.*p + pb1);
Vf1 = (abs(pf2 - 2.*pf1 + p))./(pf2 + 2.*pf1 + p);

% ε terms
eps2p_1 = (0.25).*max(V,Vf1);
eps4p_1 = max(0, (0.00625 - eps2p_1));
eps2m_1 = (0.25).*max(V,Vb1);
eps4m_1 = max(0, (0.00625 - eps2m_1));

eps2p_2 = (0.25).*max(V,Vf1);
eps4p_2 = max(0, (0.00625 - eps2p_2));
eps2m_2 = (0.25).*max(V,Vb1);
eps4m_2 = max(0, (0.00625 - eps2m_2));

eps2p_3=(0.25).*max(V,Vf1);
eps4p_3=max(0, (0.00625 - eps2p_3));
eps2m_3=(0.25).*max(V,Vb1);
eps4m_3=max(0, (0.00625 - eps2m_3));

% Artificial Diffusion Terms
dplp = (2.*(max(u+a))./(dx)).*(eps2p_1.*((q1f1 - (1./(af1.^2)).*(pf1))-(q1 - (1./(a.^2)).*(p)))) - (2.*(max(u+a))./(dx)).*(eps4p_1.*((q1f2 - (1./(af2.^2)).*(pf2)) - 3.*(q1f1 - (1./(af1.^2)).*(pf1)) + 3.*(q1 - (1./(a.^2)).*(p)) - (q1b1 - (1./(ab1.^2)).*(pb1)))));

dplm = (2.*(max(u+a))./(dx)).*(eps2m_1.*((q1 - (1./(a.^2)).*(p))-(q1b1 - (1./(ab1.^2)).*(pb1)))) - (2.*(max(u+a))./(dx)).*(eps4p_1.*((q1f1 - (1./(af1.^2)).*(pf1)) - 3.*(q1 - (1./(a.^2)).*(p)) + 3.*(q1b1 - (1./(ab1.^2)).*(pb1)) - (q1b2 - (1./(ab2.^2)).*(pb2)))));

dpl = dplp - dplm;
dplp(1) = 0;
dplm(1) = 0;
dpl(1) = dplp(1) - dplm(1);

dpl(2) = (2.*(max(u+a))./(dx)).*((eps2p_1(2)).*((q1f1(2) - (1./(af1(2).^2)).*(pf1(2)))) - 2.*(q1(2) - (1./(a(2).^2)).*(p(2)))+(q1b1(2) - (1./(ab1(2).^2)).*(pb1(2)))) - (2.*(max(u+a))./(dx)).*((eps4p_1(2)).*((q1f2(2) - (1./(af2(2).^2)).*(pf2(2))) - 4.*(q1f1(2) - (1./(af1(2).^2)).*(pf1(2))) + 5.*(q1(2) - (1./(a(2).^2)).*(p(2))) - 2.*(q1b1(2) - (1./(ab1(2).^2)).*(pb1(2)))));

dpl(Npts-1) = (2.*(max(u+a))./(dx)).*((eps2p_1(Npts-1)).*((q1f1(Npts-1) - (1./(af1(Npts-1).^2)).*(pf1(Npts-1)))) - 2.*(q1(Npts-1) - (1./(a(Npts-1).^2)).*(p(Npts-1)))) + (q1b1(Npts-1) - (1./(ab1(Npts-1).^2)).*(pb1(Npts-1)))) - (2.*(max(u+a))./(dx)).*((eps4p_1(Npts-1)).*((q1f1(Npts-1) - (1./(af1(Npts-1).^2)).*(pf1(Npts-1)))) - 4.*(q1(Npts-1) - (1./(a(Npts-1).^2)).*(p(Npts-1))) + 5.*(q1b1(Npts-1) - (1./(ab1(Npts-1).^2)).*(pb1(Npts-1))) - 2.*(q1b2(Npts-1) - (1./(ab2(Npts-1).^2)).*(pb2(Npts-1)))));
dpl(Npts) = 0;

```

```

dp2p = (2.*(max(u+a))./(dx)).*(eps2p_2.*(q2f1-q2)) -
(2.*(max(u+a))./(dx)).*(eps4p_2.*(q2f2 - 3.*q2f1 + 3.*q2 - q2b1));
dp2m = (2.*(max(u+a))./(dx)).*(eps2m_2.*(q2-q2b1)) -
(2.*(max(u+a))./(dx)).*(eps4p_2.*(q2f1 - 3.*q2 + 3.*q2b1 - q2b2));

```

```

dp2 = dp2p - dp2m;
dp2p(1) = 0;
dp2m(1) = 0;
dp2(1) = dp2p(1) - dp2m(1);

```

```

dp2(2) = (2.*(max(u+a))./(dx)).*((eps2p_2(2)).*(q2f1(2)-2.*q2(2) + q2b1(2))) -
(2.*(max(u+a))./(dx)).*((eps4p_2(2)).*(q2f2(2) - 4.*q2f1(2) + 5.*q2(2) -
2.*q2b1(2))));

```

```

dp2(Npts-1) = (2.*(max(u+a))./(dx)).*((eps2p_2(Npts-1)).*(q2f1(Npts-1)-2.*q2(Npts-
1) + q2b1(Npts-1))) - (2.*(max(u+a))./(dx)).*((eps4p_2(Npts-1)).*(q2f1(Npts-1) -
4.*q2(Npts-1) + 5.*q2b1(Npts-1) - 2.*q2b2(Npts-1))));
dp2(Npts) = 0;

```

```

dp3p = (2.*(max(u+a))./(dx)).*(eps2p_3.*(q3f1-q3)) -
(2.*(max(u+a))./(dx)).*(eps4p_3.*(q3f2 - 3.*q3f1 + 3.*q3 - q3b1));

```

```

dp3m = (2.*(max(u+a))./(dx)).*(eps2m_3.*(q3-q3b1)) -
(2.*(max(u+a))./(dx)).*(eps4p_3.*(q3f1 - 3.*q3 + 3.*q3b1 - q3b2));

```

```

dp3 = dp3p - dp3m;
dp3p(1) = 0;
dp3m(1) = 0;
dp3(1) = dp3p(1) - dp3m(1);

```

```

dp3(2) = (2.*(max(u+a))./(dx)).*((eps2p_3(2)).*(q3f1(2)-2.*q3(2) + q3b1(2))) -
(2.*(max(u+a))./(dx)).*((eps4p_3(2)).*(q3f2(2) - 4.*q3f1(2) + 5.*q3(2) -
2.*q3b1(2))));

```

```

dp3(Npts-1) = (2.*(max(u+a))./(dx)).*((eps2p_3(Npts-1)).*(q3f1(Npts-1)-2.*q3(Npts-
1) + q3b1(Npts-1))) - (2.*(max(u+a))./(dx)).*((eps4p_3(Npts-1)).*(q3f1(Npts-1) -
4.*q3(Npts-1) + 5.*q3b1(Npts-1) - 2.*q3b2(Npts-1))));
dp3(Npts) = 0;

```

% Residual Terms

```

Ra2 = dp2 - (Uprime2.*((q2f1 - q2b1)./(2.*dx))) - (mu2.*(u+a).*((q2f1 -
q2b1)./(2.*dx)));

```

```

Ra2(1) = dp2(1) - (Uprime2(1).*((q2f1(1) - q2(1))./(dx))) -
(mu2(1).*(u(1)+a(1)).*((q2f1(1) - q2(1))./(dx)));

```

```

Ra2(Npts) = dp2(Npts) - (Uprime2(Npts).*((q2(Npts) - q2b1(Npts))./(dx))) -
(mu2(Npts).*(u(Npts)+a(Npts)).*((q2(Npts) - q2b1(Npts))./(dx)));

```

```

Ra3 = dp3 - (Uprime3.*((q3f1 - q3b1)./(2.*dx))) - (mu3.*(u+a).*((q3f1 -
q3b1)./(2.*dx)));
Ra3(1) = dp3(1) - (Uprime3(1).*((q3f1(1) - q3(1))./(dx))) -
(mu3(1).*(u(1)+a(1)).*((q3f1(1) - q3(1))./(dx)));

```

```

Ra3(Npts) = dp3(Npts) - (Uprime3(Npts).*((q3(Npts) - q3b1(Npts))./(dx))) -
(mu3(Npts).*(u(Npts)+a(Npts)).*((q3(Npts) - q3b1(Npts))./(dx)));

```

```

q2star = q2 + dt.*Ra2;
q3star = q3 + dt.*Ra3;
q2star(1) = 0.7 + 5.*(1);
q3star(1) = 0.7 - 5.*(1);

```

% Calculation of a and u

```

ustar = (q2star + q3star)./(2);
astar = (q2star - q3star)./(10);
astar(1) = 1;
ustar(1) = 0.7;
astar(Npts) = astar(Npts-1);
ustar(Npts) = ustar(Npts-1);

```

% Calculation of ρ

```

rhostartemp = dt.*dp1 + (((0.4).*(a.^2))./(((1.4).*(a.^2)) - (astar.^2))).*(q1) -
dt.*(Uprime1 + mu1.*(u)).*(((1.4).*(a.^2))./(((1.4).*(a.^2)) - (astar.^2))).*(((q1f1 -
q1b1)./(2.*dx)) - (1./((1.4).*(a.^2))).*(((af1.^2).*(q1f1)) -
((ab1.^2).*(q1b1)))./(2.*dx)));

```

```

rhostartemp(1) = dt.*dp1(1) + (((0.4).*(a(1).^2))./(((1.4).*(a(1).^2)) -
(astar(1).^2))).*(q1(1)) - dt.*(Uprime1(1) +
mu1(1).*(u(1))).*(((1.4).*(a(1).^2))./(((1.4).*(a(1).^2)) - (astar(1).^2))).*(((q1f1(1) -
q1b1(1))./(dx)) - (1./((1.4).*(a(1).^2))).*(((af1(1).^2).*(q1f1(1))) -
((a(1).^2).*(q1b1(1)))./(dx))));

```

```

rhostartemp(Npts) = dt.*dp1(Npts) + (((0.4).*(a(Npts).^2))./(((1.4).*(a(Npts).^2)) -
(astar(Npts).^2))).*(q1(Npts)) - dt.*(Uprime1(Npts) +
mu1(Npts).*(u(Npts))).*(((1.4).*(a(Npts).^2))./(((1.4).*(a(Npts).^2)) -
(astar(Npts).^2))).*(((q1(Npts) - q1b1(Npts))./(dx)) -
(1./((1.4).*(a(Npts).^2))).*(((a(Npts).^2).*(q1(Npts))) -
((ab1(Npts).^2).*(q1b1(Npts)))./(dx))));

```

```

rhostar = (3.5 - (2.5).*((astar.^2)./(a.^2))).*(rhostartemp);
rhostar(1) = rhostar(2);
%rhostar(Npts)=1.0; %Fixed Density when no buffer zone technique is applied

```

```

pstar = ((astar.^2).*(rhostar))./(1.4);

```

% The second half time-step of the Runge-Kutta Integration involves essentially the same steps as shown above. Thus, no comments are included

```

q1f1 = [rhostar(2:Npts) rhostar(Npts)];
q1f2 = [rhostar(3:Npts) rhostar(Npts) rhostar(Npts)];
q1b1 = [rhostar(1) rhostar(1:Npts-1)];
q1b2 = [rhostar(1) rhostar(1) rhostar(1:Npts-2)];

q2f1 = [q2star(2:Npts) q2star(Npts)];
q2f2 = [q2star(3:Npts) q2star(Npts) q2star(Npts)];
q2b1 = [q2star(1) q2star(1:Npts-1)];
q2b2 = [q2star(1) q2star(1) q2star(1:Npts-2)];

q3f1 = [q3star(2:Npts) q3star(Npts)];
q3f2 = [q3star(3:Npts) q3star(Npts) q3star(Npts)];
q3b1 = [q3star(1) q3star(1:Npts-1)];
q3b2 = [q3star(1) q3star(1) q3star(1:Npts-2)];

af1 = [astar(2:Npts) astar(Npts)];
af2 = [astar(3:Npts) astar(Npts) astar(Npts)];
ab1 = [astar(1) astar(1:Npts-1)];
ab2 = [astar(1) astar(1) astar(1:Npts-2)];

pf1 = [pstar(2:Npts) pstar(Npts)];
pf2 = [pstar(3:Npts) pstar(Npts) pstar(Npts)];
pb1 = [pstar(1) pstar(1:Npts-1)];
pb2 = [pstar(1) pstar(1) pstar(1:Npts-2)];

uf1 = [ustar(2:Npts) ustar(Npts)];
uf2 = [ustar(3:Npts) ustar(Npts) ustar(Npts)];
ub1 = [ustar(1) ustar(1:Npts-1)];
ub2 = [ustar(1) ustar(1) ustar(1:Npts-2)];

p = pstar;

Vb1 = (abs(pstar - 2.*pb1 + pb2))./(pstar + 2.*pb1 + pb2);
V = (abs(pf1 - 2.*pstar + pb1))./(pf1 + 2.*pstar + pb1);
Vf1 = (abs(pf2 - 2.*pf1 + pstar))./(pf2 + 2.*pf1 + pstar);

eps2p_1 = (0.25).*max(V,Vf1);
eps4p_1 = max(0, (0.0625 - eps2p_1));
eps2m_1 = (0.25).*max(V,Vb1);
eps4m_1 = max(0, (0.0625 - eps2m_1));

eps2p_2 = (0.25).*max(V,Vf1);
eps4p_2 = max(0, (0.0625 - eps2p_2));
eps2m_2 = (0.25).*max(V,Vb1);
eps4m_2 = max(0, (0.0625 - eps2m_2));

eps2p_3 = (0.25).*max(V,Vf1);
eps4p_3 = max(0, (0.0625 - eps2p_3));
eps2m_3 = (0.25).*max(V,Vb1);
eps4m_3 = max(0, (0.0625 - eps2m_3));

```


$$\begin{aligned} dp1p = & (2.*(max(u+a))./(dx)).*(eps2p_1.*((q1f1 - (1./(af1.^2)).*(pf1))-(q1 - \\ & (1./(a.^2)).*(p)))) - (2.*(max(u+a))./(dx)).*(eps4p_1.*((q1f2 - (1./(af2.^2)).*(pf2)) - \\ & 3.*(q1f1 - (1./(af1.^2)).*(pf1)) + 3.*(q1 - (1./(a.^2)).*(p)) - (q1b1 - \\ & (1./(ab1.^2)).*(pb1))))); \end{aligned}$$

$$\begin{aligned} dp1m = & (2.*(max(u+a))./(dx)).*(eps2m_1.*((q1 - (1./(a.^2)).*(p))-(q1b1 - \\ & (1./(ab1.^2)).*(pb1)))) - (2.*(max(u+a))./(dx)).*(eps4p_1.*((q1f1 - \\ & (1./(af1.^2)).*(pf1)) - 3.*(q1 - (1./(a.^2)).*(p)) + 3.*(q1b1 - (1./(ab1.^2)).*(pb1)) - \\ & (q1b2 - (1./(ab2.^2)).*(pb2))))); \end{aligned}$$

$$\begin{aligned} dp1 &= dp1p - dp1m; \\ dp1p(1) &= 0; \\ dp1m(1) &= 0; \\ dp1(1) &= dp1p(1) - dp1m(1); \end{aligned}$$

$$\begin{aligned} dp1(2) = & (2.*(max(u+a))./(dx)).*((eps2p_1(2)).*((q1f1(2) - (1./(af1(2).^2)).*(pf1(2)))- \\ & 2.*(q1(2) - (1./(a(2).^2)).*(p(2)))+(q1b1(2) - (1./(ab1(2).^2)).*(pb1(2)))) - \\ & (2.*(max(u+a))./(dx)).*((eps4p_1(2)).*((q1f2(2) - (1./(af2(2).^2)).*(pf2(2)) - \\ & 4.*(q1f1(2) - (1./(af1(2).^2)).*(pf1(2))) + 5.*(q1(2) - (1./(a(2).^2)).*(p(2))) - \\ & 2.*(q1b1(2) - (1./(ab1(2).^2)).*(pb1(2))))); \end{aligned}$$

$$\begin{aligned} dp1(Npts-1) = & (2.*(max(u+a))./(dx)).*((eps2p_1(Npts-1)).*((q1f1(Npts-1) - \\ & (1./(af1(Npts-1).^2)).*(pf1(Npts-1)))-2.*(q1(Npts-1) - (1./(a(Npts-1).^2)).*(p(Npts- \\ & 1))) + (q1b1(Npts-1) - (1./(ab1(Npts-1).^2)).*(pb1(Npts-1)))) - \\ & (2.*(max(u+a))./(dx)).*((eps4p_1(Npts-1)).*((q1f1(Npts-1) - (1./(af1(Npts- \\ & 1).^2)).*(pf1(Npts-1))) - 4.*(q1(Npts-1) - (1./(a(Npts-1).^2)).*(p(Npts-1))) + \\ & 5.*(q1b1(Npts-1) - (1./(ab1(Npts-1).^2)).*(pb1(Npts-1))) - 2.*(q1b2(Npts-1) - \\ & (1./(ab2(Npts-1).^2)).*(pb2(Npts-1))))); \\ dp1(Npts) &= 0; \end{aligned}$$

$$\begin{aligned} dp2p = & (2.*(max(u+a))./(dx)).*(eps2p_2.*(q2f1-q2)) - \\ & (2.*(max(u+a))./(dx)).*(eps4p_2.*(q2f2 - 3.*q2f1 + 3.*q2 - q2b1)); \end{aligned}$$

$$\begin{aligned} dp2m = & (2.*(max(u+a))./(dx)).*(eps2m_2.*(q2-q2b1)) - \\ & (2.*(max(u+a))./(dx)).*(eps4p_2.*(q2f1 - 3.*q2 + 3.*q2b1 - q2b2)); \end{aligned}$$

$$\begin{aligned} dp2 &= dp2p - dp2m; \\ dp2p(1) &= 0; \\ dp2m(1) &= 0; \\ dp2(1) &= dp2p(1) - dp2m(1); \end{aligned}$$

$$\begin{aligned} dp2(2) = & (2.*(max(u+a))./(dx)).*((eps2p_2(2)).*(q2f1(2)-2.*q2(2)+q2b1(2))) - \\ & (2.*(max(u+a))./(dx)).*((eps4p_2(2)).*(q2f2(2) - 4.*q2f1(2) + 5.*q2(2) - \\ & 2.*q2b1(2)))); \end{aligned}$$

$$\begin{aligned} dp2(Npts-1) = & (2.*(max(u+a))./(dx)).*((eps2p_2(Npts-1)).*(q2f1(Npts-1)-2.*q2(Npts- \\ & 1) + q2b1(Npts-1))) - (2.*(max(u+a))./(dx)).*((eps4p_2(Npts-1)).*(q2f1(Npts-1) - \\ & 4.*q2(Npts-1) + 5.*q2b1(Npts-1) - 2.*q2b2(Npts-1))); \\ dp2(Npts) &= 0; \end{aligned}$$

```

dp3p = (2.*(max(u+a))./(dx)).*(eps2p_3.*(q3f1-q3)) -
(2.*(max(u+a))./(dx)).*(eps4p_3.*(q3f2 - 3.*q3f1 + 3.*q3 - q3b1));

dp3m = (2.*(max(u+a))./(dx)).*(eps2m_3.*(q3-q3b1)) -
(2.*(max(u+a))./(dx)).*(eps4p_3.*(q3f1 - 3.*q3 + 3.*q3b1 - q3b2));
dp3 = dp3p - dp3m;
dp3p(1) = 0;
dp3m(1) = 0;
dp3(1) = dp3p(1) - dp3m(1);

dp3(2) = (2.*(max(u+a))./(dx)).*((eps2p_3(2)).*(q3f1(2)-2.*q3(2)+q3b1(2))) -
(2.*(max(u+a))./(dx)).*((eps4p_3(2)).*(q3f2(2) - 4.*q3f1(2) + 5.*q3(2) -
2.*q3b1(2)));

dp3(Npts-1) = (2.*(max(u+a))./(dx)).*((eps2p_3(Npts-1)).*(q3f1(Npts-1)-2.*q3(Npts-
1) + q3b1(Npts-1))) - (2.*(max(u+a))./(dx)).*((eps4p_3(Npts-1)).*(q3f1(Npts-1) -
4.*q3(Npts-1) + 5.*q3b1(Npts-1) - 2.*q3b2(Npts-1)));
dp3(Npts) = 0;

Ra2star = dp2 - (Uprime2.*((q2f1 - q2b1)./(2).*dx))) - (mu2.*(ustar+astar).*((q2f1 -
q2b1)./(2).*dx));
Ra2star(1) = dp2(1) - (Uprime2(1).*((q2f1(1) - q2(1))./(dx))) -
(mu2(1).*(ustar(1)+astar(1)).*((q2f1(1) - q2(1))./(dx)));
Ra2star(Npts) = dp2(Npts) - (Uprime2(Npts).*((q2(Npts) - q2b1(Npts))./(dx))) -
(mu2(Npts).*(ustar(Npts)+astar(Npts)).*((q2(Npts) - q2b1(Npts))./(dx)));

Ra3star = dp3 - (Uprime3.*((q3f1 - q3b1)./(2).*dx))) - (mu3.*(ustar+astar).*((q3f1 -
q3b1)./(2).*dx));
Ra3star(1) = dp3(1) - (Uprime3(1).*((q3f1(1) - q3(1))./(dx))) -
(mu3(1).*(ustar(1)+astar(1)).*((q3f1(1) - q3(1))./(dx)));
Ra3star(Npts) = dp3(Npts) - (Uprime3(Npts).*((q3(Npts) - q3b1(Npts))./(dx))) -
(mu3(Npts).*(ustar(Npts)+astar(Npts)).*((q3(Npts) - q3b1(Npts))./(dx)));

q2new = q2 + (0.5).*dt.*(Ra2+Ra2star);
q3new = q3 + (0.5).*dt.*(Ra3+Ra3star);

q2new(1) = 0.7 + 5.*(1);
q3new(1) = 0.7 - 5.*(1);

q2=q2new; q3=q3new;

u=(q2 + q3)./(2);
a=(q2 - q3)./(10);
a(1)=1;
u(1)=0.7;
a(Npts)=a(Npts-1);
u(Npts)=u(Npts-1);

rhotemp = dt.*dp1 + (((0.4).*(a.^2))./(((1.4).*(a.^2)) - (astar.^2))).*(q1) -
dt.*(Uprime1 + mu1.*(u)).((((1.4).*(a.^2))./((1.4).*(a.^2) - (astar.^2))).*(((q1f1 -

```

```
q1b1)/(2.*dx)) - (1./((1.4).*(a.^2))).*((((af1.^2).*(q1f1)) -
((ab1.^2).*(q1b1)))/(2.*dx))));
```

```
rhotemp(1) = dt.*dp1(1) + (((0.4).*(a(1).^2))./(((1.4).*(a(1).^2)) -
(astar(1).^2))).*(q1(1)) - dt.*(Uprime1(1) +
mu1(1).*(u(1))).*(((1.4).*(a(1).^2))./((1.4).*(a(1).^2) - (astar(1).^2))).*(((q1f1(1) -
q1(1))./(dx)) - (1./((1.4).*(a(1).^2))).*((((af1(1).^2).*(q1f1(1))) -
((a(1).^2).*(q1(1))))/(dx))));
```

```
rhotemp(Npts) = dt.*dp1(Npts) + (((0.4).*(a(Npts).^2))./(((1.4).*(a(Npts).^2)) -
(astar(Npts).^2))).*(q1(Npts)) - dt.*(Uprime1(Npts) +
mu1(Npts).*(u(Npts))).*(((1.4).*(a(Npts).^2))./((1.4).*(a(Npts).^2) -
(astar(Npts).^2))).*(((q1(Npts) - q1b1(Npts))./(dx)) -
(1./((1.4).*(a(Npts).^2))).*((((a(Npts).^2).*(q1(Npts))) -
((ab1(Npts).^2).*(q1b1(Npts))))/(dx))));
```

```
rho = (3.5 - (2.5).*((astar.^2)./(a.^2))).*(rhotemp);
rho(1) = rho(2);
%rho(Npts)=1.0;
```

```
p = ((a.^2).*(rho))./(1.4);
```

```
q1 = rho;
q2 = u+5.*a;
q3 = u-5.*a;
```

```
%%%REFERENCE QUANTITIES%%%
```

```
rhoref = 1.1575;
uref = 344.28331;
aref = 344.28331;
pref = (0.5).*rhoref.*(aref.^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% perror calculation
```

```
analytic = ones(1,Npts).*((?));
if mod(nstep,50) == 1
z = z+1
time = t
E = (norm(analytic(Nleft:Nright)-p(Nleft:Nright)))/norm(analytic(Nleft:Nright));
```

```
% OUTPUT FILE INFORMATION
```

```
%% First Data File With Iteration,Time,Pressure,Velocity,a,Density(at selective
nodes), and CPUtime %%
```

```
Iteration_iteration(z) = z;
Time_t(z) = t;
```

```
Error_E(z) = E;
Pressure1_p(z) = p(1);
Pressure50_p(z) = p(50);
Pressure100_p(z) = p(100);
Velocity1_u(z) = u(1);
Velocity50_u(z) = u(50);
Velocity100_u(z) = u(100);
SpeedofSound1_a(z) = a(1);
SpeedofSound50_a(z) = a(50);
SpeedofSound100_a(z) = a(100);
Density1_rho(z) = rho(1);
Density50_rho(z) = rho(50);
Density100_rho(z) = rho(100);
Cputime_cputime(z) = cputime;
```

```
Iteration = Iteration_iteration';
Time = Time_t';
Error = Error_E';
Pressure1 = (Pressure1_p')./pref;
Pressure50 = (Pressure50_p')./pref;
Pressure100 = (Pressure100_p')./pref;
Velocity1 = (Velocity1_u')./uref;
Velocity50 = (Velocity50_u')./uref;
Velocity100 = (Velocity100_u')./uref;
SpeedofSound1 = (SpeedofSound1_a')./aref;
SpeedofSound50 = (SpeedofSound50_a')./aref;
SpeedofSound100 = (SpeedofSound100_a')./aref;
Density1 = (Density1_rho')./rhoref;
Density50 = (Density50_rho')./rhoref;
Density100 = (Density100_rho')./rhoref;
CPUtime = Cputime_cputime';
```

```
Filesave1 = [Iteration Time Error Pressure1 Pressure50 Pressure100 Velocity1
Velocity50 Velocity100 SpeedofSound1 SpeedofSound50 SpeedofSound100
Density1 Density50 Density100 CPUtime];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

%% Second, Third, and Fourth Data Files - Iteration and Dissipation Terms %%

```
Nodes_nodes(z,1:Npts) = (1:Npts);
Dissipation1_dissipation(z,1:Npts) = dp1(1:Npts);
Dissipation2_dissipation(z,1:Npts) = dp2(1:Npts);
Dissipation3_dissipation(z,1:Npts) = dp3(1:Npts);
```

```
Nodes = Nodes_nodes(1,1:Npts)';
Dissipation1 = ((Dissipation1_dissipation').*(dx))./(uref);
Dissipation2 = ((Dissipation2_dissipation').*(dx))./(uref);
Dissipation3 = ((Dissipation3_dissipation').*(dx))./(uref);
```

```
Filesave2 = [Nodes Dissipation1];
```

```

Filesave3 = [Nodes Dissipation2];
Filesave4 = [Nodes Dissipation3];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Fifth Data File - Pressure Snapshots in Time %%
Pressure_pressure(z,1:Npts)=p(1:Npts);

Pressure = (Pressure_pressure')./(pref);

Y0_y0(z,1:Npts)=0.*ones(1,Npts);

Y01_y01(z,1:Npts)=(0.1).*ones(1,Npts);

Y0 = Y0_y0(1,1:Npts)';

Y01 = Y01_y01(1,1:Npts)';

X_x(z,1:Npts)=x(1:Npts);

X = X_x(1,1:Npts)';

Filesave5 = [Nodes Pressure];
Filesave6 = [X Y0 Pressure];
Filesave7 = [X Y01 Pressure];

end
end

save Output file name # 1 Filesave1 -ascii
save Output file name # 2 Filesave2 -ascii
save Output file name # 3 Filesave3 -ascii
save Output file name # 4 Filesave4 -ascii
save Output file name # 5 Filesave5 -ascii
save Output file names #6 and #7 Filesave6 Filesave7 -ascii

```

APPENDIX D – *FS* SIMULATION RESULTS, OPTIMAL PARAMETER SETTINGS AND
COMPUTATIONAL DOMAINS

Figure D.1 – Two-dimensional parallel plate computational domain used in *FS* simulations

Table D.1 – Optimal parameter settings for the two-dimensional parallel plate simulations

c_a	1.00
k_d	0.10
CFL	1.00
Multigrid Levels	2
Runge-Kutta Stages	2
α_1 and α_2	0.5 and 1
κ_2 and κ_4	0.25 and 0.01
V_{ref} V_{local}	$V_{local} = 10 \text{ m/s}$
Relaxation Scheme	MSIP

Figure D.2 – Two-dimensional axisymmetric pipe computational domain used in *FS* simulations

Table D.2 – Optimal paramter settings for the two-dimensional axisymmetric pipe flow simulations

c_a	1.00
k_d	0.10
CFL	1.00
Multigrid Levels	2
Runge-Kutta Stages	2
α_1 and α_2	0.5 and 1
κ_2 and κ_4	0.25 and 0.01
V_{ref} or V_{local}	$V_{local} = 10 \text{ m/s}$
Relaxation Scheme	MSIP

Figure D.3 – Three-dimensional rectangular duct computational domain used in *FS* simulations

Table D.3 – Optimal paramter settings for the three-dimensional rectangular duct flow simulations

c_a	0.50
k_d	0.50
CFL	1.00
Multigrid Levels	2
Runge-Kutta Stages	2
α_1 and α_2	0.5 and 1
κ_2 and κ_4	0.25 and 0.01
V_{ref} or V_{local}	$V_{local} = 10 \text{ m/s}$
Relaxation Scheme	MSIP

Figure D.4 – Three-dimensional (quarter)-pipe computational domain used in *FS* simulations

Table D.4 – Optimal paramter settings for the three-dimensional (quarter)-pipe flow simulations

c_a	1.00
k_d	0.10
CFL	1.80
Multigrid Levels	2
Runge-Kutta Stages	2
α_1 and α_2	0.5 and 1
κ_2 and κ_4	0.25 and 0.001
V_{ref} or V_{local}	$V_{local} = 10 \text{ m/s}$
Relaxation Scheme	MSIP

Figure D.1.1 – Experimental results for $c (= c_a) = 0.5$ and 1.0 (Rectangular Duct)
 (a) Center-line pressure distribution (b) Velocity distribution
 (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals

Figure D.1.2 – Experimental results for $G = 1$ and 2 (Rectangular Duct)
 (a) Center-line pressure distribution (b) Velocity distribution
 (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals

Figure D.1.3 – Experimental results for $G = 2$ and 3 (Rectangular Duct)
 (a) Center-line pressure distribution (b) Velocity distribution
 (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals

Figure D.1.4 – Experimental results for RK = 2 and 4 (Rectangular Duct)
 (a) Center-line pressure distribution (b) Velocity distribution
 (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals

Figure D.1.5 – Experimental results comparing MSIP, ADI and ADI (no multigrid in i direction) (Rectangular Duct)
 (a) Center-line pressure distribution (b) Velocity distribution
 (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals

Figure D.1.6 – Experimental results comparing Interpolation Option 1 and 2 (Rectangular Duct)
 (a) Center-line pressure distribution (b) Velocity distribution
 (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals

Figure D.1.7 – Experimental results comparing the exchange of various data at periodic boundaries (Rectangular Duct)
 (a) Center-line pressure distribution (b) Velocity distribution
 (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals

Figure D.1.8 – Experimental results comparing $\kappa_4 = 0.005$ and $\kappa_4 = 0.01$ (3D Pipe)
 (a) Center-line pressure distribution (b) Velocity distribution
 (c) – (f) Continuity, X-, Y-, and Z-momentum equation residuals

Figure D.1.9 – Experimental results comparing values of $c = c_a$ (2-D Pipe)
 (a) Center-line pressure distribution (b) Velocity distribution
 (c) – (e) Continuity, X-, and Y-momentum equation residuals