I, Bojan Lukovic, hereby submit this as part of the requirements for the degree of:

Doctor of Philosophy

in:

Aerospace Engineering

It is entitled:

Modeling Unsteadiness in Steady Simulations with Neural Network Generated Lumped Deterministic Source Terms

Approved by:

Dr. Paul D. Orkwis
Dr. Mark G. Turner
Dr. Ali A. Minai
Dr. Bruce K. Walker
Dr. Scott D. Hunter
MODELING UNSTEADINESS IN STEADY SIMULATIONS WITH NEURAL NETWORK GENERATED LUMPED DETERMINISTIC SOURCE TERMS

A dissertation submitted to the

Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

DOCTORATE OF PHILOSOPHY
(Ph.D.)

in the Department of Aerospace Engineering and
Engineering Mechanics
of the College of Engineering

2002

by

Bojan Lukovic

B.S., University of Kragujevac, 1994
M.S., University of Kragujevac, 1997

Committee Chair: Dr. Paul D. Orkwis
Abstract

The lumped deterministic source term – neural network (LDST-NN) approach has been developed to obtain quasi-time-average solutions of cavity flows that include unsteady cavity effects in steady-state computations without the cavity. The results obtained with the LDST-NN based steady calculations are compared to the time average of fully unsteady solutions via the shear force acting on the cavity walls. Two orders of magnitude less computational time is required to obtain a quasi-time average simulation relative to time accurate simulations; a substantial savings. The estimated error, based on the calculated drag force, in these simulations is between 4% and 15% as compared to fully unsteady calculations, which is satisfactory for many design purposes. This should be compared to the 40% to 154% errors obtained by neglecting the cavity completely for these same cases. As such, the modified neural network-based LDST model is a viable tool for representing unsteady cavity effects. The LDST-NN quasi-time averaged solution was able to capture global unsteady effects correctly.

The LDSTs were found to correlate directly with observed sound pressure level trends and provide an additional means of assessing unsteadiness. The LDSTs were found to reach a maximum near the cavity/main flow interface but also extended well into the field; indicating that boundary condition representations alone would be inadequate for capturing unsteady effects.

Deterministic source terms were computed from unsteady simulations and modeled with a neural network for use in steady simulations sans cavity to capture the
entire time average effect of the cavity. This was demonstrated for the entire range of Mach numbers, length-to-depth ratios and various translational velocities of the cavity wall.

The results of the study showed that modeling flow over cavities is possible with steady simulations that include source terms provided by a neural network. This method permits a considerable reduction in CPU time and is attractive for large scale simulations since it includes the effects of the unsteady phenomena without computing the unsteady flow inside the cavity.
Acknowledgments

I would like to take this opportunity to express my gratitude to my advisor Dr. Paul Orkwis, for his guidance, support and encouragement throughout the course of my study at the University of Cincinnati. It has been a great privilege and honor to work under his supervision.

I am also grateful to Dr. Mark Turner for his continued interest in my work, advices and support.

Gratitude is extended to Dr. Ali Minai, Dr. Bruce Walker and Dr. Scott Hunter for their assistance, suggestions and encouragement that made the completion of this dissertation possible.

I wish also to thank the personnel of Metacomp Technology for providing the code and for their prompt and accurate assistance.

I extend my sincere gratitude to the University of Cincinnati and the Department of Aerospace Engineering for providing me the University Graduate Scholarship, without which this work would be much more difficult.

In addition, it is impossible to list all the friends I have become acquainted with at the University of Cincinnati. This friendship makes the stay so much more enjoyable.
Table of Contents

Table of Contents ............................................................................................................................ 1

List of Figures .................................................................................................................................. 5

List of Tables .................................................................................................................................. 12

List of Symbols ............................................................................................................................... 13

1 Introduction .................................................................................................................................. 19

1.1 Problem Description ................................................................................................................. 19

1.2 Cavity Flow Physics ................................................................................................................. 21

1.2.1 Previous Cavity Flow Work ................................................................................................. 25

1.2.1.1 Experimental Work ......................................................................................................... 25

1.2.1.2 Computational Work ......................................................................................................... 29

1.2.1.3 Turbomachinery Cavities ................................................................................................. 32

1.2.2 Lumped Deterministic Source Term Technique Background .......................................... 36

1.2.3 Neural Networks in Aerospace Industry .............................................................................. 37

1.3 Objectives and Scope of Present Study ................................................................................... 39
Appendix: Matlab Code for Neural Network Training and Source Term Modeling ............... 163

Bibliography ............................................................................................................................... 191
List of Figures

**Figure 1.1** Photographs of (a) Vulcan Bomber Bomb Bay, and (b) Boeing 737 Wheel Well. ... 20

**Figure 1.2** Schematic of Rotor/Stator Cavity Sealing. ................................................................. 21

**Figure 1.3** (a) Shear-layer and (b) Wake Mode for Subsonic Cavity Flow. .............................. 22

**Figure 1.4** (a) “Open” and (b) “Closed” Cavity Flow and Corresponding Static Pressure
Distributions and Typical Fluctuating Pressure Spectra for Supersonic Flow. (Sketches
reproduced from Ref. 2). ........................................................................................................ 23

**Figure 2.1** Instantaneous and Short-Time Favre-Averaged Density ........................................ 47

**Figure 2.2** Basic Nomenclature for Law-of-the-Wall .............................................................. 55

**Figure 2.3** Basic Nomenclature for Spatial Discretization ........................................................ 60

**Figure 2.4** Grids for $L/D = 1$ and $L/D = 2.75$. ...................................................................... 67

**Figure 2.5** Grids for Cavity with Rotational Walls ................................................................. 68

**Figure 2.6** Schematic Representation of Parallel Multiblock Topology .................................. 69

**Figure 3.1** Instantaneous and Time-Averaged Density ............................................................ 71

**Figure 4.1** Feed-Forward Network Data Relationships ............................................................ 81

**Figure 5.1** Areas Along Which Cavity Drag Force Was Calculated ......................................... 88
Figure 5.2 LDST-NN Model Architecture........................................................................................................... 90

Figure 5.3 Schematic Diagram of the Preparation Phase for LDST-NN Approach. ................... 92

Figure 5.4 Number of Neurons vs. Mean Square Source Term Error for the Source Terms Generating Neural Network........................................................................................................................................ 93

Figure 5.5 Number of Neurons vs. Mean Square Error for Boundary Conditions Generating Neural Network........................................................................................................................................ 94

Figure 6.1 Unsteady Cases Run To Train the Neural Network. .................................................. 98

Figure 6.2 Residual Convergence History. .......................................................................................... 99

Figure 6.3 Unsteady Pressure Signal at the Middle of the Cavity Aft Wall for $M = 0.9$, $Re = 135240$.................................................................................................................................................. 99

Figure 6.4 Pressure Signal and Its Power Spectral Density for $M = 0.6$, $Re = 93869$.................. 100

Figure 6.5 Pressure Signal and Its Power Spectral Density for $M = 0.9$, $Re = 135240$.............. 100

Figure 6.6 Pressure Signal and Its Power Spectral Density for $M = 1.2$, $Re = 171273$............. 101

Figure 6.7 Strouhal Number vs. Mach Number, Corresponding to the Dominant Frequency, for Stationary Cavity. ........................................................................................................................................... 101

Figure 6.8 Pressure Signal and its Power Spectrum for $M = 0.5$, $Re = 79039$......................... 102

Figure 6.9 Subsonic Vorticity Contours ($M = 0.3$, $Re = 48163$) at Different Instances.......... 103

Figure 6.10 Subsonic Schlieren Contours ($M = 0.3$, $Re = 48163$) at Different Instances....... 104

Figure 6.11 Supersonic Vorticity Contours ($M = 1.8$, $Re = 227123$) at Different Instances. .... 105

Figure 6.12 Supersonic Schlieren Contours ($M = 1.8$, $Re = 227123$) at Different Instances. .. 106

Figure 6.13 Pressure Extremes, Quasi-Time-Average Pressure and Pressure Mean................. 108
Figure 6.14 Sound Pressure Level at the Middle of the Cavity Floor for Various Inlet Mach Numbers........................................................................................................................................................................... 109

Figure 6.15 Axial Cavity Drag Force for Various Inlet Mach Numbers......................................................... 110

Figure 6.16 Time Averaged Primitive Variables for $M = 0.3$, $Re = 48163$; $a$) Density, $b$) Mach Number, $c$) Numerical Schlieren Contours, $d$) Z-Direction Vorticity.................................................. 110

Figure 6.17 Time Averaged Primitive Variables for $M = 0.6$, $Re = 93869$; $a$) Density, $b$) Mach Number, $c$) Numerical Schlieren Contours, $d$) Z-Direction Vorticity.................................................. 111

Figure 6.18 Time Averaged Primitive Variables for $M = 1.2$, $Re = 171273$; $a$) Density, $b$) Mach Number, $c$) Numerical Schlieren Contours, $d$) Z-Direction Vorticity.................................................. 111

Figure 6.19 Time Averaged Solution Vorticity Contours for $M = 0.3$-0.8................................................. 112

Figure 6.20 Boundary Layer Momentum Thickness as a function of Mach Number Just After the Cavity End. ............................................................................................................................................................................. 113

Figure 6.21 Source Terms for $M = 0.3$, $Re = 48163$; $a$) Energy, $b$) Mass, $c$) X-Momentum, $d$) Y-Momentum Source Terms............................................................................................................................................................................. 114

Figure 6.22 Source Terms for $M = 1.2$, $Re = 171273$; $a$) Energy, $b$) Mass, $c$) X-Momentum, $d$) Y-Momentum Source Terms............................................................................................................................................................................. 114

Figure 6.23 Subsonic Energy Source Terms ($M = 0.3$-0.8)................................................................. 115

Figure 6.24 Subsonic X-Momentum Source Terms ($M = 0.3$-0.8).................................................... 116

Figure 6.25 Subsonic Y-Momentum Source Terms ($M = 0.3$-0.8).................................................... 117

Figure 6.26 Supersonic Energy Source Terms ($M = 1.3$-2.0)............................................................... 118

Figure 6.27 Supersonic X-Momentum Source Terms ($M = 1.3$-2.0).................................................... 119
Figure 6.28 Supersonic Y-Momentum Source Terms ($M = 1.3-2.0$). ........................................... 120

Figure 6.29 Vorticity Contours Obtained: a) by Time-Averaging unsteady results, b) From a Steady-State Computation Without Including Source Terms and c) From a Steady-State Computation with Source Terms Included. ........................................................................... 122

Figure 6.30 Mach Number Contours Obtained: a) by Time-Averaging unsteady results, b) From a Steady-State Computation Without Including Source Terms and c) From a Steady-State Computation with Source Terms Included. ........................................................................ 122

Figure 6.31 Source Terms vs. $y$ at the Cavity Centerline for Various Incoming Flow Mach Numbers. ............................................................................................................................................ 123

Figure 6.32 Source Terms vs. $y$ at $x = 9$ for Various Incoming Flow Mach Numbers. ........... 124

Figure 6.33 Additional Training Cases Performed with Points Omitted from the Training Set. ...................................................................................................................................................... 124

Figure 6.34 Testing Error for Additional Cases Comparing to Case with All Points Included in Training Set...................................................................................................................................................... 125

Figure 6.35 Neural Network Generated Source Terms for $M = 0.3$, $Re = 48163$; a) Energy, b) X-Momentum, c) Y-Momentum Source Terms. ........................................................................... 126

Figure 6.36 Time Average Solutions Obtained from Time Mean Calculation (left) and Quasi-Time-Average Solution (right) for $M = 0.3$, $Re = 48163$; a) Density, b) Mach Number, c) Numerical Schlieren Contours, d) Z-Direction Vorticity. ............................................................. 128

Figure 6.37 Axial Cavity Drag Force vs. Incoming Flow Mach Numbers Obtained from Time Average Calculation and Steady State Calculation with and without NN Generated Source
Terms. Solid Lines Represent the Time Average of the Unsteady Solution and the Colors Correspond to the Steady State with NN Generated S.T.s. ................................................................. 129

**Figure 6.38** Error in Cavity Drag Force for LDST-NN and Steady State Solution.............. 129

**Figure 6.39** Cases Ran with Unsteady Solver for Various Length-To-Depth Ratios.......... 131

**Figure 6.40** Strouhal Number vs. Length-to-Depth ratio, Corresponding to the Dominant Frequency, for Stationary Cavity, $M = 0.3$ ................................................................. 131

**Figure 6.41** Time Averaged Solution Vorticity Contours, $M = 0.3$ and $L/D = 1-2.75$. ........ 133

**Figure 6.42** Boundary Layer Momentum Thickness as a Function of $L/D$ Ratio for $M = 0.3$. 134

**Figure 6.43** SPL at the aft wall as a Function of $L/D$ Ratio for $M = 0.3$.......................... 134

**Figure 6.44** Cavity Drag Force as a Function of Mach number. ........................................ 135

**Figure 6.45** Cavity Drag Force as a Function of $L/D$ ratio. ............................................ 135

**Figure 6.46** Energy Source Terms for Various Length-to-Depth Ratios ($M = 0.3$).......... 136

**Figure 6.47** $X$-Momentum Source Terms for Various Length-to-Depth Ratios ($M = 0.3$).... 137

**Figure 6.48** $Y$-Momentum Source Terms for Various Length-to-Depth Ratios ($M = 0.3$)..... 138

**Figure 6.49** Source Terms at the Center of the Cavity as a Function of Length-to-Depth ratio ($M = 0.3$), a) Energy, b) $X$-Momentum, c) $Y$-Momentum. ......................................................... 140

**Figure 6.50** Axial Cavity Drag Force vs. Mach Number for Various Incoming Flow Mach Numbers Obtained from the Time Average and the Steady State Calculations with and without NN Generated Source Terms. Solid Lines Represent the Time Average of the Unsteady Solution and the Colors Correspond to the Steady State with NN Generated S.T.s. .................................................................................................................. 140
Figure 6.51 Cases used to Train the Neural Network for Rotational Cavities......................... 142

Figure 6.52 Pressure Signal and Its Power Spectral Density for $M = 0.3$, $Re = 48163$, $v_{wall} = 287.6 \, m/s$. ............................................................................................................................. 143

Figure 6.53 Pressure Signal and Its Power Spectral Density for $M = 0.5$, $Re = 79039$, $v_{wall} = 287.6 \, m/s$. ............................................................................................................................. 143

Figure 6.54 Pressure Signal and Its Power Spectral Density for $M = 0.6$, $Re = 93869$, $v_{wall} = 287.6 \, m/s$. ............................................................................................................................. 144

Figure 6.55 Strouhal Number vs. Mach Number, Corresponding to the Dominant Frequency, for Stationary Cavity. ........................................................................................................... 144

Figure 6.56 Pressure Extremes, Quasi-Time-Average Pressure and Pressure Mean.............. 145

Figure 6.57 Sound Pressure Level at the Center of the Cavity Floor for Various Inlet Mach Numbers...................................................................................................................................... 146

Figure 6.58 Sound Pressure Level at the Center of the Cavity Floor for Various Cavity Wall Velocities. ...................................................................................................................................... 146

Figure 6.59 Axial Cavity Drag Force for Various Mach Numbers. ........................................... 147

Figure 6.60 Axial Cavity Drag Force for Various Cavity Wall Velocities................................ 148

Figure 6.61 Cavity Drag Force in the Z Direction for Various Mach Numbers. ......................... 148

Figure 6.62 Cavity Drag Force in the Z Direction for Various Cavity Wall Velocities. ........... 149

Figure 6.63 Time Averaged Primitive Variables for $M = 0.3$, $w_{wall} = 287.6 \, m/s$. a) Z-Direction Velocity, b) Mach Number, c) Numerical Schlieren Contours, d) $X$-Direction Vorticity, e) $Y$-Direction Vorticity, and f) $Z$-Direction Vorticity. ................................................................. 150
Figure 6.64 Source Terms for $M = 0.3$, $w_{wall} = 286.7 \, \text{m/s}$ a) Energy, b) $X$-Momentum, c) $Y$-Momentum and d) $Z$-Momentum Source Terms. ............................................................... 151

Figure 6.65 Source Terms vs. $y$ at the Cavity Centerline for Various Translating Cavity Wall Velocities ($M = 0.3$). ........................................................................................................... 152

Figure 6.66 Source Terms vs. $y$ at $x = 9$ for Various Translating Cavity Wall Velocities ($M = 0.3$). ..................................................................................................................................... 152

Figure 6.67 Source Terms vs. $y$ at the Cavity Centerline for Various Mach Numbers ($w_{wall} = 286.7 \, \text{m/s}$).................................................................................................................................................................. 153

Figure 6.68 Source Terms vs. $y$ at $x = 9$ for Various Mach Numbers ($w_{wall} = 286.7 \, \text{m/s}$)........ 153

Figure 6.69 Axial Cavity Drag Force vs. Mach Number for Various Cavity Wall Velocities Obtained from the Time Average and the Steady State Calculations with and without Source Terms. Solid Lines Represent the Time Average of the Unsteady Solution and the Colors Correspond to the Steady State with NN Generated S.T.s................................. 154

Figure 6.70 Cavity Drag Force in the Z Direction vs. Mach Number for Rotational Cavity Obtained from the Time Average and the Steady State Calculations with and without Source Terms. Solid Lines Represent the Time Average of the Unsteady Solution and the Colors Correspond to the Steady State with NN Generated S.T.s................................. 154
List of Tables

Table 6.1. Cavity Drag Force Obtained from Time Average Calculation and Steady State Calculation with and without Source Terms
                                                                                                                                             141
List of Symbols

\(a\) Actual Output Vector of a Neural Network

\(A_E\) \(k-\varepsilon\) Turbulence Model Constant

\(b_i (i=1,2,3)\) Polynomial Coefficients

\(b_i'\) Neurons Interconnection Bias

\(c\) Speed of Sound

\(cmp\) Comparison Parameter

\(C_{\varepsilon 1}\) \(k-\varepsilon\) Turbulence Model Constant

\(C_{\varepsilon 2}\) \(k-\varepsilon\) Turbulence Model Constant

\(C_{\mu}\) \(k-\varepsilon\) Turbulence Model Constant

\(D\) Cavity Depth

\(E\) Error Energy Function

\(e\) Neural Network Error

\(e_t\) Stored Energy per Unit Mass

\(F\) Cavity Drag Force
\( f \)  
Frequency

\( FFT \)  
Fast Fourier Transform

\( f(Q^{\text{Riemann}}) \)  
The Riemann Flux

\( f_{\mu} \)  
Low-Reynolds Number Function

\( H \)  
Hessian

\( h \)  
Static Enthalpy of the Gas

\( \mathbf{i} \)  
Unit Vector in \( x \) Direction

\( J \)  
Jacobian

\( \mathbf{j} \)  
Unit Vector in \( y \) Direction

\( k \)  
Turbulent Kinetic Energy

\( \mathbf{k} \)  
Unit Vector in \( z \) Direction

\( \ell \)  
Left Eigenvector

\( L \)  
Cavity Length, Length Scale

\( LDST \)  
Lumped Deterministic Source Term

\( M \)  
Mach Number

\( m \)  
Mode Number

\( n \)  
Coordinate in Normal Direction

\( \hat{n} \)  
Unit Normal Vector

\( \mathbf{n}_i \)  
Normal Movement of the Control Volume

\( P \)  
Primitive Variable Vector
\( \overline{P} \)  Quasi-Time-Average Primitive Variable Vector

\( p \)  Pressure, Neural Network Input Vector

\( \overline{p} \)  Mean Pressure

\( p_0 \)  Stagnation Pressure

\( Pr \)  Prandtl Number

\( Q \)  Conservative Variable Vector

\( \overline{Q} \)  Time-Averaged Conservative Variable Vector

\( Q^\text{in} \)  Instantaneous Fluctuation

\( q_w \)  Heat Transfer Rate at the Wall

\( q_x, q_y, q_z \)  Heat Transfer per Unit Area (corresponding to areas perpendicular to \( x, y \) or \( z \) axis)

\( R \)  Universal Gas Constant

\( Re \)  Reynolds Number

\( r^i \)  Right Eigenvector

\( RHS \)  Right-Hand Side

\( R_k(Q) \)  Vector Residual Operator

\( R_t \)  Turbulence Reynolds number

\( S \)  Source Terms

\( SPL \)  Sound Pressure Level

\( T \)  Temperature

\( t \)  Time, Neural Network Target Vector
\( T_0 \)  
Stagnation Temperature

\( T_t \)  
Realizable Estimate of the Turbulence Timescale

\( u \)  
Velocity in \( x \)-Direction

\( U_c \)  
Velocity \( U_t \) Transformed to Account for Compressibility

\( U_t \)  
Computed Tangent-to-Wall Velocity Magnitude

\( u_t \)  
Friction Velocity

\( V \)  
Volume

\( \vec{V} \)  
Velocity Vector

\( v \)  
Velocity in \( y \)-Direction,

\( v_\infty \)  
Exterior Flow Velocity

\( w \)  
Velocity in \( z \)-Direction

\( w_{ij}' \)  
Neurons Interconnection Weight Factor

\( x \)  
Cartesian Coordinate

\( y \)  
Cartesian Coordinate

\( y^+ \)  
Dimensionless Distance From the Wall to the First Wall-Adjacent Control Volume

\( z \)  
Cartesian Coordinate

Greek Letters

\( \alpha \)  
Experimental Constant in Rossiter’s Equation

\( \alpha_i \)  
Product of the Left Eigenvector and \( \Delta Q \)
\( \delta_0 \) Boundary Layer Thickness

\( \phi \) Weighting Parameter That is Taken from Uni-Directional Concepts

\( \Delta f^- \) Change in Fluxes Across the Left Running Characteristics and Shock Waves

\( \Delta f^+ \) Change in Fluxes across the right Running Characteristics and Shock Waves

\( \varepsilon \) Kinetic Energy Dissipation Rate, very Small Number

\( \gamma \) Ratio of Specific Heats of a Gas

\( \kappa \) Experimental Constant in Rossiter’s Equation

\( \kappa \) Von Karman Constant

\( \kappa^* = C_{\mu}^{1/4} \)

\( \lambda^- \) Negative Eigenvalue

\( \lambda \) Second Viscosity Coefficient

\( \lambda^+ \) Positive Eigenvalue

\( \mu \) Molecular Viscosity, learning rate

\( \mu_{\text{eff}} \) Effective Viscosity

\( \mu_t \) Eddy Viscosity

\( \nu \) Kinematic Viscosity

\( \rho \) Density

\( \sigma_k \) \( k-\varepsilon \) Turbulence Model Constant

\( \sigma_{\varepsilon} \) \( k-\varepsilon \) Turbulence Model Constant
\( \tau \)  Ratio Between Kinetic Turbulence Energy and its Dissipation Rate = \( k / \varepsilon \)

\( \tau_w \)  Shear Stress at the Wall

\( \tau_{ij} \)  Shear Stress (first subscript indicates the axis to which the face is perpendicular, the second indicates the direction to which it is parallel)

\( \xi = \sqrt{R_i / 2} \)

\( \Psi \)  Limiting Coefficient

\( \omega \)  Angular Velocity of Cavity Rotating Walls
1 Introduction

1.1 Problem Description

Cavities are unavoidable in many aerospace applications: aircraft weapon bays and wheel wells (see Figure 1.1), pressure vents in the space shuttle’s cargo bays, and many other applications. However, problems exist with their use, for example, with the deployment of the landing gear, there is an increase of 10 $dB$ in the perceived noise level from the aircraft during landing. Some of this increase can be associated with the presence of a cavity caused by the open landing gear bay. Cavities are also a major source of automobile aerodynamic noise due to flow over open windows or door seals. Elimination of this noise can offer a commercial advantage in the automobile industry by creating a more comfortable and environmentally friendly vehicle. In all of these applications, the designer would like to understand the noise generation mechanisms of cavities or eliminate the occurrence of high-pressure amplitude oscillations because they lead to unsteady loadings on components within or near the cavity. Gaps between rotating and stationary parts of turbomachinery are also important and unavoidable cavities (see Figure 1.2). These gaps prevent high-pressure gas from flowing into a region of low pressure and are impossible to seal completely. They affect the main flow by causing leakage, altering the blockage and loss distributions in the primary flow path, and these in turn may lower the effectiveness and efficiency of a turbomachine from anticipated levels. Overlap and labyrinth seals are usually used to minimize the leakage through the cavity, but they cause kinetic energy dissipation
(around seal teeth) and entropy creation (because of jet mixing with the main flow). For shrouded turbine blades, the leakage flow is always from upstream to downstream of the blade row, causing reduced blade work and pressure drop, while in the case of unshrouded blades, flow leakage may be directed upstream.

![Figure 1.1 Photographs of (a) Vulcan Bomber Bomb Bay, and (b) Boeing 737 Wheel Well.](image)

Although experience has revealed the importance of including the effects of cavities in turbomachines, literature about this problem is still scarce and mostly cavities are neglected in numeric simulations of larger scale bodies because of their small geometry and associated phenomena, like rotation and vibrations. Relatively fine grids are required to model these cavities with almost the same number of grid cells as the primary geometry. In addition, lower velocities in the cavity lead to severe restrictions on the time steps and hence considerable computation time is required, especially for unsteady computations. Elimination of cavities from numerical simulations gives results that deviate from experiments, especially near the cavity. Because of this, it is desirable to develop new methods that take into account both unsteady effects and the effects of complex geometries, such as cavities, without performing unsteady calculations or resolving their geometric details.
Understanding the global flowfield effects of the cavity unsteadiness can help determine the driving mechanisms for the cavity oscillation cycle and help reduce their negative influence on the overall turbomachinery efficiency. The following paragraphs give a description of the current knowledge about cavity flow.

### 1.2 Cavity Flow Physics

Flow over cavities is characterized by large-amplitude oscillations and intense pressure fluctuations, which lead to excessive vibration. In addition, for transonic and supersonic flows, a complex wave system forms in the flow above the cavity, which influences the aerodynamic loading on the structure and introduces changes in the acoustic far field. Currently, it is assumed that shed vortices, as they grow, are convected downstream from the cavity lip. Due to vortex instability, the shear layer deflects upward and downwards resulting in a shear layer/impingement event on the rear wall of the cavity. The shear layer impingement creates an
acoustic wave, which moves upstream at the local speed of sound and affects the front wall, which, in turn, creates new shed vortices.

For subsonic speeds, the cavity has been classified [1] as responding in either “shear-layer mode” or “wake mode” (Figure 1.3). For the shear-layer mode, the shear layer spans the mouth of the cavity and stagnates at the aft wall. The wake mode is characterized by a large-scale vortex shedding and the stagnation of the flow on the cavity floor. In addition, in the wake mode, self-oscillations cease and the drag due to the presence of the cavity greatly increases.

![Diagram](image_url)

**Figure 1.3** (a) Shear-layer and (b) Wake Mode for Subsonic Cavity Flow.

In the case of transonic and supersonic cavity flows, where shocks form above the cavity, cavity flows have been classified as “open”, “closed”, “transitionally open” and “transitionally closed”. When the shear layer separated from the upstream lip reattaches to the downstream face of the cavity, the flow is termed “open” in contrast to the case where the shear layer reattaches to the floor (“closed”). For “open” cavities, there is a single dominant frequency; while for “closed”, one observes no dominant frequency. Figure 1.4 shows sketches reproduced from [Ref. 2] depicting the different cavity flow types. Between those two extremes, there is a range of “transitional” cavities for which there is more than one dominant frequency. Transitionally closed cavity flow is characterized by the coalescence of the impingement shock and the exit shock into a single shock. Transitionally open cavity flow contains a series of expansion and compression wavelets in place of the exit shock wave. Initially, the cut-off for these regimes was given by specific values of the length-to-depth ratio $L/D$. In the literature,
vastly different values were reported for the cutoff ratio, but generally, short and deep cavities were considered “open”. Recently, Tracy and Plentovich [2] and Raman, et al. [3] concluded that the disagreement found in the literature stems from the dependence of the cavity flow type on length-to-depth ratio, as well as on Mach number.

For cavities operating in the shear-layer mode and open cavities, empirical formulae exist for predicting the frequency of oscillation:

\[
f = \frac{c}{D} \left(\frac{2m - 1}{4}\right)
\]  

Figure 1.4 (a) “Open” and (b) “Closed” Cavity Flow and Corresponding Static Pressure Distributions and Typical Fluctuating Pressure Spectra for Supersonic Flow. (Sketches reproduced from Ref. 2).
where \( c \) is the speed of sound, and \( m \) is the mode number. For cavities with larger length-to-depth ratios, in which the shear layer still spans the cavity opening, shear-layer modes dominate the spectrum. The frequencies of these modes for higher speed flows \((M > 0.4)\) are well approximated by the Rossiter equation [4]:

\[
f = \frac{V_\infty}{L} \cdot \frac{m - \alpha}{M + 1/\kappa}
\]

(1.2)

where \( V_\infty \) is the exterior flow speed and \( \alpha \) and \( \kappa \) are constants, experimentally determined as 0.25 and 0.57, respectively. The constant \( \alpha \) accounts for the phase lag between the passage of a disturbance past the cavity trailing edge and the formation of its corresponding upstream traveling disturbance (in fractions of wavelengths) and \( \kappa \) is the ratio of the disturbance convection speed to the free-stream velocity. Heller, et al. [5] proposed a modified version of Rossiter equation, which gives better predictions for oscillations of high-speed cavity flow:

\[
f = \frac{V_\infty}{L} \cdot \frac{m - \alpha}{M + 1/\kappa} \sqrt{1 + \frac{\gamma - 1}{2} M^2}
\]

(1.3)

Besides the frequency, we also need to know the flow regime, dominant mode of oscillation, the amplitude of oscillation and nonlinear interactions between modes. It was found that these parameters depend not only on cavity geometry and flow speed, but also on the boundary layer thickness at the cavity lip. Thus, a cavity simulation may be required just to determine the appropriate regime or the dominant mode of oscillation.

Since cavity flow has a large number of applications, there have been a considerable number of theoretical, numerical and experimental studies aimed at understanding various aspects of the cavity flow dynamics over a wide range of Mach numbers, from low subsonic to
hypersonic. The following section gives a brief history of the experimental and computational work performed in the area of cavity flow, as well as about the research done about turbomachinery cavities.

### 1.2.1 Previous Cavity Flow Work

Cavity flow has been investigated for over 50 years, but because of its unsteady nature the cavity driving mechanism still represents a challenge for both experimental and computational fluid dynamics.

#### 1.2.1.1 Experimental Work

The first experimental work on cavity flows that the author is aware of was published in 1955 by Krishnamurty [6], who investigated the acoustic fields produced by flow over a two-dimensional cavities in both subsonic and supersonic regimes ($\frac{Re}{L} = 0.214 \cdot 10^6 - 0.314 \cdot 10^6$, $M = 0.45 - 0.8$). Krishnamurty took Schlieren pictures of the noise radiation from a cavity and observed that the discrete frequency associated with the periodic pressure fluctuations increased with the air speed and decreased as the cavity length was increased. In addition, it was found that there exists a minimum cavity length at which sound emission is detected. The minimum length decreases as the Mach number increases. Below this minimum length, the shear layer spans the entire opening of the cavity, and no significant acoustic field is observed.

In their study, Plumblee, et al. [7] proposed that the periodic pressure fluctuations in cavities are due to an acoustic resonance excited by the unsteadiness in the turbulent boundary layer approaching the cavity. However, this hypothesis is untenable since the periodic pressure fluctuations are present even when the boundary layer is laminar, as observed by Krishnamurty.
Maull and East [8] observed that cavity flows are highly unsteady and three-dimensional, even for a cavity of low length-to-depth, $L/D$, ratio. As $L/D$ increases, there is a sudden change in the pressure distribution – it becomes unsteady with a discrete fluctuation frequency. This corresponds to the collapse of the large vertical structures found in deeper cavities.

One of the biggest contributions to understanding cavity flow mechanisms was made when Rossiter [4] determined an approximate equation for subsonic flow mode frequencies based on wind-tunnel experiments of subsonic cavity flows. For deeper cavities there is usually one peak in the amplitude spectrum which is very much larger than the others. As the Mach number of the flow increases, the dominant frequency for the periodic pressure fluctuations “jumps” from one value to another, since the flow changes modes. Heller, et al. [5] proposed a modified version of Rossiter’s equation, which gives better predictions for oscillations of high-speed cavity flow.

Sarohia [9] investigated an axisymmetrical cavity flow with a laminar incoming boundary layer and found that the flow oscillations were not a result of resonance phenomena based on the interaction of the separated shear-layer deflection and internal cavity pressures. In addition, it was found that for a length-to-depth ratio below a value of 290, no cavity oscillations occur. Ronneberger [10] performed water tunnel experiments on a flat plate containing a two-dimensional rectangular cut-out ($L/D = 1; M = 0.002; Re = 6000$). The interface was visualized by a thin filament of color dye and the oscillation was recorded with a camera. Based on those results, the behavior of the shear layer is presented along with flow characteristics such as the wall boundary layer profile upstream of the cavity and displacement of the shear layer.
The differences in cavity flow around shallow and deep cavities were discovered and a great deal of research about this problem was performed. Gharib and Roshko [1] investigated cavity drag for different cavity geometries \((L/D = 0.6 – 1.2)\) using water tunnel experiments with a two-dimensional cavity \((M = 0.002; \, Re = 6 \cdot 10^5)\). They noticed that for small length-to-depth ratios, cavity flow did not oscillate. As length-to-depth ratio was increased, drag increased abruptly due to the onset of the “wake mode”. It was also shown by measurement of the momentum balance that the drag of the cavity is related to the state of the shear layer. Erickson and Durgin [11] studied the interaction between standing waves in a deep cavity and shear-layer instabilities \((L/D = 0.05, 0.16; \, M = 0.1-0.2; \, Re = 10^5)\). They employed flow visualization techniques to reveal the vortex formation process and organization as a function of shear layer mode. The results showed that the amplitude of oscillations increases with Mach number. Flow visualization showed that, at lower velocities, a double vortex structure exists at the interface, and at higher velocities, only a single vortex structure is formed on the interface.

Results performed by Gates, et al. [12] \((M = 0.6-1.2; \, Re = 1.04 \cdot 10^8 /ft)\) showed that a shallow cavity \((L/D = 9)\) suppressed tones that existed for the deep cavity configurations \((L/D = 4.5)\). The modal frequencies for the deeper cavity followed the modified Rossiter equation well. Gates, et al. also showed that modal amplitudes for the shallow cavity do not vary with Mach number.

Tracy and Plentovich [1] determined that Reynolds number has little effect on the oscillation frequencies for a subsonic/transonic flow past a cavity, implying it is an inviscid phenomena. They tested geometries that, based on previous reports, should give “open” cavity flow and “closed” cavity flow \((L/D = 4.4-20; \, M = 0.2-0.95; \, Re = 2-100 \cdot 10^6)\) and determined that the “criterion” for open/closed cavity does not only depend upon the geometry of the cavity.
(L/D), but also on the flow Mach number. The tones measured in this experiment showed good agreement with the modified Rossiter equation. Ahuja and Mendoza [13] obtained similar results from their experiment \((L/D = 1.5-6; \ W/L \leq 1 \text{ and } W/L > 1, \ M = 0.25-1; \ Re = 0.45-1.2 \cdot 10^6)\). They reported that the frequency of the peak oscillation was unaffected by changes in cavity width. However, the farfield broadband noise was reduced with decreasing cavity width. In addition, a boundary layer study showed that for \(\delta/L\) greater than 0.7, all tones were eliminated.

In the last decade, cavity flow still attracted the attention of many scientists, because of its application in the airplane and automotive industries. Disimile, et al. [14] focused on characterizing the effect of yaw angle on flow oscillations in a subsonic open cavity \((L/D = 1; \ M = 0.04; \ Re \approx 84,800)\). The peak frequencies in the pressure spectrum did not match those predicted by the Rossiter equation, nor did the acoustic modes of the cavity match. They noted that Rossiter’s equation does not predict the peak frequencies for Mach numbers less than 0.2 well. Cattafesta, et al. [15-16] measured the velocity and density fields of the flow over a subsonic cavity \((L/D = 2; \ M = 0.4-0.6; \ Re = 2.5-3 \cdot 10^6)\) to develop an adaptive weapons-bay suppression system. Their work provided an insight into the nature of the physical feedback loop and the development and convection of vortical structures and their impingement at the trailing edge. Based on obtained results, they proposed a monolithic, piezoelectric flap actuator for cavity noise suppression.

Raman, et al. [2] studied cavity resonance suppression using miniature fluidic oscillators \((M = 0.6-1.32; \ L/D = 3, 6, 8; \ Re = 2.4 \cdot 10^6)\). They used a jet-cavity setup and showed that fluidic excitation could be used in flow and noise control applications. Dix and Bauer [17] performed wind tunnel measurements of three different cavities \((L/D = 4.5, 9, 14.4; \ M = 0.6-5; \ Re = 4.5 \cdot 10^6)\). Their experiments provided pressure spectra from transducers on the cavity floor.
and the upstream and downstream faces and showed that the dominant longitudinal mode frequencies in a rectangular cavity are predicted quite well using the modified Rossiter’s equation.

Advances in computational fluid dynamics in the last twenty-five years enabled more reliable prediction of cavity flows. A great deal of work was reported and published in this area. The chronological review of numerical simulations of cavity flow is given in the next section.

1.2.1.2 Computational Work

Borland [18] was the first to solve the unsteady two-dimensional Euler equations for subsonic flow over an open cavity in 1977. Even though the solution did not match the experimental data, it was the state-of-the-art for that time.

An unsteady compressible Reynolds-averaged Navier-Stokes calculation of supersonic cavity flow \((L/D = 2.3 \text{ and } M = 1.5)\) was performed by Hankey and Shang [19] in 1980. The Cebeci-Smith eddy viscosity turbulence model was employed on a 78 x 52 grid. The results were compared with steady and fluctuating static pressure data.

Gorski, et al. [20] performed numerical calculations of laminar and turbulent flows in a three-dimensional cavity configuration \((M = 2.36; \text{Re} = 6.6 \cdot 10^6 /m)\), a simplified version of the F-111 weapons bay, which had a \(L/D = 6.2\). The results were obtained by solving the Navier-Stokes equations with the k-ε turbulence model.

As more computational resources became available, the computations of cavity flow intensified and were able to give more information about the mechanisms that drive the cavity flow. Srinivasan and Baysal [21] performed a computational investigation of subsonic and transonic flows past three-dimensional deep and transitional cavities \((L/D = 4.4, 11.7; M = 0.58, \text{Re} = 6.6 \cdot 10^6 /m)\) using the k-ε turbulence model.
0.9; $Re = 1.6 \cdot 10^6$) using the Reynolds-Averaged Navier-Stokes (RANS) equations and the explicit MacCormack scheme. The Reynolds stresses were modeled through the Baldwin-Lomax algebraic turbulence model. The computed time-averaged results were used to investigate the effects of Mach number and the incoming boundary-layer thickness. Shih, et al. [22] developed a numerical procedure for the simultaneous implicit numerical solution of the coupled $k$-$\varepsilon$ model and Navier-Stokes equations for compressible viscous flow over the cavity ($L/D = 5; M = 1.5; Re = 1.1 \cdot 10^6$). They concluded that shear-layer deflections cause a complex flowfield, resulting in a series of compression and expansion waves both within the cavity and in the external flow.

Many researchers focused on ways to suppress cavity flow acoustics. Kim and Chokani [23] performed computations of the supersonic turbulent flow over a two-dimensional rectangular cavity with a passive control device ($L/D = 6, 17.5; M = 1.5-1.6; Re = 2 \cdot 10^6$). The simulations were performed using a RANS simulation with the explicit MacCormack scheme and a two-layer algebraic turbulence model due to Cebeci and Smith. The effect of passive control was included with a porous surface over a vent chamber in the floor of the cavity and numerically simulated by the use of a linear form of the Darcy pressure-velocity law. The results indicated that the shear layer is highly unsteady and constantly changes its shape and impingement location during the oscillation and showed that a venting system could be effective in suppressing pressure fluctuations within the cavity. Baysal, et al. [24] numerically investigated the effectiveness of two devices to suppress the cavity acoustics. They performed two-dimensional simulations of transonic, turbulent flows past a cavity ($L/D = 3.0$ and 6.7; $M = 2$) by using the unsteady, compressible RANS equations, solved via a second-order implicit, upwind, finite-volume method (Roe flux-difference splitting). The effect of turbulence was included through the modified Baldwin-Lomax model. Vortices inside the cavity and the separation
characteristics on the cavity floor were noticed to be significantly different for cavities with $L/D$ of 3 and 6.7.

Hardin and Pope [25] calculated sound generated by flow over a subsonic cavity ($M = 0.1$ and $Re = 5000$) by utilizing a two-part technique, where the viscous flow was obtained by calculating the time-dependent incompressible flow, and then the compressibility was handled by computing the difference between the compressible and incompressible flow. This technique allowed obtaining of the cavity flow quickly, but compared well only for low-Mach and Reynolds numbers.

Tam, et al. [26-29] used an upwind finite volume solution (Roe Scheme) of the double thin-layer Navier-Stokes equations to simulate supersonic flow past a cavity ($L/D = 2; M = 2; Re = 3.95 \cdot 10^6$) and compared it to experiments of Disimile, et al. Their work focused on turbulence modeling of the flow past the cavity and the results showed that neither standard Baldwin-Lomax model nor its modifications could perform consistently well across the cavity. With the turbulence model of Baysal and DTNS simulation, they were able to reproduce the supersonic flow past an open cavity qualitatively. These authors also described the oscillation cycle mechanism in detail.

Zhang, et al. [30-33] performed computations on supersonic flow over a shallow cavity ($L/D = 3; M = 1.5-2.5$) by solving RANS equations using a finite volume approach with $k-\omega$ turbulence model. The solutions were compared to data taken previously by Zhang. They showed that the shear layer driven oscillation is characterized by a coupled motion of shear layer flapping in the transverse direction due to the shear layer instability and vortex convection in the streamwise direction due to the non-linear propagation effects leading to significant wave steepening with convection.
Shieh and Morris [34] performed a simulation of unsteady turbulent subsonic cavity flow \((L/D = 2-4.4; \, M = 0.4-0.6; \, Re = 20,000-400,000)\) using RANS equations and the one equation Spalart-Allamaras turbulence model. It was noted that shallow cavities, which in simulations respond in a wake mode, do not respond as such experimentally, probably because the cavity was two-dimensional in the simulation.

As shown in this review, cavity flow is very complicated, and even though quite a few experimental and numerical studies have been carried out, the analysis is far from complete. The mechanisms that drive the cavity flow are still not fully known and the presence of cavities still has a negative influence on turbomachinery efficiency.

### 1.2.1.3 Turbomachinery Cavities

Literature about the effect of cavities on flow in turbomachinery has been aimed largely at the high pressure turbine (HPT) hub cavity with somewhat less work on compressor and low pressure turbine (LPT) cavities. Some research has focused on the flow inside these cavities and studied the effectiveness of various overlap seals using simple cavity and disk geometries.

Denton [35] investigated leakage loss of shrouded blades experimentally. He showed that the swirl velocity of the leakage flow is not greatly changed during this process and remains roughly the same as that of the flow approaching the blade row.

Hart and Turner [36] computed pressure and velocity data in rotor-stator axial gaps and throughflows \((L/D = 0.01-0.115, \, Re = 0.5\cdot10^6-10^7)\). They noted that the largest load is that generated by air pressures acting on the rear face of an impeller. Therefore, the variation in static pressure distribution as bleed air is passed through the impeller rear face cavity is of extreme importance to the design of the internal air system of the engine.
Wilson, et al. [37] performed a 2D steady computation of a relatively simple test rig geometry and the results compared well to the test data, but the Nusselt numbers were underpredicted on the rotor face.

Tekriwal [38] analyzed an actual HPT cavity comparing a 3D mesh that included protruding bolts to a 2D mesh where the bolts were treated as axisymmetric rings. Comparing the results with experimental temperature and windage loss data, Tekriwal concluded that a 3D model that includes the bolt pumping effects was necessary for an accurate heat transfer analysis.

Swoboda, et al. [39] experimentally investigated a low-speed axial flow compressor with four repeating stages. Two tests were run, with the stators configured either in cantilevered or in shrouded form, to investigate the impact of shrouding on efficiency and stall margin. They also investigated the effects of the hub clearance vortex in the cantilevered case.

Eser and Kazakia [40] used a finite-difference code to compute the flow in cavities of labyrinth seals. Periodic, analytical solutions were obtained for the time dependent flow generated by a non-axisymmetric rotation of the shaft, which compared well to experimentally obtained values. The results indicated that the inclusion of shear forces in the calculation is necessary.

Heidegger, et al. [41] performed simultaneous simulations of the seal cavity and the main power stream flow paths of an axial flow compressor to determine the impact of the seal cavity flow on the stator aerodynamic performance. They discovered a large increase in tangential velocity and total temperature for the leakage flow through the seal cavity, the existence of rotating vortices in the seal cavity trenches, and the occurrence of both positive and negative radial flow at the interface with the seal cavity trenches.
Wellborn and Okiishi [42] and Wellborn, et al. [43] performed experiments and numerical simulations on a low-speed multistage axial-flow compressor to assess the effects of shrouded stator cavity flows on aerodynamic performance. They showed the importance of including leakage flows in compressor simulations. The efficiency of the tested compressor progressively degraded as seal leakage increased due to the higher levels of blockage and mixing.

Hunter and Orkwis [44] attempted to include the effects of cavities in simulations by inserting interface region source terms into the Navier-Stokes equations. They showed that these source terms can effectively mimic boundary condition variations and are quite effective in capturing the steady state effect of the cavity. Hunter [45] developed a source term model for the steady bladerow solver to capture wheelspace purgeflow seal effects and predict secondary flow in low-pressure turbines. However, these simulations did not include the unsteady effects produced by cavity flowfield oscillations.

As seen from the above, cavity flow is unsteady and complicated, and when the cavity is just a detail of a turbomachine for which we want to increase efficiency, we need a more sophisticated model to perform reliable computations in order to understand the underlying physics. Once the physics of the flow are known, methods can be proposed for reducing their negative effects on the overall efficiency. To accomplish this task it is necessary to gain more information and quantify the influence of cavity flows on the performance of a turbomachine. It is also needed to determine all parameters that affect the flowfield, like: Mach number, Reynolds number, inflow pressure, turbulence intensity, cavity length-to-depth ratio, mass injection/ejection and the velocity of rotating cavity walls. For the purpose of this study, an academic approach is used with a simplified cavity geometry. Different Mach numbers,
Reynolds numbers, cavity length-to-depth ratios and rotation velocities of cavity walls are included.

Unsteady simulations of such flows require significant CPU time to adequately capture the unsteady effects. It is desirable to find alternate ways to include the unsteadiness effects without performing time accurate calculations. A new method for obtaining information about the flow physics should take into account unsteady effects and the presence of complex geometries, such as cavities, without performing unsteady calculations or resolving their geometric details to avoid extensive computations. One such method is based on field source terms. This method has a promising potential, since knowing source terms allows a quasi-time-average solution to be obtained in steady calculations. In addition, the influence of small geometric details on the large scale solution is obtained in simulations without computing the small-scale fluid dynamic structures.

In this research, field source terms are used, rather than just interface source terms, to model accurately the cavity unsteadiness in gas path simulations with considerable success. The feasibility of the approach has been demonstrated by the author [46-49] by calculating the source terms using the Lumped Deterministic Source Terms (LDST) approach and including them directly in steady calculations to capture the time-average effect of the cavity unsteadiness. Neural networks were then trained to create source terms instead of computing unique source terms for every test case. The general approach taken by this line of research is to develop an LDST module in a simple cavity and then increase its complexity so that cavities similar to purge cavities can be modeled.

In the next section the Lumped Deterministic Stress technique background is given. Several researchers have used this approach to model some unsteadiness effects.
1.2.2 Lumped Deterministic Source Term Technique Background

The history of a Lumped Deterministic Stress technique is rather recent. It was developed by Sondak, et al. [50] and Busby, et al. [51] to model the unsteadiness created by rotor-stator interactions and hot streak migration in turbomachinery. They obtained lumped deterministic stresses from an unsteady inviscid calculation and used a steady viscous computation to simulate the time-averaged viscous solution. The results showed that unsteady flow effects can be modeled as source terms in the steady flow equations. This is similar, but not identical, to the average passage approach of Adamczyk, et al. [52], used for closing the system of Reynolds-averaged Navier-Stokes equations in multi-blade-row turbomachinery simulations. Adamczyk’s method splits the flow quantities into a steady component, an unsteady deterministic (periodic) component and an unsteady random (turbulent) component. The flow is then solved using a procedure similar to Reynolds averaging, where the Reynolds stress terms and deterministic stress terms must be modeled. The approach provides a framework for accounting for the effects of unsteady phenomena on the average-passage flow field, but, the problem is how to model the deterministic stresses, since little data is available. Meneveau and Katz [53] calculated these deterministic stresses in the stator vane passage of a centrifugal pump by time-averaging steady RANS simulations and also performed PIV measurements to obtain the same stresses with experiments. The results showed some agreement, but discrepancies were found and attributed to inability of the turbulence model to account for mid-vane separation.

Ning and He [54] calculated the unsteady vortex shedding stresses from unsteady solutions and then solved the time-averaged Navier-Stokes equations together with the unsteady stresses. They showed that vortex shedding effects can be obtained with a time-independent solver by this technique. The vortex shedding is a natural unsteadiness similar to the cavities
studied here. Even though correlations were made between deterministic source terms and unsteady flow effects in their work, no attempts were made to model deterministic source terms and use them in a steady-state solver. Further study of source terms was needed, especially for modeling without performing unsteady simulations. One way to model them is by utilizing neural networks which can adaptively learn non-linear mappings from input to output space.

Neural networks are currently being used to address a wide range of problems in aeronautics. The next section gives a brief history of neural network applications in the aerospace industry.

### 1.2.3 Neural Networks in Aerospace Industry

Recently, neural networks have been successfully applied to a wide range of problems in the aerospace industry. Numerous papers are reviewed here, but emphasis is placed on using neural networks to “learn” nonlinear mappings based on known data.

LaMarsh, et al. [55] used neural networks in aerodynamic performance optimization of rotor blade design. Their study demonstrated that for several rotor blade designs, neural networks were advantageous in reducing the time required for the optimization.

Faller and Schreck [56] successfully trained neural networks with wind tunnel data to predict surface pressure and aerodynamic coefficients of a pitching wing.

McMillen, et al. [57] demonstrated that neural networks are capable of predicting measured data with sufficient accuracy to enable identification of aircraft instrumentation system degradation. The authors concluded that it is best to use a collection of very small networks, each representing one parameter of interest.
Steck and Rokhsaz [58] showed that a neural network could be successfully trained to predict aerodynamic forces with sufficient accuracy to model a flight control system.

Ross et al. [59] used neural nets trained using a Levenberg-Marquardt algorithm to minimize the amount of data required to define the aerodynamic performance characteristics of a wind tunnel model. It was shown that when only 50% of the data acquired from the wind tunnel test was used to train a neural network, the results had a predictive accuracy equal to or better than the experimental data.

Greenman [60] also used neural networks to minimize the amount of data required to completely define the aerodynamics of a three-element airfoil. The ability of the neural networks to predict the lift, drag or moment coefficient for any high-lift flap deflection, gap and overlap, was demonstrated for both computational and experimental training data sets.

Rai and Madavan [61] demonstrated the feasibility of applying neural nets to redesign a transonic turbine stage to improve its unsteady aerodynamic performance. Their procedure used a sequence of response surfaces based on neural networks to traverse the design space in search of a more accurate polynomial. The optimization procedure yields a modified design that improves the aerodynamics performance through small changes to the reference design geometry. A variation of the method was used by Papila, et al. [62] for optimization of supersonic turbines.

Benning, et al. [63] used neural networks to obtain the flowfield around a sphere for low Reynolds numbers, when a pattern of vortex development is obvious. They succeeded in training the artificial neural network to model the whole flow field as a function of Reynolds number. More complex flows could not be modeled, but Benning, et al. suggested that in such
cases, neural networks could be used to obtain initial conditions for computations, which in turn would shorten the computational time.

Labonté [64] used a neural network to reconstruct a fluid flow from particle-tracking velocimetry data. The network was able to remove noise from the data on the photographs.

Calise, et al. [65] proposed a neural-network-based adaptive flight control system that blends aerodynamic and propulsion actuation for safe flight operation in the presence of actuator failures.

Pindera [66] used neural networks to control flow separation over a NACA0012 airfoil using a boundary layer transpiration strategy through time dependent adjustment of transverse surface velocities. The results showed that neural networks offer a viable means for effective flowfield control.

As shown, neural networks have been successfully used in a large number of aerospace engineering applications. However, they have never been used to generate lumped deterministic source terms in a steady-state solver. It is possible to utilize neural networks with the lumped deterministic source term technique to obtain quasi-time-average solutions without performing unsteady computations. The next section gives the objectives and the scope of the present study.

1.3 Objectives and Scope of Present Study

The primary focus of this study is the development of a lumped deterministic source term – neural network (LDST-NN) approach that gives the global effects of unsteadiness in fluid flow without performing unsteady computations. As a part of this research, the approach is applied to
different cavity flows as a precursor to determining the influence of seal cavities on the flow in turbomachinery. The LDST-NN approach should be used when the level of effort, grid complexity or solver capability to model a specific flowfield is not practical. As an example of the justification for the method, a comparison in CPU time (number of iterations) is given. The unsteady simulations of the cavity flows in this research were run for 60 to 70 characteristic times to allow the solutions to become periodic. In most cases, these simulations required between thirty and sixty thousand iterations. In the case of the quasi-time-average simulations with source terms, one thousand iterations were sufficient to achieve convergence. The savings in CPU time is considerable, since an order less number of iterations is required. The time needed for one steady state iteration is significantly smaller than the time for an unsteady computation, and the grid size is reduced since the cavity can be omitted in the quasi-time-average computations. In addition, once the neural networks are trained, minimal additional computer resources are required to perform runs with different initial conditions and parameters. The estimated error, based on cavity drag force, in all these simulations was between 5 and 13 %, which is satisfactory for design purposes. The general approach is aimed at developing an LDST module for a simple cavity and then increasing its complexity so that cavities more similar to purge cavities can be modeled.

In the next chapter, the numerical procedures implemented in the current study are described. A short-time Favre averaging is applied to the governing equations, and a $k$-$\varepsilon$ turbulence model is presented for closure approximation along with both the wall-function and integration-to-the-wall approaches. In addition, boundary and initial conditions, as well as grids used for unsteady and quasi-time-average simulations are given.
In Chapter 3, lumped deterministic source terms are derived and the mathematical formulation illustrating how they can be used to obtain quasi-time-average solution is given.

Chapter 4 gives a brief description of neural networks and their ability to model non-linear relationships.

In Chapter 5, the unsteady analysis and the analytical tools used on the unsteady data are given. A full description of the LDST-NN approach is given along with an explanation of how neural networks are used to model the source terms and generate quasi-time-average solutions. Different ways are described to improve the neural network efficiency and reduce the amount of training data.

In Chapter 6 the unsteady results of cavity flow are presented. The time-averaged solution obtained by time-averaging unsteady data is given as well as source terms extracted from the unsteady solution. The neural network generated source terms are shown then along with quasi-time-average results obtained from LDST-NN approach.

Chapter 7 draws some overall conclusions from the research and makes recommendations for future work.
2 Numerical Techniques

In this chapter, the numerical procedures used in the current study for steady and unsteady RANS computations are described. The governing equations that describe the flow are introduced first. A short-time Favre averaging is then applied to the governing equations, and a turbulence model is used to model the Favre (Reynolds) stress tensor.

The commercial code CFD++ [67-68], developed by Metacomp Technologies, was used to obtain the unsteady cavity flow and the steady flow with source terms. A multi-dimensional higher-order Total Variation Diminishing (TVD) interpolation was used to avoid spurious numerical oscillations in the computed flow fields. These polynomials are exact fits of multi-dimensional linear data. An approximate Riemann solver was used to define upwind fluxes and preconditioning used for low-speed flows. The code, as employed in this study, has second order spatial accuracy, fourth order accuracy in time, and a finite-volume framework. A wall-distance-free cubic $k$-$\varepsilon$ turbulence model was adopted. This model is tensorially invariant and frame-indifferent and is applicable to moving surfaces. It accounts for normal-stress anisotropy, swirl and streamline curvature effects.

At the end of the chapter, boundary and initial conditions, as well as the grids used for the unsteady and quasi-time-average simulations are given. In addition, the distributed parallel computing paradigm used in this work is presented.
2.1 Governing Equations

The CFD++ code solves for fluid motion governed by the compressible Navier-Stokes equations. Since cavity flow is turbulent and highly unsteady in nature, a short-time Favre averaging should be applied to the governing equations.

The Navier-Stokes equations in conservative form are:

\[
\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = \frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y} + \frac{\partial G_v}{\partial z},
\]  

(2.1)

where:

\[
Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e_t \\ (\rho e_t + p) u \end{bmatrix},
\]

\[
E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho v u \\ \rho w u \\ (\rho e_t + p) u \end{bmatrix},
\]

\[
F = \begin{bmatrix} \rho v \\ \rho v^2 + p \\ \rho v w \\ \rho w v \\ (\rho e_t + p) v \end{bmatrix},
\]

\[
G = \begin{bmatrix} \rho w \\ \rho w u \\ \rho w v \\ \rho w^2 + p \\ (\rho e_t + p) w \end{bmatrix},
\]

and:

\[
E_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u \tau_{xx} + v \tau_{xy} + w \tau_{xz} - q_x \end{bmatrix},
\]

\[
F_v = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u \tau_{yx} + v \tau_{yy} + w \tau_{yz} - q_y \end{bmatrix},
\]

(2.2)

\[
E_v = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u \tau_{zx} + v \tau_{zy} + w \tau_{zz} - q_z \end{bmatrix},
\]
\[ q_x = -k \frac{\partial T}{\partial x}, \quad q_y = -k \frac{\partial T}{\partial y}, \quad q_z = -k \frac{\partial T}{\partial z} \]

\[ \tau_{xx} = \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial u}{\partial x}, \quad \tau_{yy} = \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial v}{\partial y}, \quad \tau_{zz} = \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial w}{\partial z}, \quad \tau_{xy} = \tau_{yx} = \mu \left( \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right), \quad \tau_{xz} = \tau_{zx} = \mu \left( \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right), \quad \tau_{yz} = \tau_{zy} = \mu \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right). \]

(2.3)

Here \( \mu \) is the molecular viscosity which is a weak function of temperature. The coefficient \( \lambda \) is the second viscosity coefficient. For gases, \( \lambda \) was chosen by Stokes so that the sum of the normal stresses \( \tau_{xx}, \tau_{yy} \) and \( \tau_{zz} \) is zero. This so called Stokes hypothesis is commonly used in standard CFD applications, where:

\[ \lambda = -\frac{2}{3} \mu, \quad (2.4) \]

Molecular viscosity increases with temperature because viscous effects are associated with random molecular motion. It can be modeled by Sutherland’s law:

\[ \mu = \mu_{\text{ref}} \left[ \frac{T_{\text{ref}} + T_{\text{Suther}}}{T + T_{\text{Suther}}} \right] ^{1.5} \cdot \left( \frac{T}{T_{\text{ref}}} \right), \quad (2.5) \]

in which \( T_{\text{ref}} = 55.555K \), \( \mu_{\text{ref}} = 3.64 \cdot 10^{-6} \frac{kg}{m \cdot s} \) and \( T_{\text{Suther}} = 110.333K \)

To complete the set of governing equations, a thermal and a caloric equation of state are added:

\[ e_r = c_r T + \frac{1}{2} \left( u^2 + v^2 + w^2 \right), \quad (2.6) \]
The governing equations are often used in nondimensional form. The advantage in doing this is that the characteristic parameters, such as Mach number, Reynolds number and Prandtl number can be varied independently, and flow variables are “normalized” so that their values fall between certain prescribed limits such as 0 and 1. Most common ways to nondimensionalize governing equations are either pressure based or velocity based.

In pressure-based non-dimensionalization the following quantities are used to nondimensionalize the data: \( p_{\text{ref}}, \rho_{\text{ref}}, L_{\text{ref}}, T_{\text{ref}} \) and \( \mu_{\text{ref}} \), while the reference velocity is derived from the square root of the ratio of reference pressure to reference density.

In velocity-based non-dimensionalization, the following quantities are used to nondimensionalize the data: \( v_{\text{ref}}, \rho_{\text{ref}}, L_{\text{ref}}, T_{\text{ref}} \) and \( \mu_{\text{ref}} \), while the reference pressure is derived from the square of the reference velocity multiplied by the reference density.

The nondimensional form of the governing equations has the same form as equations (2.1)-(2.2), while now:

\[
q_x = -\frac{\mu}{R_{\text{ref}} \Pr (\gamma - 1) M_{\text{ref}}^2} \frac{\partial T}{\partial x}, \quad q_y = -\frac{\mu}{R_{\text{ref}} \Pr (\gamma - 1) M_{\text{ref}}^2} \frac{\partial T}{\partial y},
\]

\[
q_z = -\frac{\mu}{R_{\text{ref}} \Pr (\gamma - 1) M_{\text{ref}}^2} \frac{\partial T}{\partial z},
\]

\[
\tau_{xx} = \frac{1}{R_{\text{ref}}} \left[ \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial u}{\partial x} \right], \quad \tau_{yy} = \frac{1}{R_{\text{ref}}} \left[ \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial v}{\partial y} \right],
\]

\[
\tau_{zz} = \frac{1}{R_{\text{ref}}} \left[ \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial w}{\partial z} \right], \quad \tau_{xy} = \tau_{yx} = \frac{\mu}{R_{\text{ref}}} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y},
\]
\[ \tau_{xz} = \tau_{zx} = \frac{\mu}{\text{Re}_\text{ref}} \left( \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right), \quad \tau_{yz} = \tau_{zy} = \frac{\mu}{\text{Re}_\text{ref}} \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right). \]

where for velocity-based non-dimensionalization: \( \text{Re}_\text{ref} = \frac{\rho_{\text{ref}} u_{\text{ref}} L_{\text{ref}}}{\mu_{\text{ref}}}, \quad \text{Pr} = \frac{\mu_c c_p}{k} \) and

\[ M_{\text{ref}} = \frac{u_{\text{ref}}}{\sqrt{\gamma R T_{\text{ref}}}} \] (equal to free stream Mach number), while for pressure-based non-dimensionalization: \( \text{Re}_\text{ref} = \frac{\rho_{\text{ref}} L_{\text{ref}}}{\mu_{\text{ref}}} \sqrt{\frac{p_{\text{ref}}}{\rho_{\text{ref}}}} \) and \( M_{\text{ref}} = \sqrt{\frac{p_{\text{ref}}}{\rho_{\text{ref}}}} = \frac{1}{\sqrt{\gamma}}. \)

Using the above equations to compute all flow variables as functions of time and spatial position is, to say the least, overkill. This approach is usually used as the last resort, when nothing else succeeds, or occasionally to check the validity of a simplified model. For engineering purposes, it is enough to know a few quantitative properties of a turbulent flow, such as the average forces on a surface, or the degree of mixing between two incoming streams of fluid, or perhaps the unsteady solution variations that fall above a particular scale. One simplified method commonly used in CFD is called the Favre-averaging method [69], in which the stochastic unsteadiness is regarded as part of the turbulence. Since the turbulent flow is unsteady (Figure 2.1), the instantaneous value of any given flow variable such as density, \( \rho \) can be decomposed into a time-dependent average, \( \bar{\rho} \) and a small scale high frequency perturbation \( \rho' \), given by:

\[ \rho(x, y, z, t) = \bar{\rho}(x, y, z, t) + \rho'(x, y, z, t), \quad (2.8) \]
\[
\bar{\rho}(x, y, z, t) = \frac{1}{\Delta T} \int_0^{t+\Delta T} \rho(x, y, z, t) \, dt.
\]

**Figure 2.1** Instantaneous and Short-Time Favre-Averaged Density.

The averaging time, \( \Delta T \), is of the order of the computational time step for unsteady simulations, so that small scale high frequency turbulent perturbations (at scales smaller than \( \Delta T \)) are filtered out in short-time averaging.

The mass-averaged velocity, \( \bar{u} \), is defined as:

\[
\bar{u}(x, y, z, t) = \frac{1}{\rho} \frac{1}{\Delta T} \int_0^{t+\Delta T} \rho(x, y, z, t) u(x, y, z, t) \, dt,
\]

and the velocity fluctuation is:

\[
u''(x, y, z, t) = u(x, y, z, t) - \bar{u}(x, y, z, t),
\]

From the definitions of short-time Favre averaged variables, it follows that Favre averaging eliminates density fluctuations from the averaged compressible equations, but does not remove the effect of density fluctuations on turbulence:
\[
\overline{\rho u} = \overline{\rho \cdot \overline{u}}, \quad \overline{\rho u''} = 0 \quad \text{and} \quad u'' \neq 0.
\] (2.10)

Substituting these into the governing equations, Favre-averaged equations are obtained:

\[
\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = \frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y} + \frac{\partial G_v}{\partial z},
\] (2.11)

where:

\[
Q = \begin{bmatrix}
\rho \\
\overline{\rho u} \\
\overline{\rho v} \\
\overline{\rho w} \\
\overline{\rho \varepsilon_r} + \frac{\rho(u'^2 + v'^2 + w'^2)}{2}
\end{bmatrix}, \quad E = \begin{bmatrix}
\overline{\rho \bar{u}} \\
\overline{\rho u''} + \overline{p} \\
\overline{\rho \bar{v}} \\
\overline{\rho u \bar{w}} \\
(\overline{\rho \varepsilon_r} + \overline{p})\overline{u} + \overline{u} \frac{\rho(u'^2 + v'^2 + w'^2)}{2}
\end{bmatrix},
\]

\[
F = \begin{bmatrix}
\overline{\rho \bar{v}} \\
\overline{\rho v \bar{u}} \\
\overline{\rho \bar{v}^2} + \overline{p} \\
\overline{\rho \bar{v} \bar{w}} \\
(\overline{\rho \varepsilon_r} + \overline{p})\overline{v} + \overline{v} \frac{\rho(u'^2 + v'^2 + w'^2)}{2}
\end{bmatrix}, \quad G = \begin{bmatrix}
\overline{\rho \bar{w}} \\
\overline{\rho w \bar{u}} \\
\overline{\rho \bar{w}^2} + \overline{p} \\
(\overline{\rho \varepsilon_r} + \overline{p})\overline{w} + \overline{w} \frac{\rho(u'^2 + v'^2 + w'^2)}{2}
\end{bmatrix},
\]

\[
E_v = \begin{bmatrix}
0 \\
-\frac{\tau_{xx} - \rho u'^2}{\tau_{xx} - \rho u''} \\
-\frac{\tau_{xy} - \rho u''v'}{\tau_{xy} - \rho u''} \\
-\frac{\tau_{xz} - \rho u''w'}{\tau_{xz} - \rho u''} \\
-\frac{u''(\rho \varepsilon_r'' + p'') + \tau_{xx} u'' + \tau_{xy} v'' + \tau_{xz} w''}{\tau_{xx} - \rho u''} + \overline{u}(\tau_{xx} - \rho u'^2) + \overline{v}(\tau_{xy} - \rho u''v'') + \overline{w}(\tau_{xz} - \rho u''w'') - q_x
\end{bmatrix}.
\]
Because of the averaging, the nonlinearity of the governing equations gives rise to terms that must be modeled. To solve for the Favre-averaged (Reynolds) stress tensor:

\[
F_v = \begin{bmatrix}
0 \\
\tau_{xx} - \rho u'' u^n \\
\tau_{yy} - \rho v'' v^n \\
\tau_{zz} - \rho w'' w^n \\
-\nu'' (\rho e'' + p'') + \tau_{yy} u^n + \tau_{yy} v^n + \tau_{yy} w^n \\
+ \bar{u}(\tau_{xx} - \rho u'' u^n) + \bar{v}(\tau_{yy} - \rho v'' v^n) + \bar{w}(\tau_{zz} - \rho w'' w^n) - q_y
\end{bmatrix},
\]

\[
G_v = \begin{bmatrix}
0 \\
\tau_{xx} - \rho w'' w^n \\
\tau_{yy} - \rho w'' v^n \\
\tau_{zz} - \rho w'' w^n \\
-\nu'' (\rho e'' + p'') + \tau_{zz} u^n + \tau_{zz} v^n + \tau_{zz} w^n \\
+ \bar{u}(\tau_{xx} - \rho w'' w^n) + \bar{v}(\tau_{yy} - \rho w'' v^n) + \bar{w}(\tau_{zz} - \rho w'' w^n) - q_z
\end{bmatrix},
\]

which is not present in laminar equations, turbulence model equations are added to the set of governing equations. The complexity of turbulence makes it unlikely that any model will be able to represent all turbulent flows, hence they are regarded as engineering approximations rather than scientific laws.

A two-equation $k$-$\epsilon$ turbulence model was used in this study and hence the explanation of this approach is given in the next section.
2.1.1 $k - \varepsilon$ Turbulence Model

In the $k$-$\varepsilon$ model, the Reynolds-stresses are related to the mean flow strains [68] via the linear Boussinesq relation:

$$\rho u_i u_j = \frac{2}{3} \delta_{ij} \rho k - \mu \left( \frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3} \frac{\partial \tilde{u}_k}{\partial x_k} \delta_{ij} \right).$$  \hspace{1cm} (2.14)

The model consists of the following transport equations for the turbulent kinetic energy $k = \frac{1}{2}(u'^2 + v'^2 + w'^2)$ and its dissipation rate $\varepsilon$:

$$\frac{\partial (\rho k)}{\partial t} + \frac{\partial}{\partial x_j} (\rho u_j k) = \frac{\partial}{\partial x_j} \left[ \left( \mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] + \frac{\partial}{\partial y} \left[ \left( \mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial y} \right] + \frac{\partial}{\partial z} \left[ \left( \mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial z} \right] + P_k - \rho \varepsilon$$  \hspace{1cm} (2.15)

and

$$\frac{\partial (\rho \varepsilon)}{\partial t} + \frac{\partial}{\partial x_j} (\rho u_j \varepsilon) = \frac{\partial}{\partial x_j} \left[ \left( \mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right] + \frac{\partial}{\partial y} \left[ \left( \mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial y} \right] + \frac{\partial}{\partial z} \left[ \left( \mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial z} \right] + C_{\varepsilon 1} P_k - C_{\varepsilon 2} \varepsilon + E$$  \hspace{1cm} (2.16)

in which $P_k$ is the rate of turbulence production, $P_k = -\rho u_i u_j \frac{\partial \tilde{u}_i}{\partial x_j}$, $T_t$ is a realizable estimate of the turbulence timescale: $T_t = \frac{k}{\varepsilon} \max \{ \rho, \zeta^{-1} \}$, $\zeta = \sqrt{R_t / 2}$ and $R_t$ is the turbulence Reynolds number, $R_t = \rho k^2 / (\mu \varepsilon)$. The additional term, $E$, in the dissipation-rate equation is designed to improve the model response to adverse pressure-gradient flows. This term has the form:

$$E = A_{\varepsilon} \rho \sqrt{\varepsilon T} \Psi \max \{ k^{0.5}, (\mu \varepsilon)^{0.25} \},$$  \hspace{1cm} (2.17)

where:
\[ \Psi = \max \left\{ \frac{\partial k}{\partial x_j} \frac{\partial \tau}{\partial x_j}, 0 \right\}, \quad \tau = k / \varepsilon. \]  

(2.18)

The model constants are given by: \( C_\mu = 0.09, \ C_{\varepsilon 1} = 1.44, \ C_{\varepsilon 2} = 1.92, \ \sigma_k = 1.0, \ \sigma_\varepsilon = 1.3, \ A_E = 0.3. \)

The eddy viscosity, \( \mu_t \), is obtained from: \( \mu_t = C_\mu f_\mu \rho k^2 / \varepsilon \), where \( f_\mu \) is a low-Reynolds number function, designed to account for viscous and inviscid damping of turbulent fluctuations in the proximity of solid surfaces:

\[ f_\mu = \frac{1 - e^{-0.01 R}}{1 - e^{-\sqrt{R}}} \max \left\{ 1, \left( \frac{1}{\varepsilon} \right)^{1/2} \right\}. \]  

(2.19)

The implementation of this model in a code is simple: the Favre-averaged equations have the same form as the laminar equations provided the molecular viscosity, \( \mu \), is replaced by the effective viscosity \( \mu_{\text{eff}} = \mu + \mu_t \). The problem here is that the time scales associated with the turbulence are much shorter than those connected with the mean flow. For this reason, an outer iteration of the governing equations is performed first with the value of the eddy viscosity, based on the values of \( k \) and \( \varepsilon \) at the end of the preceding iteration. After this, an outer iteration of the linearized turbulent kinetic energy and dissipation equations is made, after which the eddy viscosity is recalculated. Coupling the equations would make convergence very difficult to obtain. Too large a time step can lead to negative values of either \( k \) or \( \varepsilon \) and numerical instability.

Boundary conditions are needed to solve the turbulence equations. At solid walls, the kinetic energy of turbulence and its first normal-to-wall derivative vanish. The former condition is implemented directly:

\[ k_w = 0. \]  

(2.20)
The wall boundary condition for $\varepsilon$ is based on its wall-limiting value ($n \to 0$):

$$
\varepsilon_w = 2\nu \left( \frac{\partial \sqrt{k}}{\partial n} \right)^2.
$$

(2.21)

At free stream boundaries, the following boundary condition can be used:

$$
k \approx 0, \varepsilon \approx 0, \mu_t = C_\mu \frac{k^2}{\varepsilon} \approx 0.
$$

(2.22)

At the inflow, $k$ and $\varepsilon$ are often not known. In that case, $k$ is usually taken as a small value, approximately $10^{-4} \bar{u}^2$, and the value of $\varepsilon$ should be selected so that the length scale $L = k^{3/2} / \varepsilon$ is approximately $10-20\%$ of $\sqrt{A_{\text{inflow}}}$. For external flows, length scale is adjusted until the ratio $\mu_t / \mu$ in the range of 5-20 is achieved.

The $k$-$\varepsilon$ model can be integrated directly to the wall or used in conjunction with wall functions. The spacing of the normal grid point from the wall is very important in determining the accuracy and the performance of a turbulence model. At a finite Reynolds number, a turbulent boundary layer exhibits three flow regimes: viscous sublayer, logarithmic layer and defect layer. The normal direction is called the $y$ direction for the sake of discussion.

When integrating to the wall, it is recommended that the grid be constructed such that the dimensionless distance from the wall to the first wall-adjacent control volume, $y^+$, is approximately 1, where:

$$
y^+ = \frac{\rho u_* y}{\mu}
$$

(2.23)

and $u_* = \sqrt{\tau_w / \rho}$ is the friction velocity. This can also be written in terms of physical dimensions as
\[ 10^{-4} < \frac{\Delta y}{\delta} < 5 \cdot 10^{-4}. \] (2.24)

where \( \delta \) is a typical boundary layer thickness.

In complex geometries, particularly at high Reynolds numbers, the viscous sublayer of a boundary layer is so thin that the condition \( y^+ < 1 \) can be rather stringent, requiring an excessive total number of grid points. This problem can be avoided by using wall functions, which allow much larger values for \( y^+ \), at the expense of additional modeling assumptions introduced in the wall-fluxes and wall-adjacent cells. Practical predictions of 3D turbulent flows often involve regions of coarse mesh that preclude the application of low-\( Re \) turbulence models for solving the Favre-averaged equations directly to the walls. It is common practice, therefore, to use wall functions, which return acceptable wall fluxes for momentum and energy even though the control volumes nearest to the walls are located deep in the so-called logarithmic overlap region of the boundary layer (\( y^+ > 50 \)). The wall function approach works reliably on both fine and coarse grids (\( 0.1 < y^+ < 200 \)) and it takes into account effects of compressibility, heat transfer and the pressure gradient. It is important to remember that wall functions work best on geometries that do not grossly deviate from a flat plate and for flows subject to mild pressure gradients. The more the topology and flow deviate from these conditions, the less reliable the performance of wall functions. However, there is no general rule as to when to avoid using wall functions, and experience often indicates reasonable results even in cases that sharply deviate from the above guidelines. The next section gives a brief description of the wall function approach and its implementation.
2.1.2 Wall Function Approach

The wall function approach, first suggested by Launder and Spalding, assumes variation of velocity with normal distance from the wall from which a mean shear stress, turbulent kinetic energy and dissipation rate are calculated and used as the wall boundary conditions. The method is based on a velocity scale of $\sqrt{k}$, rather than the more traditional approach of solving for the friction velocity, $u_c$. The two main advantages of using $\sqrt{k}$ as the velocity scale are:

- immunity to reversed flow regions, including separation and re-attachment points where $\tau_w = 0$,

- avoidance of iterative solutions for $u_c$, since $k$ is readily available from the turbulence model.

The logarithmic wall function approach relates the wall shear stress to the relative velocity at the center of the first control volume above the wall (Figure 2.2), and in the case of compressible flow, the relationship is as follows:

$$
\tau_w = \begin{cases} 
\kappa^* \rho_w \sqrt{k} \left( U_{c1} - U_{iw} \right) \ln(E y_v^*), & y_v^* \geq y_v^* \\
\mu_w \left( U_{c1} - U_{iw} \right), & y_v^* < y_v^* 
\end{cases}
$$

(2.25)

where the subscript ‘$w$’ denotes the wall value, and ‘1’ the value in the first control volume next to the wall.

In this equation, $\kappa^* = C_\mu^{1/4} \kappa$, $\kappa = 0.41$ (von Karman constant), $E = 8.8$, $y_v^* = 11.2$, $y^* = C_\mu^{1/4} \rho_w y \sqrt{k} \mu_w$, where $y$ is the local normal-to-wall coordinate, $U_t$ is the computed tangent-to-wall velocity magnitude in the control volume next to the wall and $U_c$ is velocity $U_t$ transformed.
to account for compressibility, heat transfer and pressure gradient effects using the Van Driest approach [70], which results in a low functional variation defined as:

\[
U_c = \sqrt{B} \left[ \sin^{-1} \left( \frac{A + \tilde{U}_l}{D} \right) - \sin^{-1} \left( \frac{A}{D} \right) \right],
\]

(2.26)

where:

\[
A = \frac{q_w}{\tau_w}, \quad B = 2 \frac{C_p T_w}{\text{Pr}_t}, \quad D = \sqrt{A^2 + B}.
\]

(2.27)

Here \( \text{Pr}_t \) is the turbulence Prandtl number (often \( \text{Pr}_t = 0.9 \)), \( q_w \) is the wall heat transfer rate, and

\[
\tilde{U}_l = \begin{cases} 
U_l - \frac{1}{2} \frac{dp}{dt} \left[ \frac{y_v}{\kappa^* \rho \sqrt{k}} \ln \left( \frac{y}{y_v} \right) + \frac{y - y_v}{\kappa^* \rho \sqrt{k}} + \frac{y_v^2}{\mu} \right], & y_v^* \geq y_t^*, \\
U_l, & y_t^* < y_v^*,
\end{cases}
\]

(2.28)

where \( dp/dt \) is the corresponding component of the streamwise pressure gradient there.

A wall-law for temperature is also used:

Figure 2.2 Basic Nomenclature for Law-of-the-Wall
\[ T = T_w - \text{Pr}_T \frac{\bar{U}_t}{C_p} \left[ \frac{q_w}{\tau_w} + \frac{\bar{U}_t}{2} \right], \]  

(2.29)

from which the wall heat transfer rate is derived as:

\[ q_w = \tau_w \left[ \frac{C_p(T_w - T_1)}{\text{Pr}_T(U_{1l} - U_{nw})} - \frac{(U_{1l} - U_{nw})}{2} \right]. \]  

(2.30)

While the equation for \( k \) is solved in the entire domain, the equation for \( \varepsilon \) is not applied in the control volume next to the wall. Instead, \( \varepsilon \) is set using the formula at the logarithmic overlap:

\[ \varepsilon_i = \frac{C_{\mu}^{3/4} k_1^{3/2}}{\kappa y_i}. \]  

(2.31)

The production and dissipation of \( k \) are set in the first control volume (point 1 in Figure 2.2) as follows. For flows without significant pressure gradient, the local equilibrium assumption leads to:

\[ P_k = \frac{\tau_w^2}{\kappa \rho_w \sqrt{k_1 y_i}}, \]  

\[ \varepsilon = \frac{C_{\mu}^{3/4} k_1^{3/2}}{\kappa y_i}. \]  

(2.32)

If pressure gradients are significant, or the boundary layer is otherwise out of equilibrium, a two-layer approach is adopted in which the appropriate levels of \( P_k \) and \( \varepsilon \) are imposed such that the portion of the wall-adjacent cell with \( y^* < y_v^* \) assumes sub-layer values whereas the portion for which \( y^* \geq y_v^* \) assumes log-layer values.

This is followed by integration across the cell to obtain the following averaged levels of production and dissipation rate.
\[
P_k = \frac{\tau_w^2}{2\kappa \rho_w \sqrt{k_1 y_v}} \ln \left( \frac{2y_i}{y_v} \right),
\]

(2.33)

\[
\varepsilon = \frac{k_1}{2y_i} \left[ \frac{2\mu_1}{\rho_i y_v} + \frac{C_{\mu}^{3/4} k_i}{\kappa} \ln \left( \frac{2y_i}{y_v} \right) \right].
\]

If the first control volume is inside the viscous sublayer \((y^* < y_{w^*})\) then the following is used:

\[
P_k = 0,
\]

\[
\varepsilon = \frac{2\mu_1 k_1}{\rho_i y_i^2},
\]

(2.34)

### 2.1.3 Wall Function Approach vs. Integration to the Wall for Cavity Flow

Before the final method was selected for unsteady cavity computations, both the wall-function method and integration-to-the-wall approaches were tested for several cases. Even though satisfactory agreement was achieved with these two approaches for unsteady computations, discrepancies do exist when comparing results obtained with a steady-state solver with inserted lumped deterministic source terms. These discrepancies may be attributed to the wall-function formulation and perhaps to the \(k-\varepsilon\) model formulation. The conclusion is that the wall-function method can provide fast, approximate turbulent flow-field information. It can be used as a first glimpse at a new flow case, to give the user a general feel for the main features of the flow, such as approximate location and size of reversed flow regions and as a preliminary design tool or flow-field initializer prior to the application of more refined grids and models. The wall-function method uses coarse meshes, enabling considerable savings in computing time compared to
integration-to-the-wall Navier-Stokes methods and an order of magnitude decrease relative to wall-function computations.

The wall-function simulation results for LDST-NN computations were not as good as those of the regular Navier-Stokes solver with the $k-\varepsilon$ turbulence model. The reason is a nonlinear formulation of wall functions and coarser grids, which causes loss of lumped deterministic source terms especially at places where they have considerable values. This makes the wall function approach unsuitable for the LDST-NN method and therefore, the wall-function simulation results were not included in this study. All cases performed for neural network training and its testing used the integration-to-the-wall approach. Next, the methodology is given for spatial and time discretization of the governing equations.

### 2.2 Spatial and Time Discretization

CFD++ utilizes a finite volume approach to discretize the governing differential equations. Assume a computational cell with volume $V$. The differential form of the equations is integrated over this volume in the following manner:

$$\int\int\int_{V} \left( \frac{\partial Q}{\partial t} + \frac{\partial (E - E_v)}{\partial x} + \frac{\partial (F - F_v)}{\partial y} + \frac{\partial (G - G_v)}{\partial z} - S \right) dxdydz = 0,$$

where $S$ represents the source terms (production and dissipation of turbulence and body forces if necessary). The spatial terms can be rewritten as:

$$\frac{\partial (E - E_v)}{\partial x} + \frac{\partial (F - F_v)}{\partial y} + \frac{\partial (G - G_v)}{\partial z} = \nabla \cdot \tilde{R},$$

where:

$$\tilde{R} = (E - E_v)\hat{i} + (F - F_v)\hat{j} + (G - G_v)\hat{k}.$$
Using Gauss and Leibnitz’s rules, the equation (2.35) can be written as:

\[
\frac{\partial (\bar{Q}V)}{\partial t} + \iiint_A (\vec{R} \cdot \hat{n} + \vec{n}_i \bar{Q})dA = \iiint_V \dot{S}dV,
\] (2.37)

where \( \bar{Q} \) is the cell average of the dependent variables, \( \hat{n} \) is the outward pointing unit normal and \( \vec{n}_i \) is normal movement of the control volume (see Figure 2.3). These quantities are defined as the following:

\[
\bar{Q} = \frac{1}{V} \iiint_V \bar{Q}dV, \quad \hat{N} = n_x \vec{i} + n_y \vec{j} + n_z \vec{k}, \quad \vec{n}_i = -\hat{n}_x \vec{i} - \hat{n}_y \vec{j} - \hat{n}_z \vec{k}.
\] (2.38)

The integral term \( \iiint_A (\vec{R} \cdot \hat{n} + \vec{n}_i \bar{Q})dA \), in the equations is evaluated by a midpoint rule formula:

\[
\iiint_A (\vec{R} \cdot \hat{n} + \vec{n}_i \bar{Q})dA \approx \sum_{\text{Faces}} (\vec{R}_i \cdot \hat{n}_i + \vec{n}_i \bar{Q}_i) A_i, \] (2.39)

where \( i \) denotes the control volume location of each face. In order to calculate the fluxes and cell-face control volumes (midpoint integration), pointwise values are needed at those locations. These pointwise values have to be reconstructed from control volume values (which are the same as the cell averages up to second order).

A multi-dimensional linear polynomial can be written out with four coefficients in the following way:

\[
B(x, y, z) = b_0 + b_1(x - x_c) + b_2(y - y_c) + b_3(z - z_c)
\] (2.40)

where \( x_c, y_c \) and \( z_c \) are the coordinates of a control volume center. The polynomial coefficients for this type of polynomial are calculated using the following formulae:
Figure 2.3 Basic Nomenclature for Spatial Discretization.

\[ b_0 = Q_i \]
\[ b_1 = \frac{1}{V} \sum_{k} \frac{Q_i + Q_k}{2} n_x \]
\[ b_2 = \frac{1}{V} \sum_{k} \frac{Q_i + Q_k}{2} n_y \]
\[ b_3 = \frac{1}{V} \sum_{k} \frac{Q_i + Q_k}{2} n_z \] (2.41)

For vertices lying on any boundary, a polynomial is not initially constructed. After an extrapolation is done to compute boundary conditions, these boundary conditions can then be included in polynomial evaluations.

In order to avoid introducing new maxima and minima in polynomial evaluations, a Multi-Dimensional TVD approach is employed. For each face, “in-face” $\Delta Q$ and “out-of-face” $\Delta Q$ quantities are computed and compared using the following limiters:
\[ \Delta Q_{\text{TVD}} = \frac{\text{sign}(\Delta Q_{\text{IF}}) + \text{sign}(\Delta Q_{\text{OF}})}{2} \min\left(\frac{\Delta Q_{\text{IF}}}{\text{cmp} \Delta Q_{\text{OF}}}, c\right) \] (Minmod limiter) \tag{2.42}

\[ \Delta Q_{\text{TVD}} = \left[\text{sign}(\Delta Q_{\text{IF}}) + \text{sign}(\Delta Q_{\text{OF}})\right] \frac{|\Delta Q_{\text{IF}}| + |\Delta Q_{\text{OF}}|}{|\Delta Q_{\text{IF}}| + |\Delta Q_{\text{OF}}| + \varepsilon} \] (Continuous limiter)

where \( \text{cmp} \) is the compression parameter and takes on the values between one and two and \( \varepsilon \) is a very small number that is used to avoid division by zero in the numerical implementation of the continuous limiter. Once the TVD \( \Delta Q \)'s are calculated for each cell face, a weighted average of these quantities is:

\[ \Delta Q_{\text{TVD}} = \frac{1+\phi}{2} \left(\text{in - face}\right) + \frac{1-\phi}{2} \left(\text{out - of - face}\right), \quad -1 \leq \phi \leq 1 \tag{2.43} \]

The quantity \( \phi \) is a weighting parameter that is taken from uni-directional concepts. For the viscous fluxes, a simple average of all the vertex polynomials (non-TVD) of a face are used for the evaluation of the viscous fluxes.

The Riemann problem is also solved approximately. If two semi-infinite states, \( U_L \) and \( U_R \), are separated by a membrane that vanishes at \( t = 0 \):

\[ Q(x,0) = Q_L (x < 0), \quad Q(x,0) = Q_R (x > 0), \tag{2.44} \]

the Riemann flux, \( f(Q_{\text{Riemann}}) \) at the \( t = 0 \) line can be expressed as:

\[ f(Q_{\text{Riemann}}) = f(Q_L) + \Delta f^-, \quad f(Q_{\text{Riemann}}) = f(Q_R) + \Delta f^+ \tag{2.45} \]

where \( \Delta f^+ \) is the change in fluxes across the right running characteristics and shock waves (positive eigenvalues) and \( \Delta f^- \) is the change in fluxes across the left running characteristics and shock waves (negative eigenvalues). These two equations can be combined into a single one:
\[ f(Q_{Riemann}) = \frac{f(Q_L) + f(Q_R)}{2} - \frac{\Delta f^+ - \Delta f^-}{2}. \] (2.46)

In Roe’s scheme [71], all the waves are treated as linear discontinuous waves. The total jump in fluxes across all the waves can be written as:

\[ f(Q_R) - f(Q_L) = (R \Lambda L) \Delta \Omega. \] (2.47)

Roe found that the previous equation holds if variables are averaged with a density weighted averaging:

\[ \rho_{Roe} = \sqrt{\rho_R \rho_L}, \]

\[ u_{Roe} = \frac{u_R \sqrt{\rho_R} + u_L \sqrt{\rho_L}}{\sqrt{\rho_R} + \sqrt{\rho_L}}, \quad v_{Roe} = \frac{v_R \sqrt{\rho_R} + v_L \sqrt{\rho_L}}{\sqrt{\rho_R} + \sqrt{\rho_L}}, \quad w_{Roe} = \frac{w_R \sqrt{\rho_R} + w_L \sqrt{\rho_L}}{\sqrt{\rho_R} + \sqrt{\rho_L}}, \] (2.48)

\[ h_{Roe} = \frac{h_R \sqrt{\rho_R} + h_L \sqrt{\rho_L}}{\sqrt{\rho_R} + \sqrt{\rho_L}}, \]

where \( h \) is the enthalpy of the gas. The speed of sound can be calculated as:

\[ c_{Roe} = \sqrt{\left\{ h_{Roe} - \frac{u_{Roe}^2 + v_{Roe}^2 + w_{Roe}^2}{2} \right\}(\gamma - 1)}. \] (2.49)

The jumps across the left running and right running characteristics can be written as:

\[ \Delta f^- = \sum_i r^i \lambda_i \alpha_i, \quad \Delta f^+ = \sum_i r^i \lambda_i \alpha_i \] (2.50)

where \( r^i \) is the right eigenvector corresponding to the positive (\( \lambda^+ \)) or negative (\( \lambda^- \)) eigenvalues, and \( \alpha_i \) is the product of the left eigenvector (\( l^i \)) and \( \Delta Q \). The eigenvalues for Navier-Stokes Equations with two turbulence equations are:
\( \lambda_1 = u - c \), for modified continuity equation,

\( \lambda_i = u \), (\( i = 2-4, 6,7 \)) for modified momentum and turbulence equations, \( \lambda_{e_s} = u + c \), for modified energy equation.

Combining equations (2.50) and (2.54), we obtain:

\[
f(Q^{Riemann}) = \frac{f(Q_L) + f(Q_R)}{2} - \sum_i r^i \lambda_i^+ \alpha_i - \sum_i r^i \lambda_i^- \alpha_i,
\]

or:

\[
f(Q^{Riemann}) = \frac{f(Q_L) + f(Q_R)}{2} - \sum_i r^i \lambda_i^+ l^i (Q_R - Q_L) - \sum_i r^i \lambda_i^- l^i (Q_R - Q_L).
\]

Once the solution of the Riemann problem is found, the inviscid flux at the integration node is known, thus, together with the viscous fluxes, the entire right hand side is known and can be updated using time integration scheme:

\[
\frac{Q^{n+1} - Q^n}{\Delta t} = RHS^{n+1}.
\]

RHS\(^{n+1}\) may be linearized about the current time level, resulting in a linear system of equations for \( Q \), or rather its increment:

\[
\frac{Q^{n+1} - Q^n}{\Delta t} = RHS^n + \left( \frac{\partial RHS}{\partial Q} \right)^n (Q^{n+1} - Q^n).
\]

The solution of this linear system results in a numerical scheme with much better stability properties than any explicit scheme. Theoretically, implicit schemes are stable for any time step size, although, non-linear effects often prevent the use of very large time steps, at least
in the early stages of a calculation. Once most transients have settled down, however, it is usually possible to increase the size of the time step significantly.

The next sections describe the boundary and initial conditions used in the numerical scheme for this study, details about the grid and parallel implementation of CFD++ code on multi-CPU platform.

2.3 Boundary Conditions

Subsonic, transonic and supersonic cases were run for this study. For incoming flow with Mach number $M = 0.3 – 0.9$, subsonic boundary conditions were applied, and for incoming flow with $M = 1.1 – 1.3$, supersonic boundary conditions were used. To assure proper boundary conditions at the limit ($M = 1$), the incoming flow was assumed to be $M = 1.01$ so that supersonic boundary conditions can be applied.

2.3.1 Subsonic Boundary Conditions

For both unsteady and quasi-time-average calculations, the inflow boundary conditions should maintain the mean aerodynamics of the flow field and minimize numerical reflections that inhibit convergence. Therefore, a subsonic reservoir boundary was used, in which the total temperature, total pressure and turbulence variables are imposed. The inflow velocity is directly extrapolated from the interior, while the inflow pressure and temperature are determined by assuming that the following isentropic relations hold across the inlet:

$$
\frac{T_0}{T} = 1 + \frac{\gamma - 1}{2} M^2, \quad \frac{p_0}{p} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}}.
$$
Turbulence kinetic energy and its dissipation were taken such that the turbulence intensity was 5%, then the value of $\varepsilon$ was obtained by adjusting the length scale until a $\mu_c/\mu$ ratio of 5-20 was achieved.

For solid walls, the no-slip boundary condition was applied. The wall was also assumed adiabatic and the turbulence variables set by: $k_w = 0$, $\varepsilon_w = 2v\left(\frac{\partial \sqrt{k}}{\partial n}\right)^2$.

Back pressure was imposed at the outflow boundary and all other quantities extrapolated from the interior.

A symmetry boundary condition was used at the upper plane, in which all scalar and vector quantities are reflected in the plane defined by the boundary. The flow velocity experiences a free slip at the boundary, becoming tangential to the surface on the boundary itself.

### 2.3.2 Supersonic Boundary Conditions

For supersonic cavity flow, all primitive variables ($p, \rho, u, v, w, k, \varepsilon$) were set at the inlet plane.

At the outflow and at the upper plane all quantities were extrapolated from the interior since no signals can travel back into the flow domain. The boundary values vector, $U_b$, was set equal to the corresponding cell centroid: $U_b = U_c$.

For solid walls, the no-slip boundary condition was again applied.

### 2.3.3 Boundary Conditions for Quasi-Time Average Computations

For the quasi-time-average computations, the boundary conditions for the inlet, outlet, upper plane and walls were the same as for the unsteady simulations. Since now calculations were performed without a cavity, the primitive variables at the cavity interface were
extracted from the time-averaged unsteady simulations at the cavity interface. These variables were then frozen at the interface and used as inflow/outflow conditions. This ensures that the correct velocities are obtained at the cavity interface even when the cavity is not physically present in the simulation.

### 2.4 Initial Conditions

Free stream variables for each length-to-depth ratio were set everywhere in the domain as the initial condition for the unsteady flow with $M = 0.3$ and $Re = 48163$. For each subsequent case of the same length-to-depth ratio, the previous solution was taken as the initial condition (i.e., the solution for $M = 0.3$, was taken as the initial condition for $M = 0.4$ and so on). This resulted in much faster cyclic convergence and savings in computation time.

The initial conditions for the quasi-time-average computations were free stream variables everywhere for all cases.

### 2.5 Grids

Several meshes were generated for the highest Reynolds number run with points clustered along the walls, such that the dimensionless distance from the wall to the first wall-adjacent control volume, $y^+$ was approximately 1. The number of grid points was varied with length-to-depth ratio so that similar local grid densities were achieved around walls. For $L/D = 1$ there were 292 x 72 grid points in the passage and 100 x 100 grid points in the cavity, while for $L/D = 2.75$, the mesh consisted of 410 x 72 grid points in the passage and 100 x 137 grid points in the cavity (see Figure 2.4). For lower Mach number cases, Reynolds number was also lower, which enabled
coarser grids. However, grids generated for the highest Mach number runs were used for all cases with one cavity geometry, so that the neural network training becomes possible.

A 2D mesh was extruded in the third dimension in order to implement translating wall boundary conditions (see Figure 2.5). The minimum number of cells (5) was used in $z$-direction.

Figure 2.4 Grids for $L/D = 1$ and $L/D = 2.75$. 
2.6 Parallel Implementation on Multi-CPU Platform

The CFD++ code can be used in parallel so the computational domain was decomposed into subdomains with the same number of grid cells using the MeTis algorithm [72]. This code performs decomposition with the minimum number of connections between subdomains resulting in minimal communication. Each subdomain is solved concurrently on an individual processor. Multiple CPUs communicate to maintain the integrity of the overall solution. A schematic representation of a parallel multiblock implementation is shown in Figure 2.6.

A Beowulf cluster consisting of five dual processor nodes with a total of 10 AMD Athlon 1600MP (1.4 GHz) processors was used for fluid flow calculations and neural network training. Since many cases were needed, they were run on two processors each, since using more processors increases CPU communication time and, therefore, decreases the efficiency of the computation. If a single case were needed, all available CPUs would be used to obtain the solution in the shortest possible time.
In the next chapter, lumped deterministic source terms are derived and the mathematical formulation illustrating how they can be used to obtain quasi-time-average solution is given.
3 Lumped Deterministic Source Term Approach

The following paragraphs illustrate the lumped deterministic source term concept, including how they are obtained and used to create quasi-time-average solutions. This is done by using the inviscid form of the 2D equations; viscous terms, the third dimension and the turbulence equations have been dropped for clarity, but are included in the actual equations used in this work.

The Lumped Deterministic Source Term methodology is obtained by first considering the unsteady governing equations:

\[
\frac{\partial p}{\partial t} + \frac{\partial}{\partial x} (\rho u) + \frac{\partial}{\partial y} (\rho v) = 0
\]

\[
\frac{\partial}{\partial t} (\rho u) + \frac{\partial}{\partial x} (\rho u^2 + p) + \frac{\partial}{\partial y} (\rho uv) = 0
\]

\[
\frac{\partial}{\partial t} (\rho v) + \frac{\partial}{\partial x} (\rho uv) + \frac{\partial}{\partial y} (\rho v^2 + p) = 0
\]

\[
\frac{\partial}{\partial t} \left( \rho e_i + \frac{\partial}{\partial x} \left[ (\rho e_i + p) u \right] + \frac{\partial}{\partial y} \left[ (\rho e_i + p) v \right] \right) = 0
\]

When the time derivatives are zero, the spatial derivatives represent the steady state governing equations. For numerical simulations these terms are often called the solution residual, \( R_k(Q) \) (here \( k = 1, 2, 3 \) or 4 ), as illustrated below:
\[ R_1(Q) = \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0 \]

\[ R_2(Q) = \frac{\partial}{\partial x}(\rho u^2 + p) + \frac{\partial}{\partial y}(\rho uv) = 0 \]

\[ R_3(Q) = \frac{\partial}{\partial x}(\rho uv) + \frac{\partial}{\partial y}(\rho v^2 + p) = 0 \]

\[ R_4(Q) = \frac{\partial}{\partial x}[(\rho e_i + p)u] + \frac{\partial}{\partial y}[(\rho e_i + p)v] = 0 \]

(3.2)

It is important to note that the conserved variable vector \( Q = [\rho, \rho u, \rho v, \rho e_i]^T \), which satisfies equation (3.2), is not the same as \( \overline{Q} \) obtained by time-averaging the unsteady solution (see Figure 3.1). This difference can be observed and the LDSTs defined by considering the vector residual operator, \( R_4(Q) \).

Figure 3.1 Instantaneous and Time-Averaged Density.

To understand this better start by considering the **conservative** variable vector \( Q \), which can be represented as:
\[ Q = \bar{Q} + Q', \quad (3.3) \]

where \( \bar{Q} \) is the time-averaged value of \( Q \) (to be distinguished from short-time Favre-averaged value \( \bar{Q} \)) over time \( \Delta T \), and \( Q' \) is an instantaneous fluctuation.

Another assumption is necessary to obtain the primitive variables from the time-averaged solution. If the primitive variables are calculated with the same relations that hold for unsteady variables from the time-averaged conservative variables, the quasi-time-averaged primitive vector is obtained:

\[
\bar{P} = \begin{bmatrix}
\frac{\gamma - 1}{\bar{\rho}} \left( \overline{\rho e_i} - \frac{1}{2\bar{\rho}} \left[ \overline{(\rho u)^2} + \overline{(\rho v)^2} + \overline{(\rho w)^2} \right] \right) \\
\frac{\gamma - 1}{\bar{\rho} R} \left( \overline{\rho e_i} - \frac{1}{2\bar{\rho}} \left[ \overline{(\rho u)^2} + \overline{(\rho v)^2} + \overline{(\rho w)^2} \right] \right) \\
\frac{\rho u}{\bar{\rho}} \\
\frac{\rho v}{\bar{\rho}}
\end{bmatrix}.
\quad (3.4)
\]

Unlike conservative variables, for which the time average of the fluctuation is zero \( \bar{Q}' = 0 \), the time average of primitive variable fluctuation vector is not identically zero: \( \bar{P}' \neq 0 \), since primitive variables are non-linear functions of conservative variables.

Decomposed conservative variables are then substituted into the unsteady governing equations. Since these equations are nonlinear, the resulting equation set will consist of three distinct sets of terms: terms containing only \( \bar{Q} \) products, terms containing only \( Q' \) products and terms containing a mixture of \( \bar{Q} \) and \( Q' \) variables. Similar to the Favre decomposition, upon time averaging, all the linear \( Q' \) terms and the unsteady terms vanish and one is left with the steady state residual operator acting on the time-averaged solution plus the time average of all
the higher order perturbation terms. It is then the higher order perturbation terms that define the
effect of unsteadiness in the steady state solution. They can be obtained from the equation:

\[ R_k(\overline{Q}) + LDST_k = 0. \]  \hspace{1cm} (3.5)

Again, note that the first term in equation (3.5) is the usual residual operator \( R_k(\overline{Q}) \), while the
second term represents the source terms that must be added to the steady state equations to
include the effect of unsteadiness, i.e., the lumped deterministic source terms. The two solutions,
one obtained from the steady-state equation (\( R_k(\overline{Q}) = 0 \)) and the solution of Equation (3.5) are
different, since the second represents the time average of the unsteady solution and includes the
effect of unsteadiness on the flow, while the first does not.

Interestingly, the continuity equation, since it is linear in the conservative variables,
reduces upon variable decomposition and time averaging to:

\[ \frac{\partial}{\partial x}(\overline{\rho u}) + \frac{\partial}{\partial y}(\overline{\rho v}) = 0. \]  \hspace{1cm} (3.6)

This implies that the mass source terms should be identically zero:

\[ LDST_1 = R_1(\overline{Q}) = 0. \]  \hspace{1cm} (3.7)

This is significant, since, if it holds for the discrete form of equation (3.7), the number of
required source terms is reduced and a smaller neural network is needed to approximate the
source terms. This idea was tested and the results showed that mass source terms do not affect
the quasi-time-average solution. Furthermore, error when zero mass terms were used was smaller
than when applying the source terms coming as the result of the discretization.

The momentum equation, upon decomposition and time averaging becomes:
\[
\frac{\partial}{\partial x}\left[ (\bar{\rho} + \rho^i) \frac{u + u^i}{2} + p + p^i \right] + \frac{\partial}{\partial y}\left[ (\bar{\rho} + \rho^i) \frac{v + v^i}{2} \right] = 0, \tag{3.8}
\]

or:
\[
\frac{\partial}{\partial x}\left[ \bar{\rho} u + \bar{\rho}^i u^i \right] + \frac{\partial}{\partial y}\left[ \bar{\rho} v + \bar{\rho}^i v^i \right] = 0. \tag{3.9}
\]

An important property of the velocity fluctuations is:
\[
\bar{\rho} u = (\bar{\rho} + \rho^i) u + (\bar{\rho} + \rho^i) u^i = \bar{\rho} u + (\bar{\rho} + \rho^i) u^i = \bar{\rho} u + (\bar{\rho} + \rho^i) u^i \Rightarrow (\bar{\rho} + \rho^i) u^i = 0 \tag{3.10}
\]

Using this property, the momentum equation reduces to:
\[
\frac{\partial}{\partial x}\left[ \bar{\rho} u + (\bar{\rho} + \rho^i) u^i \right] + \frac{\partial}{\partial y}\left[ \bar{\rho} v + (\bar{\rho} + \rho^i) v^i \right] = 0. \tag{3.11}
\]

In the momentum equation, the pressure is decomposed using the sum of its quasi-time-average value, obtained from time-averaged conservative variables and the unsteady difference: \( \bar{p} + p^i \).

Since pressure appears as a linear term here, it could be time-averaged, such that \( \bar{p} \) is present instead of \( \bar{p} + p^i \). However, to be consistent with the energy equation, the same procedure is used for all equations.

If all the quasi-time-average terms are separated on one side, and all terms that contain instantaneous fluctuations on the other, one obtains:
\[
\frac{\partial}{\partial x}\left( \bar{\rho} u^2 + \bar{p} \right) + \frac{\partial}{\partial y}\left( \bar{\rho} u v \right) = -\frac{\partial}{\partial x}\left[ (\bar{\rho} + \rho^i) u^i + \bar{p} \right] - \frac{\partial}{\partial y}\left[ (\bar{\rho} + \rho^i) v^i \right]. \tag{3.12}
\]
The LHS is again the residual operator $R_k(Q)$ for the steady state, while the RHS represents the lumped deterministic source terms that must be added to the steady state equations to obtain unsteady quasi-time-average solution:

$$LDST_2 = R_k(Q)$$  \hspace{1cm} (3.13)

The energy equation, upon decomposition and time averaging becomes:

$$\frac{\partial}{\partial x} \left\{ \rho \bar{e}_i + (\rho e_i)^{in} + p + p^{in} \left[ u + u^{in} \right] \right\} + \frac{\partial}{\partial y} \left\{ \rho \bar{e}_i + (\rho e_i)^{in} + p + p^{in} \left[ v + v^{in} \right] \right\} = 0$$  \hspace{1cm} (3.14)

Upon expanding, one obtains:

$$\frac{\partial}{\partial x} \left[ \rho \bar{e}_i u + (\rho e_i)^{in} u + pu + p^{in} u + \bar{\rho} e_i u^{in} + (\rho e_i)^{in} u^{in} + p u^{in} + p^{in} u^{in} \right] +$$

$$+ \frac{\partial}{\partial y} \left[ \rho \bar{e}_i v + (\rho e_i)^{in} v + pv + p^{in} v + \bar{\rho} e_i v^{in} + (\rho e_i)^{in} v^{in} + p v^{in} + p^{in} v^{in} \right] = 0$$  \hspace{1cm} (3.15)

which can be rearranged as:

$$\frac{\partial}{\partial x} \left[ \rho \bar{e}_i + p \bar{v} \right] + \frac{\partial}{\partial y} \left[ \rho \bar{e}_i + p \bar{v} \right] =$$

$$- \frac{\partial}{\partial x} \left[ \rho \bar{e}_i + (\rho e_i)^{in} u + pu^{in} + p^{in} u + p^{in} u^{in} \right] - \frac{\partial}{\partial y} \left[ \rho \bar{e}_i + (\rho e_i)^{in} v + pv^{in} + p^{in} v + p^{in} v^{in} \right]$$  \hspace{1cm} (3.16)

In the case of the energy equation, the LHS is again the residual operator $R_k(Q)$ for the steady state, while the RHS represents the lumped deterministic source terms that must be added to the steady state equations to obtain an unsteady quasi-time-averaged solution:

$$LDST_4 = R_k(Q)$$  \hspace{1cm} (3.17)

For completeness, it should be noted that for viscous flows, the source terms are again equal to the negative of the steady state residual.
To summarize, the lumped deterministic source terms are found by forming the time mean of the unsteady conservative variable vector and applying the residual operator to it (equations 3.7, 3.12 and 3.16). Using this technique, the unsteady effects and the influence of small geometric details can be included in simulations that do not compute these structures. However, a means to predict the source terms is needed to avoid repeated computation of the unsteady solution. Since source terms are highly nonlinear functions of the input parameters, linear interpolation was not an appropriate approach to their modeling, unless each parameter of the data set was divided in an enormous number of small increments. It was found that neural networks are more appropriate for this purpose if the source terms are reasonable functions of some governing parameters. The next chapter briefly describes the neural network approach used to model source terms.
4 Neural Networks

Although neural networks have been a research subject for the last fifty years, it is only recently that they are becoming popular for engineering applications. A brief history of neural networks is given at the beginning of this chapter, simply to give some feel for how knowledge in this field has progressed. Then, a mathematical discussion of neural networks is presented with the emphasis on how they can be used to model nonlinear relationships.

4.1 Brief History of Neural Networks

The modern era of neural networks began with the pioneering work of McCulloch and Pitts [73] in 1943. Their formal model of a neuron was assumed to follow an “all-or-none” law. With a sufficient number of binary threshold functions and synaptic connections, McCulloch and Pitts showed that a network could compute any computable function. Their work was followed by Donald Hebb [74], who introduced a physiological learning rule, which repeated activation of one neuron by another and across a particular synapse.

Gabor invented a “learning filter” that used gradient descent to obtain the optimal weights that will minimize the mean squared error between the predicted value and the observed value [75].
All the mentioned work was mostly theoretical. The first practical application of neural networks came with the invention of “perceptron,” and supervised learning rule by Rosenblatt [76].

Neural networks were first used to solve a major real-world problem by Widrow and Hoff [77], who introduced the least mean square (LMS) algorithm. The algorithm was used to formulate the Adaline – adaptive linear neuron, which is a simple network trained by a gradient descent rule. Widrow’s algorithm was limited and could not be used to train the more complex network. In their book, Minsky and Papert [78] used mathematics to demonstrate that there are fundamental limits on what single-layer neural networks can compute. They also stated that there was no reason to assume that multi-layer neural networks could overcome the limitations of single-layer neural nets. This resulted in a drastic reduction in funding and support for neural networks research. Many people believed that neural network research was dead.

During the 1980s, since new personal computers and workstations were developed and widely available, neural network research was reborn. Hopfield [79] used the idea of statistical mechanics to explain the operation of a certain class of recurrent networks with symmetric synaptic connections. This particular class of neural networks with feedback is today known as Hopfield networks. Rumelhart and McClelland [80] discovered the back-propagation algorithm for training multilayer perceptron networks.

In the last fifteen years, thousands of papers have been written, and neural networks have found many applications, not only in engineering, but also in medicine, business, finance and literature, which makes them especially attractive. A recent newspaper article stated, “The networks can be taught to recognize individual writing styles, and the researchers used it to compare works attributed to Shakespeare and his contemporaries.” Scientist at NASA Ames
Research Center and Boeing/McDonnell Douglas Aircraft Corporation are developing neural network software that will allow airplanes that suffer major equipment failures to land safely [81]. In its flight control application, the software takes data from the aircraft's air data sensors and compares the pattern of how the aircraft is actually flying with the pre-programmed pattern of how it should fly. If there is a mismatch, the aircraft's flight control computer will immediately “relearn” to fly the damaged plane with a new pattern. The software, which contains aeronautical stability and control equations, then determines the correct pattern that the aircraft should fly under the new conditions.

In the following section, the mathematical background of neural networks is given, along with the description of their concept and relevant definitions.

4.2 Architecture of Neural Networks

A neural network is a system that, due to its topological structure, can adaptively learn nonlinear mappings from input to output space when the network has a large database of prior examples from which to draw. It simulates human functions such as learning from experience, generalizing from previous to new data, and abstracting essential characteristics from inputs containing irrelevant data.

The basic architecture of a neural network consists of layers of interconnected processing units called neurons (comparable to the dendrites in the biological neuron), which transform an input vector \([p_1, p_2, \ldots, p_M]\) into an output vector \([a_1^n, a_2^n, \ldots, a_N^n]\).

Neurons without predecessors are called input neurons and constitute the input layer. All other neurons are called computational units. A nonempty subset of the computational units
is specified as the output units. All computational units, which are not output neurons, are called hidden neurons. Each interconnection between two neurons \( n_{ij}^{l-1} \rightarrow n_j^l \) has an associated weight factor \( w_{ij}^l \) and bias \( b_j^l \) that can be adjusted by using an appropriate learning algorithm like the Levenberg-Marquardt method (to be discussed later). The output of each neuron is:

\[
f^l \left( \sum_{j=1}^{s^{l-1}} w_{ij}^l a_j^{l-1} + b_j^l \right),
\]

where

\[
a_j^{l-1} = \begin{cases} 
  p_j & \text{if } l = 1 (j \text{ is an input unit}), \\
  f^{l-1} \left( \sum_{j=1}^n w_{ji}^{l-1} a_j^{l-2} + b_i^{l-1} \right) & \text{otherwise}
\end{cases}
\]

and \( f^{l-1} \) is the transfer function.

Neural networks are used for modeling complex data relationships. The Universal Approximation Theorem says that a neural network with one hidden layer is able to approximate any continuous function \( f: \mathbb{R}^M \rightarrow \mathbb{R}^S \) (\( M, S \) are dimensions of the function domain and range respectively) in any domain, with a given accuracy. Features of the input data are extracted in the hidden layer with a hyperbolic tangent transfer function, and in the output layer with a pure linear transfer function (Figure 4.1). Based on the theorem and the topological structure of the neural network one can generate complex data dependencies without performing time-consuming computations.

However, any neural network application depends on the training algorithm. In the current research, the Levenberg-Marquardt method was used to train the neural network. Next, the fundamentals of this backpropagation scheme are given.
4.3 Levenberg-Marquardt Method and Backpropagation

The learning algorithm is the repeated process of adjusting weights to minimize the network errors. These errors are defined by $e = t - a$, where $t$ is the desired network output vector, and $a = a(p, [w])$ is the actual network output vector. This weight adjustment is repeated for many training samples and is stopped when the errors reach a sufficiently low level.

The majority of neural network applications are based on the backpropagation algorithm. The term backpropagation refers to the process by which derivatives of the network error, with respect to network weights and biases, are calculated - from the last layer of the network to the first. The Levenberg-Marquardt backpropagation scheme [82-83] is a fast and
accurate method that was designed for least squares minimization. For a network with \( S^2 \) outputs and \( N \) pairs of input and desired output in the training data, the instantaneous error energy is:

\[
E(t - a) = \frac{1}{2} \sum_{r=1}^{N} [t - a]^T [t - a] = \frac{1}{2} \sum_{n=1}^{N} \sum_{s=1}^{S^2} [r(s) - a(s)]_n^2 = E([w])
\] (4.1)

At a global or local minimum of the error energy defined by equation (4.1), the derivatives with respect to the weights \([w]\) must be zero. Forming these derivatives for both layers \((l = 1 - 2)\) and all neurons in these layers \((i = 1, S^{l+1}; j = 1, S^{S^2+1})\), the following system of nonlinear equations is obtained:

\[
[f^l_{i,j}] = \sum_{r=1}^{N} \sum_{m=1}^{S^l} [t(m) - a(m)] \left[ - \frac{\partial a(m)}{\partial w^{l}_{i,j}} \right] = 0
\] (4.2)

where biases were considered as additional weights for each neuron.

The Levenberg-Marquardt algorithm is a process of iterative searching for the \([w]\) matrix, such that error energy function defined by equation (4.1) reaches a minimum.

To explain the method better, consider the gradient and Hessian of the error energy \( E([w]) \), which are defined respectively as the following:

\[
\nabla E([w]) = \left[ \frac{\partial E}{\partial w^{l}_{1,1}}, \frac{\partial E}{\partial w^{l}_{1,2}}, \ldots, \frac{\partial E}{\partial w^{l}_{S^{l+1},S^{S^2+1}}} \right]^T
\]

or

\[
\nabla E_i(w_{i,j}) = -\sum_{n=1}^{N} \sum_{s=1}^{S^2} [r(s) - a(s)]_n \left[ \frac{\partial a(s)}{\partial w^{l}_{i,j}} \right]_n.
\] (4.3)
The Hessian can be written as:

\[ [H] = \nabla^2 E([w]) = \begin{bmatrix}
\frac{\partial^2 E}{\partial w_{1,1}^j} & \cdots & \frac{\partial^2 E}{\partial w_{1,2}^j} \\
\frac{\partial^2 E}{\partial w_{1,1}^j} & \frac{\partial^2 E}{\partial w_{1,2}^j} & \cdots \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 E}{\partial w_{s+1,1}^j} & \cdots & \frac{\partial^2 E}{\partial w_{s+1,2}^j}
\end{bmatrix}
\]

or

\[ [H] = \begin{bmatrix}
\frac{\partial^2 E}{\partial w_{1,1}^j} & \frac{\partial^2 E}{\partial w_{1,2}^j} & \cdots & \frac{\partial^2 E}{\partial w_{s+1,2}^j} \\
\frac{\partial^2 E}{\partial w_{1,1}^j} & \frac{\partial^2 E}{\partial w_{1,2}^j} & \cdots & \frac{\partial^2 E}{\partial w_{s+1,2}^j} \\
\vdots & \ddots & \vdots & \vdots \\
\frac{\partial^2 E}{\partial w_{s,1}^j} & \cdots & \frac{\partial^2 E}{\partial w_{s+1,2}^j}
\end{bmatrix}
\]

(4.4)

\[ H_{\alpha,\beta} = \frac{\partial^2 E}{\partial w_{a,j}^j \partial w_{p,j}^j} = \frac{\partial (\nabla^2 E)}{\partial w_{p,j}^j} \approx \frac{\partial}{\partial w_{p,j}^j} \left( -\sum_{n=1}^{N} \sum_{s=1}^{S^2} \frac{\partial a(s)}{\partial w_{a,j}^j} J_{m} [t(s) - a(s)]_{n} \right) \]

(4.5)

The Hessian can be written as:

\[ [H] = [J]^T \cdot [J], \]

(4.5)

and the gradient as:

\[ \nabla E([w]) = [J]^T \cdot \hat{e}, \]

(4.6)

where \( \hat{e} \) is an error vector:

\[ \hat{e}_i = \sum_{n=1}^{N} \sum_{s=1}^{S^2} [t(s) - a(s)]_{n} \]

(4.7)

and \([J]\) is the Jacobian of vector \( \hat{e} \):
If we expand $E([w] + \Delta [w])$ into the Taylor series:

$$E([w] + \Delta [w]) = E([w]) + \nabla E([w]) \cdot \Delta [w] + \frac{1}{2} \Delta [w]^T \cdot [H] \cdot \Delta [w] + H.O.T., \quad (4.9)$$

and differentiate equation (4.9) with respect to $\Delta [w]$, we get:

$$\frac{\partial E([w] + \Delta [w])}{\partial (\Delta [w])} = \nabla E([w]) + [H] \cdot \Delta [w] \quad (4.10)$$

since we want to minimize $E([w] + \Delta [w])$, the first term is zero, so we finally get:

$$\Delta [w] = -[H]^{-1} \cdot \nabla E([w]) = -([J]^T \cdot [J])^{-1} \cdot [J]^T \cdot \bar{e} \quad (4.11)$$

One problem with this method is that the matrix $[H] = [J]^T \cdot [J]$ may not be invertible. This can be overcome by using the following modification to the Hessian matrix. If, instead of $[H]$, we use $[H] = [H] + \mu [I]$, which is positive semi-definite, and can be made positive definite by increasing $\mu > 0$, then the matrix will be invertible and $\Delta [w]$ will be the unique solution of equation (4.2) and:

$$[w]^{(k+1)} = [w]^{(k)} - [J]^T \cdot [J] + \mu^{(k)} [I]^{-1} \cdot [J]^T \cdot \bar{e} \quad (4.13)$$

This is known as the Levenberg-Marquardt Method. The scalar $\mu^{(k)}$ controls both the magnitude and iteration direction.
The algorithm is as follows:

1. Create an initial parameter vector, \([w]^{(0)}\), and an initial value \(\mu^{(0)}\).

2. Determine the search direction \([w]^{(k+1)}\) from equation (4.13).

3. If \(E([w]^{(k+1)}) < E([w]^{(k)})\), then \(\mu^{(k+1)}\) is reduced, otherwise it is increased. Here \(E([w])\) is the total error, given by Equation (4.1).

4. Check the stop criterion. If it is satisfied, then stop; else go to 2 (stop criterion: testing error has reached its minimum).

The backpropagation algorithm applies a correction, \(\Delta w_{i,j}'\), to the synaptic weight, \(w_{i,j}'\), connecting the output of neuron \(i\) to the input of neuron \(j\). Hence, we can iteratively update the weights and improve the performance of the network by applying new input-target pairs.

In this work, neural networks are utilized to generate source terms that are then used to include the effect of cavity unsteadiness in calculations of flow fields without the presence of a cavity. The next chapter describes the methodology of the coupled lumped deterministic source term – neural network approach.
5 Methodology

This chapter explains the analysis performed, the analytical tools applied to the computed data, the LDST-NN approach, how neural networks were applied to obtain the LDSTs and the quasi-time-average solution, and different approaches attempted to improve neural network efficiency and reduce the amount of training data.

5.1 Unsteady Data Analysis

This section presents the analytical tools used to post-process the unsteady results. The first tool used in this work was the sound pressure level in the cavity, used often to define the numerical accuracy of the scheme in comparisons to experimental cavity results. The second was the power spectrum of the unsteady solution, used to define the dominant oscillation frequency and the resultant cycle period for time averaging.

The correct prediction of Sound Pressure Level (SPL) is a critical measure of unsteady simulation accuracy. The SPL is a logarithmic scale measure of the pressure unsteadiness in which the minimum pressure fluctuation detected by the human ear ($2 \cdot 10^{-5} Pa$) is taken as the reference. For continuous pressure signals, the SPL is defined by:
\[
SPL = 10 \log \left( \frac{1}{T} \int_{t_1}^{t_2} (p - p_{\text{mean}})^2 \, dt \right) \left( \frac{2 \cdot 10^{-5} \, Pa}{Pa} \right)^2 [dB], \tag{5.1}
\]

where \( p_{\text{mean}} \) is a mean pressure:

\[
p_{\text{mean}} = \frac{1}{T} \int_{t_1}^{t_2} p \, dt. \tag{5.2}
\]

The appropriate summation replaces the integral for discrete data.

It should be noted that all variables are nondimensionalized in the CFD++ algorithm; hence, a value of the free stream pressure is required to compute SPL. In this work, 101,325 Pa was used.

SPL is presented to illustrate the accuracy of the computational scheme, but the goal of this work was to recreate the time average of an unsteady cavity solution. A standard Fast Fourier Transform (FFT):

\[
F(k) = \sum_{n=1}^{N} f(n) e^{-j2\pi(k-1)(\frac{n-1}{N})}, \quad k = 1, 2, \ldots, N \tag{5.3}
\]

was used to obtain this time average. First, the dominant frequency of the unsteady pressure signal is obtained and then, once the period of the cavity flow is known, the time-average of the unsteady solution is computed. The pressure signal had the largest variation at the middle of the cavity aft wall. Therefore, that location was selected to determine the dominant frequency of the flow and the period. The dominant frequencies were compared to frequencies obtained from the modified Rossiter equation (relation 1.3).
5.2 Cavity Drag Force

The last analysis tool used in this study was the comparison of the forces acting on a control volume sans the cavity, as defined from the control volume form of Newton’s Second Law:

\[ \vec{F} = \int_A \vec{V} \rho \vec{V} \cdot \hat{n} dA + \int_A p \hat{n} dA \]  \hspace{1cm} (5.4)

where \( \vec{F} \) is the force on the fluid volume, \( A \) is the control surface without the cavity, \( p \) is the pressure, \( \rho \) is the density, \( \vec{V} \) velocity vector, respectively and \( \hat{n} \) is the local unit normal. The integrals are replaced with appropriate summations for the discrete data and for the obtained LDST-NN solution, the variables in equation (5.4) were approximated with their quasi-time-average values. Figure 5.1 shows the areas along which cavity drag force was calculated. The force represents the net exchange of momentum due to unsteadiness and viscous shear and is meant to provide an accuracy assessment of the unsteady solution and a basis upon which the time averaged effect of unsteadiness can be evaluated.

\[ \vec{F} = \int_A \vec{V} \rho \vec{V} \cdot \hat{n} dA + \int_A p \hat{n} dA \]

**Figure 5.1** Areas Along Which Cavity Drag Force Was Calculated.

The next section describes the process of time-averaging the unsteady computation; the first step in obtaining the lumped deterministic source terms.
5.3 Time-Average of the Unsteady Solution

The time-average of the unsteady solution is obtained by averaging the unsteady conservative variables. Since CFD++ stores results as primitive variable vectors \((p, T, u, v, w, k, \varepsilon)\) the conservative variable vectors were created \((Q = Q(P))\), time-averaged:

\[
\overline{Q} = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} Q(t) dt
\]

and then converted back into primitive variable vectors (see equation 3.4). The period for the time averaging \((t_2 - t_1)\) can be obtained for all cases based on the FFT analysis of the pressure signal at the middle of the cavity aft wall. The time-average contains the unsteady effects, but they are not present in the steady-state solution. In the case of a cavity flow, the time-average solution has an altered boundary layer thickness which demonstrates how much the cavity affects the main-stream flow when compared with a steady flow sans cavity. It also includes effects of translating walls on the flow, and illustrates the position of the shock in the case of supersonic flow.

Once the time-averaged variables are known, they are inserted into the steady-state equations to obtain the source terms as discussed in Chapter 3. The following section describes the process of obtaining source terms and how they can be used to obtain quasi-time-average solution.

5.4 Lumped Deterministic Source Terms

Lumped deterministic source terms represent the difference between the time-averaged unsteady equations and the equivalent steady-state equations. They can be obtained by inserting the time-
average of the unsteady solution in the steady-state solver and then computing the right-hand-side of the steady-state equations after one iteration is performed. If source terms are added to the left-hand-side of the steady-state equations in each iteration, the quasi-time-average solution is obtained, which contains the time-average effects. However, finding source terms in this way for each case would be counter-productive since full unsteady simulations would be used to obtain a quasi-time-average solution in steady-state computations. Therefore, neural networks are utilized to generate source terms for any combination of input parameters based on the data in the training set. The following section gives the description of the LDST-NN approach.

## 5.5 LDST-NN Approach

The model architecture of the developed LDST-NN approach is given in Figure 5.2. It consists of the trained neural network, which gives source terms for any combination of input parameters. The source terms are then used in a steady state solver for the same input parameters (cavity geometry and boundary conditions) and uniform flow as the initial condition. The steady state

![Figure 5.2 LDST-NN Model Architecture.](image-url)
solver outputs the quasi-time-averaged solution, which includes flow unsteadiness effects, not found in the steady state solution.

The first and the longest phase in the LDST-NN approach is the preparation phase of the neural network. The schematic diagram of the preparation phase is shown in Figure 5.3. It starts with defining the input space. Depending on design requirements, the input space will include different parameters and their ranges. The neural network needs to be able to predict source terms within the space of input parameters for which the study is performed. Thus, the training set must contain information throughout the design space including the bounds of the design space. In this study, inflow Mach number, length-to-depth ratio and translational velocity of the cavity walls were used as the input parameters. For various combinations of these parameters source terms were found at each cell of the flow domain (various \( x \) and \( y \) locations) and those data were then used for neural network training.

The next step in neural network preparation is selecting critical points for training. The training data consists of input/output sets, so that the neural network can learn to predict outputs for a given set of inputs that are not in the training set. An important step is to determine a sufficient number of input/output sets and define which sets are required to successfully train the neural networks. A study was performed (to be discussed later) to determine the minimum amount of training data for each input parameter.

For cavity flow, source terms were obtained in the passage sans cavity. However, boundary conditions at the interface of the passage and the cavity are needed when the cavity is removed. Therefore, two neural networks were designed for modeling data dependencies. The number of source terms in the case of compressible 2D inviscid computations was 3 (correspond-
ing to the energy equation and 2 momentum equations), while for 3D inviscid computations there were 4 source terms in each cell of the domain (corresponding to the energy equation and all of the momentum equations). In the case of turbulent viscous computations, 2 additional source terms were generated, corresponding to the $k$ and $\varepsilon$ turbulence equations. The number of boundary conditions that were modeled at the cavity – passage interface was one larger than the number of source terms (dimension of the conservative variable vector). It was expected that the network for generating the source terms will require many more neurons and available memory than the network for generating boundary conditions and this was found to be the case.

The unsteady solution data, consisting of each individual source term at every $x,y$ coordinate for all the computed cases, was stored in a series of vectors. These vectors were then randomly reordered to achieve efficient training and then divided into training data (80 %) and testing data (20 %) [84]. The latter were used to determine the error when the network generates data it had never seen before. This error was used to set a stopping criterion, so that the network would not “overfit” data by continued training after the testing error reached its minimum. The training process was then stopped and the weights corresponding to the minimum testing error
were used to generate the source terms and the cavity-passage interface boundary conditions. The source term generation network used in this work had 50 neurons in the hidden layer, while the boundary condition generation network had 15 neurons in the hidden layer. This was found to be the optimum number for source term modeling (see Figures 5.3 and 5.4). In the case of various $L/D$ ratios and translational velocities of cavity walls, 30 neurons were used in the hidden layer of the source term generating network, which was the maximum that could be run on the available computer resources. Insufficient memory and CPU time did not allow training with the optimal number of neurons. However, this was not an issue, since the error obtained was within an acceptable range. The trained neural network generates source terms for any given $(M, L/D, \omega)$ combination in the training range. Source terms are then inserted in the steady state solver to capture the effect of cavity unsteadiness. The steady-state solver gives then quasi-time-average solution.

The following section gives details about the research performed to improve the efficiency of the neural network and reduce its training.

![Figure 5.4](image)  
**Figure 5.4** Number of Neurons vs. Mean Square Source Term Error for the Source Terms Generating Neural Network.
5.6 Accuracy and Training Improvements

Several experiments were performed to reduce the training set and improve the efficiency of the network. Although the attempt to reduce the training set was not successful, preprocessing of data helped to increase the accuracy of the network and improve its training.

5.6.1 Training Set Reduction

Training was successful for very large networks with many training data and a long training cycle. This was thought to be perhaps too time-consuming, so network optimization was studied with the goal of finding the minimal amount of training data required from the unsteady computations. Even though the computational database that was used for training was sparse, a study was conducted to see how much further the training set could be reduced and still allow the neural network to predict within the acceptable error. By reducing the training data set further, the required computational resources can be decreased, which in turn, increases the efficiency of the method. Several subsets of the computational training data were created by removing certain

---

**Figure 5.5** Number of Neurons vs. Mean Square Error for Boundary Conditions Generating Neural Network.
cases from the training set. Subsets were then used to determine points in the input space that could be removed from the training set without reducing the neural network efficiency. The accuracy of the neural network-source term approach was checked for all of the subsets, so that points that do not much affect the training process were determined. Even though a small number of such points were found, choosing \textit{a priori} the training set such that it contains the key points is problematic and at best only enabled a slight reduction of the computational data set, allowing a smaller number of preliminary computations. In the end, no workable approach for predicting these points was found and as such this idea was left in the future research.

5.6.2 Training Set Preprocessing

The network training set was preprocessed by normalizing the inputs and targets so that they fall in the interval \([-1,1]\); this produced the most efficient training. The generated output was then converted back from neural network to standard units.

The distribution of small-scale source terms (order less than \(10^{-10}\)) at places with unsteadiness not present and where it is logical to expect all source terms to be zero, obviously represents the noise in data set. These source terms were, therefore, removed from the training data by setting a threshold, under which the source terms were considered zero. This reduced the training time and increased the efficiency of the neural network because neurons were no longer used to reproduce this noise. The generated MATLAB code for neural network training in the case of various Mach number cases is given in the appendix.

Generated networks were able to give the solution of the cavity fluid flow for various \((M, L/D \text{ and } \omega)\) combinations in an order of milliseconds.
In the following chapter, the results obtained with unsteady computations and with LDST-NN approach were given with their analysis and the study of reducing data for neural network training.
6 Results

In this chapter, the feasibility of the LDST-NN approach is demonstrated by results presented in three parts: first for a single cavity geometry and various Mach numbers ($M = 0.3 – 1.3$), then for a spectrum of cavity length-to-depth ratios ($L/D = 1 – 2.75$) and Mach numbers ($M = 0.3 – 0.7$), and then for several translational velocities of the cavity walls ($v_{wall} = 0 – 288 \text{ m/s}$) and several Mach numbers of the incoming flow ($M = 0.3-0.8$). In each section, the results obtained from the unsteady solver and their analysis are given first, then the source terms extracted from the time average of the unsteady solution are presented, and finally results obtained with the LDST-NN approach are given and compared to the time average solution. In addition, the first section contains discussion of a training set reduction experiment that was ultimately unsuccessful.

6.1 Mach Number Variation

This section is organized such that the effect of unsteadiness on the time-averaged solution is presented first, then the source terms obtained from the time-averaged solution are given along with the corresponding source terms obtained from the trained neural network. The quasi-time-average results obtained with neural network derived source terms are also given. Quantitative measures of the LDST are shown as they vary with Mach number. Finally, the results conclude with examples depicting the accuracy of the neural network-based LDSTs. A qualitative comparison is made between the time averaged cavity result and the quasi-time-averaged LDST
results sans cavity. Quantitative measures of the neural network-based LDST are provided through presentations of the axial force error incurred by using the neural network-based LDST quasi-time-averaged solution and that obtained with the time average solution.

Eighteen cases were run with the unsteady solver for Mach numbers that varied from 0.3 to 2.0 to study the influence of the incoming flow Mach number on the cavity. Eleven of these cases were used to train a neural network to model source terms in the range of Mach numbers between 0.3 and 1.3 (Figure 6.1). The source terms for M > 1.3 were omitted because they had significant values around the shock and around the shock reflection location, which increased the error for the subsonic cases. A more general approach that covers the subsonic, transonic and supersonic regimes is possible with an improved source term implementation, but this was left for future work. Subsonic, transonic and supersonic cases were calculated for a single cavity geometry \((L = 2 \text{ cm}, D = 2 \text{ cm})\).

![Figure 6.1 Unsteady Cases Run To Train the Neural Network.](image)

As in all of the test cases, the unsteady simulations were run for 60 to 70 characteristic times to allow the solutions to become periodic. Convergence was tracked by checking the residuals (see Figure 6.2) and the unsteady pressure signal at the middle of the cavity aft wall (see Figure 6.3). The residuals were obtained as a summation of absolute values of each governing equation right-hand-side over all grid cells. In Figure 6.2, convergence was
achieved after approximately 40,000 iterations, after which the residuals became periodic. Two

![Figure 6.2 Residual Convergence History.](image)

![Figure 6.3 Unsteady Pressure Signal at the Middle of the Cavity Aft Wall for $M = 0.9$, $Re = 135240$.](image)
periods of the unsteady pressure signal at the middle of the cavity aft wall and its power spectrum are shown for one subsonic, one transonic and one supersonic case: \( M = 0.6, \ Re = 93869 \) in Figure 6.4, \( M = 0.9, \ Re = 135240 \) in Figure 6.5 and \( M = 1.2, Re = 171273 \) Figure 6.6. A fast Fourier Transform of the unsteady pressure signal at the middle of the cavity aft wall was performed and one dominant frequency was observed in the power spectrum, corresponding to the first (for \( M = 0.3 – 0.5 \)), second (\( M = 0.6 – 1.1 \)) or third Rossiter mode (\( M = 1.2 – 1.8 \)). Periods for all cases were obtained based on this analysis. Strouhal numbers were calculated
based on the dominant frequencies and also from the modified Rossiter equation (equation 1.3) and plotted in Figure 6.7. The discrepancies exist because values for constants $\alpha = 0.25$ and $\kappa = 0.57$ available in the literature were experimentally determined for a specific cavity length-to-depth ratio. It is interesting to note that at the points where mode shift occurs, there is no discrete

**Figure 6.6** Pressure Signal and Its Power Spectral Density for $M = 1.2$, $Re = 171273$.

**Figure 6.7** Strouhal Number vs. Mach Number, Corresponding to the Dominant Frequency, for Stationary Cavity.
change in the flow physics, but there is a change in the dominant frequency. The original dominant frequency reduces, as the secondary frequency becomes dominant. At these points more than one frequency was present in the spectrum; therefore the power spectrum has lower values, as illustrated in Figure 6.8. The period for the time average can be obtained for all cases based on the above. The observed mode shift was at first thought to be a challenge for assessing the accuracy of the proposed approach and how well the neural network can generate source terms to capture mode shift related phenomena. However, it was found that no abrupt changes were present in the time averaged solutions and hence no major problem appeared from mode shift. It is interesting to note that there may be no correlation between Strouhal number and the Sound Pressure Level.

![Figure 6.8](image)

**Figure 6.8** Pressure Signal and its Power Spectrum for \( M = 0.5, Re = 79039 \).

The unsteady vorticity contours and numerical Schlieren contours for subsonic cavity oscillations \( (M = 0.3) \) at different instances in time are shown in Figures 6.9 and 6.10, while vorticity contours and numerical Schlieren contours for supersonic cavity oscillations \( (M = 1.8) \) at different instances are shown in Figures 6.11 and 6.12. Schlieren images are obtained traditionally from experiment and represent the density gradient in the direction perpendicular to a knife-edge. Experimental Schlieren pictures thus have a preferred direction, i.e., they are either
Figure 6.9 Subsonic Vorticity Contours \((M = 0.3, Re = 48163)\) at Different Instances.
Figure 6.10 Subsonic Schlieren Contours ($M = 0.3, \text{Re} = 48163$) at Different Instances.
Figure 6.11 Supersonic Vorticity Contours ($M = 1.8, Re = 227123$) at Different Instances.
Figure 6.12 Supersonic Schlieren Contours ($M = 1.8, Re = 227123$) at Different Instances.
\( \partial \rho / \partial x \) or \( \partial \rho / \partial y \) depending on the orientation of the knife-edge. In the current research, numerical Schlieren images were simulated by obtaining the absolute value of both the \( x \) and \( y \) density gradients:

\[
|\Delta \rho| = \sqrt{\left( \frac{\partial \rho}{\partial x} \right)^2 + \left( \frac{\partial \rho}{\partial y} \right)^2}
\]

These simulated Schlieren images thus do not represent exactly the same thing as the Schlieren images obtained from experiment. However, these simulated Schlierens represent the true density gradient magnitude.

The important flow feature of the subsonic cases is vortex motion that dominates the flow physics. The shear layer that develops from the boundary layer floats over the cavity. This shear layer rolls into a vortex that detaches from the shear layer and moves toward the aft wall of the cavity. The shear layer impinges on the aft wall of the cavity and breaks into two, part of which enters the cavity and travels to the lower wall. The other half is washed into the passage. Although the formation and consequent shedding of the vortex can be observed in the Schlieren images, the oscillation cycle is better observed by following the behavior of the shed and resident vortices in the cavity.

In the case of supersonic flow, the important flow features are a moving compression wave at the leading edge of the cavity, a bow shock wave at the trailing edge of the cavity, an oscillating shear layer and vortex shedding. The wave patterns observed are closely associated with the vortex motion of the flow in the cavity. The appearance of these complicated features is believed to be associated with the amplification and damping of small disturbances in the unsteady shear layer. The oscillating shear layer causes a compression wave to be formed at the leading edge of the cavity. Shock waves form and recede as the shear layer oscillates. The shear
layer rolls up and forms a vortex that grows in strength as the fluid enters the cavity. The entering fluid causes a pressure wave to form inside the cavity below the vortex. The pressure wave reflects from the bottom aft corner and interacts with the vortex causing it to shed. The shed vortex then travels downstream and impinges on the aft lip of the cavity breaking into two. Unlike subsonic flow, the part of the vortex that enters the cavity disappears before reaching the lower wall. This oscillation cycle should be compared with that of Tam, et al. [27-28], although the current geometry is slightly different because of the upper wall symmetry plane.

The extremes of the unsteady pressure signal at the middle of the back wall are shown in Figure 6.13. The figure also gives mean pressure \( \bar{p} \) (as defined earlier in equation 5.2) and the quasi-time-average pressure, \( \bar{\bar{p}} \), obtained from the time-averaged conservative variables. As shown in the plot, \( \bar{\bar{p}} \) is a good approximate representation of the mean pressure (0.1- 0.2 % difference). It should be noted that in its present form the LDST-NN technique can only generate \( \bar{\bar{p}} \), not \( \bar{p} \), since the source terms are found from the time averaged conservative variables. However, the figure demonstrates that this is not a major issue for pressure.

![Figure 6.13](image-url)

**Figure 6.13** Pressure Extremes, Quasi-Time-Average Pressure and Pressure Mean.
Figure 6.14 shows the SPL variation at the middle of the cavity floor over a range of Mach numbers. This figure shows that SPL increases slightly with increasing Mach number until $M = 0.7$, it then increases steeply with Mach number due to transonic effects.

Figure 6.15 shows the drag force variation with Mach number incurred on the fluid volume described in Section 5.2. It represents a measure of drag on the fluid due to the cavity. As shown in the figure, drag force increases with Mach number up to $M = 0.8$. For transonic Mach numbers, drag force decreases slightly until $M = 1.2$ because of the shock reflection, present for transonic Mach numbers. This cavity drag force is meant to provide an accuracy assessment of the unsteady solution and will be compared to the quasi-time-average solution to demonstrate the accuracy of the approach and its ability to capture unsteady effects.

![SPL at the Middle of the Cavity Floor](image)

**Figure 6.14** Sound Pressure Level at the Middle of the Cavity Floor for Various Inlet Mach Numbers.

A qualitative example of the effect of the cavity on the main flow is shown by the time average contours of pressure, Mach number, Schlieren contours and z-direction vorticity. These contours are presented for one low-Mach number subsonic case - $M = 0.3$, $Re = 48163$ (Figure 6.16), one higher Mach number subsonic case - $M = 0.6$, $Re = 93869$ (Figure 6.17) and
for one supersonic case \(-M = 1.2, Re = 171273\) (Figure 6.18). The time average results show the
effect of the cavity on the flow. After the cavity, the boundary layer thickness increases
significantly, which is noticeable in the Mach number contours, Schlieren contours and \(z\)-
direction vorticity. Figure 6.19 shows that the vorticity intensifies as the Mach number increases.

**Figure 6.15** Axial Cavity Drag Force for Various Inlet Mach Numbers.

**Figure 6.16** Time Averaged Primitive Variables for \(M = 0.3, Re = 48163\); \(a\) Density, \(b\) Mach
Number, \(c\) Numerical Schlieren Contours, \(d\) Z-Direction Vorticity.
As shown in the figures, the boundary layer thickness increases with the increase of

Figure 6.17 Time Averaged Primitive Variables for $M = 0.6$, $Re = 93869$; $a$) Density, $b$) Mach Number, $c$) Numerical Schlieren Contours, $d$) Z-Direction Vorticity.

Figure 6.18 Time Averaged Primitive Variables for $M = 1.2$, $Re = 171273$; $a$) Density, $b$) Mach Number, $c$) Numerical Schlieren Contours, $d$) Z-Direction Vorticity.
Mach number in a similar way as the sound pressure level. The time averaged results and LDST reflect the variations in the unsteady behavior of the flow and present a good means for comparison.

Figure 6.19 Time Averaged Solution Vorticity Contours for $M = 0.3$-$0.8$.

Figure 6.20 shows boundary layer momentum thickness just after the cavity end as a function of Mach number. Figure 6.21 shows the LDSTs for a subsonic flow case ($M = 0.3$, $Re =$
Analytically, the continuity equation source terms are expected to be identically zero, since the continuity equation is linear with respect to the conservation variables $\rho$, $\rho u$ and $\rho v$. However, the numerical dissipation implicit in the CFD solution technique is non-linear; hence areas with high numerical dissipation produce non-zero continuity equation LDSTs. As expected, these occur in the areas of highest solution gradients and where the grid spacing is discontinuous. The continuity equation LDSTs were neglected in the neural network modeling because they represent a discretization error and were one order smaller than the other source terms. This removal was studied and it was found that it did not increase the error when compared to time-averaged unsteady results.

\[
\theta = \int \frac{\partial \theta}{\partial z} \left(1 - \frac{u}{U_u}\right) dy
\]

**Boundary Layer Momentum Thickness**

Figure 6.20 Boundary Layer Momentum Thickness as a function of Mach Number Just After the Cavity End.

It is interesting to note that apart from the discrepancies noted above, the source terms appear to be centered on the cavity shear layer and its impingement on the cavity aft wall. They also extend downstream from the cavity in the boundary layer, indicating that considerable
Figure 6.21 Source Terms for $M = 0.3$, $Re = 48163$; a) Energy, b) Mass, c) $X$-Momentum, d) $Y$-Momentum Source Terms.

Figure 6.22 Source Terms for $M = 1.2$, $Re = 171273$; a) Energy, b) Mass, c) $X$-Momentum, d) $Y$-Momentum Source Terms.
unsteadiness exists in this region because of the cavity as shown earlier by the vorticity contour time histories.

For supersonic flow, all source terms appear in a wide belt around the shock position and especially in the boundary layer around shock, meaning that a moving shock wave and the shock-boundary layer interaction are equally important unsteady phenomena in the supersonic

\[ M = 0.3 \]
\[ M = 0.4 \]
\[ M = 0.5 \]
\[ M = 0.6 \]
\[ M = 0.7 \]
\[ M = 0.8 \]

**Figure 6.23** Subsonic Energy Source Terms (\(M = 0.3\)-0.8).
cavity flow.

As evidenced by the earlier SPL variations with Mach number, it is expected that the source terms will also depend heavily on this parameter. Figures 6.23-6.25 show energy, \( X \)-momentum and \( Y \)-momentum source terms for various Mach numbers in the case of subsonic flows, while Figures 6.26-6.28 show the same source terms in the case of supersonic flows. Even

\[
\begin{align*}
M &= 0.3 \\
M &= 0.4 \\
M &= 0.5 \\
M &= 0.6 \\
M &= 0.7 \\
M &= 0.8
\end{align*}
\]

**Figure 6.24** Subsonic \( X \)-Momentum Source Terms \((M = 0.3-0.8)\).
though the neural network was trained for cases up to $M = 1.3$, higher Mach number cases were calculated to investigate the properties of source terms and their distribution. For supersonic flows, the angle of the source term stripe decreased as Mach number increased, but always followed the shock position.

Figure 6.25 Subsonic $Y$-Momentum Source Terms ($M = 0.3-0.8$).
Figure 6.26 Supersonic Energy Source Terms ($M = 1.3-2.0$).
Figure 6.27 Supersonic $X$-Momentum Source Terms ($M = 1.3-2.0$).
Figure 6.28 Supersonic Y-Momentum Source Terms ($M = 1.3-2.0$).
Source term dependence on Mach number is also shown in Figures 6.29 and 6.30, in which the source terms are plotted versus $y$ distance at the center of the cavity and downstream the cavity, at $x = 9$ for various incoming flow Mach numbers up to $M = 1.1$. In general, the greatest source term magnitude was in the shear layer; however, it is important to point out that the source terms are not limited to the cavity interface. Hence, a simulation must include the source terms in the field not simply on the boundary if the unsteady effects are to be effectively modeled. If only boundary conditions were used, without including source terms in a steady-state simulation, the results would not have the effect of the cavity (see Figure 6.29 and 6.30). In addition, the error in terms of the drag force for the steady-state computation without including the source terms is 96% compared to time-averaged unsteady results. Figures 6.31 and 6.32 show that the source term magnitude increases with Mach number in accordance with the similar SPL dependence. Based on these results, it follows that Mach number is an important parameter for source term modeling and also that the source terms correlate with the sound pressure level and the boundary layer momentum thickness.

The obtained training set was too large and required an extensive amount of unsteady computations. The following section gives details of the attempt to reduce the training set.

### 6.1.1 Reducing the Training Set

An experiment was performed to see if the amount of data obtained from unsteady computations could be reduced without increasing the resulting NN error. Nine additional neural network training cases were performed to determine if any points in the training set could be removed without significantly increasing the error. It was felt that if, so called, “critical points” were found to exist, a follow on study could be conducted to find ways to predict them a priori.
**Figure 6.29** Vorticity Contours Obtained: a) by Time-Averaging unsteady results, b) From a Steady-State Computation Without Including Source Terms and c) From a Steady-State Computation with Source Terms Included.

**Figure 6.30** Mach Number Contours Obtained: a) by Time-Averaging unsteady results, b) From a Steady-State Computation Without Including Source Terms and c) From a Steady-State Computation with Source Terms Included.
Otherwise this knowledge could not be successfully applied. In each case one point was omitted from the training set, with the exception of the limiting points \((M = 0.3 \text{ and } M = 1.3)\). These points were always kept to insure that the range remained the same (see Figure 6.33). Again, the testing error was determined on data the network “had never seen” while the testing data consisted of data from all cases, since the goal was to train a neural network that was efficient throughout the whole range of input parameters. Figure 6.34 shows the instantaneous error (calculated from equation 4.1) for the training cases illustrated in Figure 6.35, compared to the baseline training case that utilized all computed data. As shown in the figure, omitting cases from the training set always produced an increase in the testing error. However, this increase was insignificant for the cases in which \(M = 0.7 \text{ or } M = 0.9\) and therefore these cases were omitted from the training data. Because of this, an additional training set was created with both \(M = 0.7\)

![Energy Source Terms](image1)

![X-Momentum Source Terms](image2)

![Y-Momentum Source Terms](image3)

**Figure 6.31** Source Terms vs. \(y\) at the Cavity Centerline for Various Incoming Flow Mach Numbers.
**Figure 6.32** Source Terms vs. $y$ at $x = 9$ for Various Incoming Flow Mach Numbers.

**Figure 6.33** Additional Training Cases Performed with Points Omitted from the Training Set.
Figure 6.34 Testing Error for Additional Cases Comparing to Case with All Points Included in Training Set.

and $M = 0.9$ cases removed. As shown in Figure 6.34, this produced approximately double the error as compared to the baseline case, but still considerable less than cases in which $M = 0.6$ and 0.8 were omitted. Based on the above, it appears the number of cases run with the unsteady solver was properly selected. However, no discernable pattern appeared which could lead to a case elimination criterion; hence it is clear that some reduction in test cases may be possible if additional error can be tolerated. Equal Mach number increments might still be appropriate with larger increments and hence fewer computations. This study should be performed in future research.
6.1.2 Neural Network Training

The training time with the chosen training set was approximately 15 hours for each source terms on an ORIGIN 2000 platform. Figure 6.35 shows the neural network generated source terms for $M = 0.3$, $Re = 48163$ flow. When compared to Figure 6.21, the $x$-momentum and $y$-momentum source terms are modeled quite satisfactory, at least in a qualitative sense, while the energy source terms contain some small discrepancies. As stated earlier, the mass source terms were not modeled with the neural network, but were set to zero for the steady-state simulations. If the neural network generated source terms are included in steady state equations, as described in Chapter 5, a quasi-time-average solution is obtained. For this case, one thousand iterations were sufficient to achieve convergence.

![Figure 6.35 Neural Network Generated Source Terms for $M = 0.3$, $Re = 48163$; a) Energy, b) $X$-Momentum, c) $Y$-Momentum Source Terms.](image)

Figure 6.36 compares the results obtained by (a) time averaging the unsteady solution and (b) from the quasi-time-average solution obtained with the LDST-NN approach. The right-
hand images were computed as a steady-state flat-plate problem with the NN generated LDSTs added to the governing equations. As shown in the figure, the quasi-time-average solution captures cavity unsteadiness effects in the flow. Figure 6.37 gives axial cavity drag force as a function of incoming flow Mach number obtained by time-averaging the unsteady solution and from steady state calculations with and without NN generated source terms. If source terms are not included in the steady-state computations, no unsteadiness is present and the effects of the cavity are not obtained (the lowest data in Figure 6.37). A qualitative comparison of the axial cavity drag force shows that the results are nearly identical. To quantitatively assess the results of the LDST-NN approach, cavity drag force was calculated using formula (5.4) and compared to the drag force calculated on the time averaged solution. The comparison is given in Figures 6.37 and 6.38. As shown in the figure, the error between the time-mean solution and the quasi-time average solution increases with Mach number, but is between 1.3% (for $M = 0.3$) and 14.4% ($M = 1.15$). When steady state computations were performed without the LDSTs, the computed cavity drag force contains between 40.7% and 54.4% error when compared to the time-mean solution. The quasi-time-average solution predicts the flow better for lower Mach numbers, while for transonic and supersonic flows, the error is more significant due to the increased source term extent associated with the appearance of moving shock waves in the field. In any event, the quasi-time-average solutions always produced much lower error than the corresponding steady-state solution while capturing global unsteady effects like the change of boundary layer thickness due to the cavity.
Figure 6.36 Time Average Solutions Obtained from Time Mean Calculation (left) and Quasi-Time-Average Solution (right) for $M = 0.3$, $Re = 48163$; a) Density, b) Mach Number, c) Numerical Schlieren Contours, d) Z-Direction Vorticity.
Figure 6.37 Axial Cavity Drag Force vs. Incoming Flow Mach Numbers Obtained from Time Average Calculation and Steady State Calculation with and without NN Generated Source Terms. Solid Lines Represent the Time Average of the Unsteady Solution and the Colors Correspond to the Steady State with NN Generated S.T.s.

Figure 6.38 Error in Cavity Drag Force for LDST-NN and Steady State Solution.

In this section a series of calculations were performed to assess the effect of inlet Mach number on cavity flow. The unsteady simulations were used to obtain the lumped deterministic source terms that were then used to train a neural network. The source terms obtained with this technique were used to represent the effect of cavity unsteadiness in the quasi-
time-average solution that neglects the cavity. The drag force obtained with LDST-NN technique for cases not used in training was found to have the error between 1.3 and 14.4% compared to fully unsteady computations.

In the next section various cases were calculated to assess the effect of inlet Mach number and cavity length-to-depth ratio on cavity flow.

### 6.2 Length-to-Depth Variation

In this section, the neural network-based LDST prediction technique was extended to handle variable cavity length-to-depth ratios. The network was trained by considering test cases in which the cavity length, $L$, was varied from 2 to 5.5, and the cavity depth, $D$, was held constant to achieve length-to-depth ratios, $L/D$, from 1 to 2.75. Mach number was limited for these runs to between 0.3 and 0.7 (see Figure 6.39) because of the increased error experienced with transonic and supersonic flows as described in the previous section. The trained neural network was able to produce LDSTs, and therefore the quasi-time average solution, for any combination of input parameters in that range.

Once again, in this section the time-average of the unsteady solution is presented first, followed by the source terms extracted from the time-average, and finally the LDST-NN quasi-time-average solution are given.

A frequency analysis was again performed and three modes were determined to exist across the length-to-depth ratios (see Figure 6.40). Even though Rossiter’s formula (equation 1.3) states that oscillation frequencies are independent of $L/D$ ratio, it was found that Strouhal number increases slightly with increasing length-to-depth ratio.
**Figure 6.39** Cases Ran with Unsteady Solver for Various Length-To-Depth Ratios.

There is a graph showing Computed Data for $M_{\infty}$ ranges from 0.3 to 0.7 and $L/D$ ranges from 1.0 to 2.75. The data points are plotted to show the trend of $M_{\infty}$ vs. $L/D$.

**Figure 6.40** Strouhal Number vs. Length-to-Depth ratio, Corresponding to the Dominant Frequency, for Stationary Cavity, $M = 0.3$.

![Strouhal Number Graph](image)

A qualitative depiction of $L/D$ variation effects is shown through a series of time-averaged vorticity contours (Figure 6.41). It can be seen that as the length-to-depth ratio increases, so too does the influence of the cavity on the main flow. A general thickening of the
boundary layer is observed with increased $L/D$, anticipating the results of the drag force calculation.

Figure 6.42 shows boundary layer momentum thickness just after the cavity aft end as a function of length-to-depth ratio. As shown in the Figure, boundary layer momentum thickness changes with $L/D$ ratio in a similar way to the sound pressure level shown in Figure 6.43.

Quantitative measures of cavity unsteadiness effects are again presented in the form of SPL and cavity drag force vs. $L/D$. Figure 6.43 shows the variation of SPL with $L/D$ for the $M = 0.3$ runs. The SPL variation with $L/D$ cannot be as easily described but reaches a maximum over this range at $L/D = 2.25$, due to mode shift phenomena that affect the unsteady flow results. Results for other $L/D$ ratios are similar, and are not presented, since the goal of this section is to obtain source term dependence on $L/D$ ratio. The SPL variation with Mach number was already presented for one cavity in the previous section.

Figures 6.44 and 6.45 show the drag force variation with Mach number and $L/D$. As shown in the figures, drag force increases with both Mach number and length-to-depth ratio, as expected. The cavity drag force is again meant to compare the quasi-time-average results with the time-mean solutions.

As evidenced by the earlier SPL variations with $L/D$, it is expected that the source terms will also depend heavily on this parameter. Figures 6.46-6.48 give the energy, $X$-momentum and $Y$-momentum source term dependence on $L/D$ for $M = 0.3$. It is of particular interest to note that the energy and $Y$ Momentum source terms increase considerably for the $L/D = 2.25$ case, again corroborating the SPL result. Figure 6.49 also gives an idea of the source term dependence on length-to-depth ratio, where the source terms are plotted at the center of the cavity versus $y$ distance for various $L/D$ test cases ($y = 0$ is the plane of the main flow – cavity
Figure 6.41 Time Averaged Solution Vorticity Contours, $M = 0.3$ and $L/D = 1-2.75$. 
Figure 6.42 Boundary Layer Momentum Thickness as a Function of $L/D$ Ratio for $M = 0.3$.

Figure 6.43 SPL at the aft wall as a Function of $L/D$ Ratio for $M = 0.3$. 
**Figure 6.44** Cavity Drag Force as a Function of Mach number.

**Figure 6.45** Cavity Drag Force as a Function of $L/D$ ratio.
Figure 6.46 Energy Source Terms for Various Length-to-Depth Ratios ($M = 0.3$).
Figure 6.47 $X$-Momentum Source Terms for Various Length-to-Depth Ratios ($M = 0.3$).
Figure 6.48 $Y$-Momentum Source Terms for Various Length-to-Depth Ratios ($M = 0.3$).
interface). The center of each of those cavities are at different axial locations, but since the source terms span throughout the flow in a similar way for each cavity, only the center of the cavity is a place for suitable comparison of source terms magnitude. The source terms are still largest in the shear layer, but are not limited to the cavity interface. The figure indicates that their magnitude directly correlates to the SPL. Based on these results, it follows that length-to-depth ratio, in addition to Mach number, is an important parameter for source term modeling.

Next, we move to the NN modeling of these flows.

6.2.1 Neural Network Training

In this section, the results obtained with the neural network-based LDSTs are explored. The results are qualitatively compared to those obtained by the fully unsteady computations. Figure 6.50 shows a comparison of the drag force on the cavity for several Mach number and $L/D$ combinations to quantitatively assess the accuracy of the technique. The results obtained with the quasi-time averaged neural network-based LDST model are compared numerically to those obtained with the fully unsteady calculations in Table 6.1.

These results indicate that the error between the time average and quasi-time averaged LDST computations is between 4.5 and 14% for the cases considered. This is similar to the error obtained in the previous section, when the network had only one parameter (Mach number), but its range was wider ($M = 0.3 – 1.3$). The error increases with both Mach number and length-to-depth ratio. If the steady state computations are performed, the error is on the order of 40 to 153% (the lowest curve in Figure 6.50), assuming a no-slip boundary at the cavity/flow path interface. This coupled with the two order of magnitude reduction in computer time inherent
in the neural network-based LDST approach makes this technique a compelling alternative to fully unsteady calculations for flows that can be easily parameterized.

Figure 6.49 Source Terms at the Center of the Cavity as a Function of Length-to-Depth ratio ($M = 0.3$), a) Energy, b) $X$-Momentum, c) $Y$-Momentum.

Figure 6.50 Axial Cavity Drag Force vs. Mach Number for Various Incoming Flow Mach Numbers Obtained from the Time Average and the Steady State Calculations with and without NN Generated Source Terms. Solid Lines Represent the Time Average of the Unsteady Solution and the Colors Correspond to the Steady State with NN Generated S.T.s.
In this section a series of calculations were performed to assess the effects of inlet Mach number and cavity length-to-depth ratio. The unsteady simulations were used again to obtain the lumped deterministic source terms that were then used to train a neural network. The drag force obtained with LDST-NN technique for cases not used in training was found to have error between 4.5 and 14% compared to fully unsteady computations.

The next parameter considered in this study was the translational velocity of the cavity walls. For such cases 3D unsteady computations are necessary to obtain the effects of cavities on the main flow and therefore the quasi-time-average solution represents a significant saving.
6.3 Translational Velocity Variation for Cavities with Translating Walls

This section presents the results from the unsteady computations and their analysis, the time-averaged results and the quasi-time-averaged results obtained from the LDST-NN approach for cavity flows with translating aft walls.

Data were computed with the unsteady solver for various velocities of the inlet flow and wall translation velocities. The graph shown in Figure 6.51 lists the cases used to train the neural network.

![Graph showing cases used to train the neural network for rotational cavities.](image)

**Figure 6.51** Cases used to Train the Neural Network for Rotational Cavities.

Again, a dominant frequency was observed in the power spectrum corresponding to the first mode for $M = 0.3$ to $0.5$ and to the second mode for $M = 0.6$ to $0.8$ (Figures 6.52-6.54). The computations exhibit a mode shift between $M = 0.5$ and $M = 0.6$ (see Figure 6.55). When the detailed analysis is considered, two frequencies are noticeable in the FFT for the $M = 0.6$ case, with one clearly dominant. For cases with only one frequency, time averaging was performed on the appropriate period for that frequency. The period for the time average was again obtained for all cases based on the frequency spectrum analysis.
The extremes of the unsteady pressure signal at the middle of the back wall are shown in Figure 6.56. The figure gives mean pressure \( \bar{p} \) and the average pressure, \( \overline{p} \), obtained from the time-averaged conservative variables. As shown in the plot, \( \overline{p} \) is a good approximate representation of the mean pressure (max error is less than 2%). It should be noted that in its present form the LDST-NN technique can only generate \( \bar{p} \), not \( \overline{p} \). However, the figure demonstrates that this is not a major issue for computation of the pressure.

**Figure 6.52** Pressure Signal and Its Power Spectral Density for \( M = 0.3, \ Re = 48163, v_{wall} = 287.6 \text{ m/s} \).

**Figure 6.53** Pressure Signal and Its Power Spectral Density for \( M = 0.5, \ Re = 79039, v_{wall} = 287.6 \text{ m/s} \).
Figure 6.54: Pressure Signal and Its Power Spectral Density for $M = 0.6$, $Re = 93869$, $v_{wall} = 287.6 \, m/s$.

Figure 6.55: Strouhal Number vs. Mach Number, Corresponding to the Dominant Frequency, for Stationary Cavity.

Figures 6.57 and 6.58 show the SPL at the center of the cavity floor versus Mach number and translational velocity. The cavity wall velocity effect is very small but does increase slightly the SPL. However, the SPL increases significantly with increasing Mach number, especially for $M = 0.6-0.8$, perhaps related to the Rossier mode shift. Once again, it should be noted that mode shift affects only the unsteady results of the flow, not its time-average.
Figures 6.59 and 6.60 show axial cavity drag force (x-direction) as a function of Mach number and cavity wall velocity. Again, axial drag increases considerably with Mach number, but only marginally with wall translation. The situation is dramatically different in Figures 6.61 and 6.62, where the dependence of cavity z-direction drag force on Mach number is shown. As expected, z-direction force increases as either Mach number or cavity wall velocity increase, although the range of cavity drag force for different cavity wall velocities becomes slightly smaller as the Mach number increases. It should again be noted that cavity wall velocities affect significantly only the z-direction drag force. It is clear from the unsteady results that cavity wall velocity affects the global flow field justifying its introduction as a parameter. Furthermore, the SPL results suggest that the time averaged behavior is likely to change as Mach number increases because of the Rossiter mode shift. With this in mind, the next section explores the time averaged data.

![Pressure Extremes and its Quasi-Time Average](image)

**Figure 6.56** Pressure Extremes, Quasi-Time-Average Pressure and Pressure Mean.
The time averaged results are obtained by averaging the unsteady conservative variables. Figure 6.63 shows the time-average primitive variables: $z$-direction velocity, Mach number, Schlieren contours, and vorticity contours in $x$, $y$, and $z$ direction for $M = 0.3$.

**Figure 6.57** Sound Pressure Level at the Center of the Cavity Floor for Various Inlet Mach Numbers.

**Figure 6.58** Sound Pressure Level at the Center of the Cavity Floor for Various Cavity Wall Velocities.
Z-direction velocity contours show the upstream influence of $w$ velocity throughout the cavity, as indicated by the light blue “hook” in $w$ that extends from the aft to the forward wall.

Numerical Schlieren contours show that the cavity affects the flow by considerably increasing boundary layer thickness. Other effects of the translating walls produce nonzero $X$ and $Z$ direction vorticity in and downstream of the cavity. Again, the effect of the aft wall is felt in these variables both on the front wall and in the shear layer because of convection in the cavity. The cavity wall translation clearly affects the external flow in a significant way. The next step in the process is to understand how these changes manifest themselves in the source terms and complicate the neural network development.

![Axial Cavity Drag Force for Various Mach Numbers](image)

Figure 6.59 Axial Cavity Drag Force for Various Mach Numbers.
Figure 6.60 Axial Cavity Drag Force for Various Cavity Wall Velocities.

Figure 6.61 Cavity Drag Force in the Z Direction for Various Mach Numbers.
Once the time-averaged variables are known they are inserted into the steady-state equations to obtain the source terms, as shown in Figure 6.64. Negative $X$ and $Y$-momentum source terms above the cavity indicate that the flow is slowed down by the cavity. Positive $Z$-momentum source terms above and downstream of the cavity indicate the flow is speeded up by the translating walls. Source terms were extracted for vertical cuts at the middle of the cavity ($x = 6$) and downstream of the cavity ($x = 9$) to study the influence of various parameters on the source terms. Those source terms are shown as functions of cavity wall velocities in Figures 6.65 and 6.66 and also as functions of the Mach number in Figures 6.67 and 6.68. $X$-momentum source terms are virtually identical for all cavity wall velocities, since the wall translation does not greatly affect this component. $Y$ and $Z$-momentum source terms, as well as the energy source terms at the cavity centerline, become higher as the wall velocity increases. However, downstream of the cavity, the $X$-momentum source term dependence on the translating wall velocity is very noticeable, as all source terms increase with increasing cavity wall velocity.

Source term dependence on Mach number is slightly different. At the cavity centerline, $X$-momentum and $Y$-momentum source terms increase as Mach number increases,
while Energy and Z-momentum source terms just above the cavity decrease as Mach number increases. Downstream of the cavity, high Mach numbers cause large changes in the energy and Y-momentum source terms. All the above indicate that source terms are related to the level of unsteadiness in the fluid flow.

Figure 6.63 Time Averaged Primitive Variables for $M = 0.3$, $w_{wall} = 287.6$ m/s. a) Z-Direction Velocity, b) Mach Number, c) Numerical Schlieren Contours, d) X-Direction Vorticity, e) Y-Direction Vorticity, and f) Z-Direction Vorticity.
Figures 6.69 and 6.70 show cavity drag forces in the $X$ and $Z$ directions obtained with the LDST-NN approach and the corresponding forces obtained with steady state calculations sans source terms. As shown in the plots, the results obtained with the proposed method show good agreement with the time-averaged unsteady results. The maximum error obtained was 5% for axial drag force and 13% for the $Z$-direction force. The figure also shows points that were not included in the training set. (crosses in yellow boxes), for which quasi-time-average results are satisfactorily accurate. For steady state computations performed without source terms but with a no-slip boundary at the cavity/flow path interface (black points in the figure), the corresponding error is 52% for the axial drag force and 68% for the $Z$-direction force. This coupled with the two order of magnitude reduction in computer time inherent in the neural network-based LDST approach makes this technique a compelling alternative to fully unsteady calculations for flows that can be easily parameterized.

**Figure 6.64** Source Terms for $M = 0.3$, $w_{wall} = 286.7$ m/s a) Energy, b) $X$-Momentum, c) $Y$-Momentum and d) $Z$-Momentum Source Terms.
Figure 6.65 Source Terms vs. \( y \) at the Cavity Centerline for Various Translating Cavity Wall Velocities (\( M = 0.3 \)).

Figure 6.66 Source Terms vs. \( y \) at \( x = 9 \) for Various Translating Cavity Wall Velocities (\( M = 0.3 \)).
Figure 6.67 Source Terms vs. \( y \) at the Cavity Centerline for Various Mach Numbers \((w_{\text{wall}} = 286.7 \text{ m/s})\).

Figure 6.68 Source Terms vs. \( y \) at \( x = 9 \) for Various Mach Numbers \((w_{\text{wall}} = 286.7 \text{ m/s})\).
Figure 6.69 Axial Cavity Drag Force vs. Mach Number for Various Cavity Wall Velocities Obtained from the Time Average and the Steady State Calculations with and without Source Terms. Solid Lines Represent the Time Average of the Unsteady Solution and the Colors Correspond to the Steady State with NN Generated S.T.s.

Figure 6.70 Cavity Drag Force in the Z Direction vs. Mach Number for Rotational Cavity Obtained from the Time Average and the Steady State Calculations with and without Source Terms. Solid Lines Represent the Time Average of the Unsteady Solution and the Colors Correspond to the Steady State with NN Generated S.T.s.
In this section a series of calculations were performed to assess the effect of cavity translational velocity. The axial drag force obtained with LDST-NN technique for cases not used in training was found to have the error less than 5%, while the Z-direction force had the error of 13%. The axial force error is smaller than for cases with higher Mach numbers (Section 6.1) or for cases with length-to-depth ratio as an input parameter (Section 6.2). As expected, cavity translational velocity has less effect on the axial drag force than length-to-depth ratio or higher Mach numbers.

In the next chapter some overall conclusions from the research are given along with recommendations for future work.
7 Summary and Conclusions

In this final chapter, contributions made in the current research are summarized and the relevant conclusions from the research are drawn. Recommendations are made for additional research and improvements to the LDST-NN approach to increase its accuracy and the efficiency.

7.1 Cavity Flow

Unsteady cavity flows are currently too computationally intensive even on parallel platforms for realistic design purposes. In many cases, unsteady 3D computations are necessary, which require significant computational resources and CPU time.

In this study, a series of calculations were performed to assess the effect of cavity length-to-depth ratios, Mach number and cavity aft wall translational velocities variations.

Fast Fourier Transforms of the unsteady pressure signal at the middle of the cavity aft wall were performed and a dominant frequency was observed in the power spectrum, corresponding to either the first (for $M = 0.3 – 0.5$), second ($M = 0.6 – 1.1$) or third Rossiter mode ($M = 1.2 – 1.8$).

The unsteady results indicate that sound pressure level increases with both Mach number and cavity aft wall translational velocity. The SPL, however, varies in an uncorrelated way with $L/D$ but reaches a maximum for $L/D = 2.25$ values due to mode shift effects. The drag
induced by the cavity increases with Mach number and length-to-depth ratio almost linearly. Both axial cavity drag and z-direction drag force increase with cavity aft wall velocity.

Time averaged unsteady results indicated a general thickening of the cavity induced boundary layer with both increased Mach number and $L/D$. The flow physics governing the subsonic cavity is different from the flow physics of a supersonic cavity. Subsonic cavity oscillations are caused primarily by vortex shedding and the consequent break up of the vortices into two parts after hitting the back wall of the cavity. In the case of supersonic flow, the important flow features are a moving compression wave at the leading edge of the cavity, a bow shock wave at the trailing edge of the cavity, an oscillating shear layer and vortex shedding.

### 7.2 Lumped Deterministic Source Terms

The unsteady simulations were used to obtain lumped deterministic source terms that were then used to train a neural network to model source terms. The source terms obtained with this technique were used to represent the effect of cavity unsteadiness in quasi-time averaged solutions that neglect the cavity. The LDSTs were found to correlate directly with observed sound pressure level trends and provide an additional means of assessing unsteadiness. The LDSTs were found to reach a maximum near the cavity/main flow interface but also extended well into the field; indicating that boundary conditions alone would be inadequate for capturing unsteady effects.

Deterministic source terms can be computed from unsteady simulations and generated with a neural network for use in steady simulations sans cavity to capture the entire time average effect of the cavity. This was demonstrated for the entire range of Mach numbers, length-to-depth ratios and various translational velocities of the cavity wall.
Source terms can be used to understand the unsteady flow physics associated with cavity flows. They clearly indicate unsteady vortex and shock wave paths and their parametric variation offers insights into how cavity flow physics change with parametric changes. Source terms also extend downstream from the cavity in the boundary layer, indicating that considerable unsteadiness exists in this region because of the cavity effects.

7.3 LDST-NN Approach

Once the neural networks are trained to produce the source terms and boundary conditions at the cavity/main flow interface, one thousand steady-state iterations are sufficient to perform runs with different initial conditions and parameters. The modified neural network-based LDSTs were found to be capable of representing much of the time averaged Mach number, \( L/D \) variation and cavity wall translational velocity effects. Drag force results obtained with the technique for conditions not used to train the neural network were found to have error between 1.3 and 14.4% as compared to fully unsteady calculations for the range of incoming Mach numbers between 0.3 and 1.3. When length-to-depth ratio was varied, an additional parameter was added, and therefore the range for Mach numbers was decreased \((M = 0.3 – 0.8)\), which again resulted in a maximum error of 14%. The maximum error in the case of translational cavity walls was around 5% for axial cavity force and 13% for the \( Z \)-direction force. This should be compared to the 40 to 154% error obtained by neglecting the cavity completely for these same cases. These accuracies were obtained in two orders of magnitude less computational time than that required by the fully unsteady solutions. As such, the modified neural network-based LDST model is a viable tool for representing unsteady cavity effects.
The results of the study showed that modeling of flow over cavities is possible with steady simulations that include source terms, generated by a neural network. This method permits a considerable reduction of CPU time and is attractive for large scale simulations because it includes the effects of all the unsteady phenomena without computing the unsteady flow inside the cavity. The feasibility of the approach is demonstrated for Mach number, cavity length-to-depth ratio and cavity wall translational velocity variations.

Positive attributes of the Lumped Deterministic Source Term - Neural Network (LDST-NN) approach are:

1. relative ease of the approach, with which the effects of unsteadiness can be obtained without performing unsteady computations, except for the ones needed to generate the training set,

2. only limited unsteady computational data are required to train the neural network and get a solution that is accurate throughout the parameter space,

3. quasi-time-average solutions can be obtained from a steady state solver, and

4. the quasi-time-average solution can be obtained quickly, on the order of one thousand steady state iterations.

Negative sides of the approach are:

1. unsteady computations have to be performed a priori in order to train a neural network,

2. neural networks generally do not allow extrapolation of the data, i.e., neural networks can generate an accurate output vector only within the range of input vectors. The solution does not contain any information that is not present in the training data.
7.4 Lessons Learned

In this section the problems encountered and overcome during the course of the research will be explained.

At the beginning of the research, the developers from Metacomp Technologies incorporated the input/output option in their code for the lumped deterministic source terms. They thought that the magnitude of the source terms, not their sign, was the only important parameter. All the obtained source terms were then positive. After plugging them in the steady state solver, it was determined that the obtained results were not comparable to the time mean of an unsteady solution. Therefore, that option was changed, so that the both positive and negative source terms were enabled.

The solution file in CFD++ code is written in terms of primitive variables. The time-averaging however, should be performed on conservative variable vectors. Therefore it is necessary to transform the solution vector to a conservative form before time-averaging is performed. Failure to do so gives, again, wrong results.

The preliminary mesh was formed such that grid lines were concentrated along the walls, as explained in section 2.5 and equation 2.24. Discontinuities in the mesh spacing between zones did not affect the unsteady results and the boundary layer was fully captured. However, at those discontinuities, source terms existed, even though they were not expected, because of discretization error. Those source terms disappeared after the mesh was reconstructed. It seems that the mesh generation process is very important. The mesh not only needs to fully capture the boundary layer but must also be free of discontinuities.
The distribution of small-scale source terms (order less than $10^{-10}$) at places with no
unsteadiness present and where it is logical to expect all source terms to be zero, obviously
represents the noise in data set. These source terms were, therefore, removed from the training
data by setting a threshold, under which the source terms were considered zero. This reduced the
training time and increased the efficiency of the neural network because neurons were no longer
used to reproduce this noise.

### 7.5 Future Research

The time for this research was not unlimited, and therefore some issues had to be left for future
work. This section addresses such problems and work that could improve the LDST – NN
approach.

Based on the conclusions given in the previous section, it follows that the next step in
studying the LDST – NN approach would be to add additional parameters which affect the cavity
flow (such as boundary layer thickness at the cavity lip and pressure gradient), study the
dependence of source terms on these parameters and determine their distribution tendencies. In
addition, for each case, a very careful study of the input-output relationship should be performed
so that source terms can be modeled with less training data or with faster training, and eventually
without neural networks at all.

Rather than performing NN training for the whole domain of input parameters, which
requires a large number of neurons, the parameter space should be divided into subregions with
similar characterstics, so that smaller networks can learn their input-output relationships.
The model was used on identical grids or grids with similar local node densities. The method should be extended to determine how to use source terms on coarser or finer grids.

Unfortunately, the LDST-NN model cannot be currently used in other CFD codes, since they do not have the option to output lumped deterministic source terms, or to use them in steady state computations to obtain the quasi-time-average solution. This option is currently available only in Metacomp CFD++, which is a limiting factor for broad use of the approach in modeling fluid flow.

Finally, the model should be extended to other unsteady flows, like synthetic jets, purge cavities, etc. In all those cases, LDST-NN approach could allow obtaining time-average solution that includes unsteady effects without performing unsteady simulations.
Appendix: Matlab Code for Neural Network Training and Source Term Modeling

% Reading and Preprocessing of Source Terms
fid=fopen('nodesin.net','r');
fiktn1=fscanf(fid,'%d',[]);
fiktn2=fscanf(fid,'%d',[]);
fiktn3=fscanf(fid,'%d',[]);
for i=0:fiktn2-1
    a=fscanf(fid,'%d %f %f',[]);
    fiktn4=a(1);
    anode1(i)=a(2);
    anode2(i)=a(3);
end
clear a;

% Reading and Preprocessing of Cell Terms
fid=fopen('cellsin.net','r');
fiktc1=fscanf(fid,'%d',[]);
fiktc2=fscanf(fid,'%d',[]);
fiktc3=fscanf(fid,'%d',[]);
fid2=fopen('rhvecout-0.3.net','r');
fikt1=fscanf(fid2,'%d',[]);
fikt2=fscanf(fid2,'%d',[]);
fikt3=fscanf(fid2,'%d',[]);
fid3=fopen('fixed-0.3.net','w');
fprintf(fid3,'%d
',fikt1);
fprintf(fid3,'%d
',fikt2);
fprintf(fid3,'%d
',fikt3);
for i=0:fiktc2-1
    a=fscanf(fid,'%d %d %d %d %d',[]);
    fiktc4=a(1);
    fiktc5=a(2);
    fiktc6=a(3);
    fiktc7=a(2);
    fiktc8=a(3);
    b=fscanf(fid2,'%d %d %f %f %f %f %f %f %f',[]);
    fikt4=b(1);
    fikt5=b(2);
source1 = b(3);
source2 = b(4);
source3 = b(5);
source4 = b(6);
source5 = b(7);
source6 = b(8);
source7 = b(9);

x = (anode1(fikt5) + anode1(fikt6) + anode1(fikt7) + anode1(fikt8)) / 4
y = (anode2(fikt5) + anode2(fikt6) + anode2(fikt7) + anode2(fikt8)) / 4

if ((x < 0.5) || (x > 11.5) || (y > 2.5))
    source1 = 0.0
    source3 = 0.0
    source4 = 0.0
    source5 = 0.0
    source6 = 0.0
    source7 = 0.0
else
    if (abs(source1) < 1.0E-09)
        source1 = 0.0
    end
    if (abs(source3) < 1.0E-09)
        source3 = 0.0
    end
    if (abs(source4) < 1.0E-09)
        source4 = 0.0
    end
    if (abs(source5) < 1.0E-09)
        source5 = 0.0
    end
    if (abs(source6) < 1.0E-09)
        source6 = 0.0
    end
    if (abs(source7) < 1.0E-09)
        source7 = 0.0
    end
end

source2 = 0.0
fprintf(fid3, '
', fikt4, fikt5, source1, source2, source3, source4, source5, source6, source7);
end
d = fscanf(fid2, '
', [1 2]);
fikt7 = d(1);
fikt8 = d(2);
printf(fid3, '
', fikt7, fikt8);
close('all');
clear all;

% Read all Files with Computed Data
n1 = 10;
counter = 0;
a = zeros(n1, 231325);
fid=fopen('fixed-0.3.net','r');
b=fscanf(fid,'%f %f',[1 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-0.3.net'
st=fopen(fid);
 fid=fopen('fixed-0.4.net','r');
b=fscanf(fid,'%f %f',[1 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-0.4.net'
st=fopen(fid);
 fid=fopen('fixed-0.5.net','r');
b=fscanf(fid,'%f %f',[1 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-0.5.net'
st=fopen(fid);
 fid=fopen('fixed-0.6.net','r');
b=fscanf(fid,'%f %f',[1 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-0.6.net'
st=fopen(fid);
 fid=fopen('fixed-0.7.net','r');
b=fscanf(fid,'%f %f',[1 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-0.7.net'
st=fopen(fid);
 fid=fopen('fixed-0.8.net','r');
b=fscanf(fid,'%f %f',[1 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-0.8.net'

"165"
end
counter=counter+b(2);
'Read fixed-0.9.net'
st=fclose(fid);
 fid=fopen('fixed-1.0.net','r');
b=fscanf(fid,'%f %f',[l 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-1.0.net'
st=fclose(fid);
 fid=fopen('fixed-1.1.net','r');
b=fscanf(fid,'%f %f',[l 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-1.1.net'
st=fclose(fid);
 fid=fopen('fixed-1.2.net','r');
b=fscanf(fid,'%f %f',[l 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-1.2.net'
st=fclose(fid);
 fid=fopen('fixed-1.3.net','r');
b=fscanf(fid,'%f %f',[l 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
counter=counter+b(2);
'Read fixed-1.3.net'
st=fclose(fid);

% Randomly reorder source term data
 fid1=fopen('reordered.source1','w');
 fid3=fopen('reordered.source3','w');
 fid4=fopen('reordered.source4','w');
 fid6=fopen('reordered.source6','w');
 fid7=fopen('reordered.source7','w');

    fprintf(fid1,'%6.0f \n',counter);
    fprintf(fid3,'%6.0f \n',counter);
    fprintf(fid4,'%6.0f \n',counter);
    fprintf(fid6,'%6.0f \n',counter);
    fprintf(fid7,'%6.0f \n',counter);
for i=1:counter
    j=ceil(rand(1)*counter);

fprintf(fid1,'%12.7f %12.7f %12.7f
%16.8e
',a(1,j),a(2,j),a(3,j),a(4,j));
fprintf(fid3,'%12.7f %12.7f %12.7f
%16.8e
',a(1,j),a(2,j),a(3,j),a(6,j));
fprintf(fid4,'%12.7f %12.7f %12.7f
%16.8e
',a(1,j),a(2,j),a(3,j),a(7,j));
fprintf(fid6,'%12.7f %12.7f %12.7f
%16.8e
',a(1,j),a(2,j),a(3,j),a(9,j));
fprintf(fid7,'%12.7f %12.7f %12.7f
%16.8e
',a(1,j),a(2,j),a(3,j),a(10,j));
end
st=fclose('all');
clear all;

% Start NN training
% Mass source terms
% n1=number of variables (independent + dependent)
% n2=number of rows in a file
n1=4;
 fid=fopen('reordered.source1','r');
 n2=fscanf(fid,'%f',[1]);
a=zeros(4,n2);
p=zeros(3,n2);
t=zeros(1,n2);
a=fscanf(fid,'%f %f',[n1 n2]);
 st=fclose(fid);
for i=1:n2
    p(1,i)=a(1,i);
    p(2,i)=a(2,i);
    p(3,i)=a(3,i);
    t(i)=a(4,i);
end
clear a;

% Normalize data so that all variables are in [-1;1] range
[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
iival=ceil(3/4*n2)+1:n2;
iitr=1:ceil(3/4*n2);
v.P=pn(:,iival);
v.T=tn(:,iival);
ptr=pn(:,iitr);
ttr=tn(:,iitr);
net=newff([-1 1; -1 1; -1 1], [50,1], {'tansig','purelin'},'trainlm');
net.trainParam.show=10;
%net.trainParam.lr=0.005;
net.trainParam.mem_reduc=10;
net.trainParam.epochs=3000;
net=init(net);
%load parmreze1 W B L minp maxp mint maxt;
%net.IW=W;
%net.LW=L;
%net.b=B;
[net,tr]=train(net,ptr,ttr,[],[],v);
plot(tr.epoch,tr.perf,tr.epoch,tr.vperf,tr.epoch,tr.tperf)
legend('Training','Validation','Test',-1);
ylabel('Squared Error');
xlabel('Epoch');

W=net.IW;
B=net.b;
L=net.LW;
save parmreze1.mat W B L minp maxp mint maxt;
clear all;

% X-Momentum Source Terms
%n1=number of variables (independent + dependent)
%n2=number of rows in a file
n1=4;
fid=fopen('reordered.source3','r');
n2=fscanf(fid,'%f',[1]);
a=zeros(4,n2);
p=zeros(3,n2);
t=zeros(1,n2);
a=fscanf(fid,'%f %f',[n1 n2]);
st=fclose(fid);
for i=1:n2
    p(1,i)=a(1,i);
    p(2,i)=a(2,i);
    p(3,i)=a(3,i);
    t(i)=a(4,i);
end

clear a;

% Normalize data so that all variables are in [-1;1] range
[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
iival=ceil(3/4*n2)+1:n2;
iitr=1:ceil(3/4*n2);
v.P=pn(:,iival);
v.T=tn(:,iival);
ptr=pn(:,iitr);
ttr=tn(:,iitr);
net=newff([-1 1; -1 1; -1 1], [50,1], {'tansig','purelin'},'trainlm');
net.trainParam.show=10;
net.trainParam.lr=0.005;
net.trainParam.mem_reduc=10;
net.trainParam.epochs=3000;
net.trainParam.min_grad=1e-10;
net.trainParam.max_fail=10;
net=init(net);
load parmreze3 W B L minp maxp mint maxt;
%net.IW=W;
%net.LW=L;
%net.b=B;
[net,tr]=train(net,ptr,ttr,[],[],v);
% Y-Momentum Source Terms
% n1 = number of variables (independent + dependent)
% n2 = number of rows in a file
n1 = 4;
fid = fopen('reordered.source4', 'r');
n2 = fscanf(fid, '%f', [1]);
a = zeros(4, n2);
p = zeros(3, n2);
t = zeros(1, n2);
a = fscanf(fid, '%f %f', [n1 n2]);
st = fclose(fid);
for i = 1:n2
    p(1, i) = a(1, i);
    p(2, i) = a(2, i);
    p(3, i) = a(3, i);
    t(i) = a(4, i);
end

% Normalize data so that all variables are in [-1;1] range
[pn, minp, maxp, tn, mint, maxt] = premnmx(p, t);
iival = ceil(3/4*n2) + 1:n2;
iitr = 1:ceil(3/4*n2);
v.P = pn(:, iival);
v.T = tn(:, iival);
ptr = pn(:, iitr);
ttr = tn(:, iitr);
net = newff([-1 1; -1 1; -1 1], [50, 1], {'tansig', 'purelin'}, 'trainlm');
net.trainParam.show = 10;
% net.trainParam.lr = 0.005;
net.trainParam.mem_reduc = 10;
net.trainParam.epochs = 3000;
net.trainParam.min_grad = 1e-10;
% net.trainParam.max_fail = 10;
net = init(net);
% load parmreze3 W B L minp maxp mint maxt;
% net.IW = W;
% net.LW = L;
% net.b = B;
[net, tr] = train(net, ptr, ttr, [], [], v);
plot(tr.epoch, tr.perf, tr.epoch, tr.vperf, tr.epoch, tr.tperf)
legend('Training', 'Validation', 'Test', -1);
ylabel('Squared Error');
xlabel('Epoch');
W = net.IW;
B = net.b;
L = net.LW;
save parmreze3.mat W B L minp maxp mint maxt;
clear all;
ylabel('Squared Error');
xlabel('Epoch');
W=net.IW;
B=net.b;
L=net.LW;
save parmreze3.mat W B L minp maxp mint maxt;
clear all;

% Turb 1 Source Terms
%n1=number of variables (independent + dependent)
%n2=number of rows in a file
n1=4;
 fid=fopen('reordered.source6','r');
n2=fscanf(fid,'%f',[1]);
a=zeros(4,n2);
p=zeros(3,n2);
t=zeros(1,n2);
a=fscanf(fid,'%f %f',[n1 n2]);
st=fclose(fid);
for i=1:n2
    p(1,i)=a(1,i);
p(2,i)=a(2,i);
p(3,i)=a(3,i);
t(i)=a(4,i);
end
clear a;

% Normalize data so that all variables are in [-1;1] range
[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
iival=ceil(3/4*n2)+1:n2;
iitr=1:ceil(3/4*n2);
v.P=pn(:,iival);
v.T=tn(:,iival);
ptr=pn(:,iitr);
ttr=tn(:,iitr);
net=newff([-1 1; -1 1; -1 1], [50,1], {'tansig','purelin'},'trainlm');
net.trainParam.show=10;
net.trainParam.lr=0.005;
net.trainParam.mem_reduc=10;
net.trainParam.epochs=3000;
net.trainParam.min_grad=1e-10;
net=init(net);
load parmreze6 W B L minp maxp mint maxt;
%load parmreze3.mat W B L minp maxp mint maxt;
net.IW=W;
net.LW=L;
net.b=B;
[net,tr]=train(net,ptr,ttr,[],[],v);
plot(tr.epoch,tr.perf,tr.epoch,tr.vperf,tr.epoch,tr.tperf)
legend('Training','Validation','Test','-1');
ylabel('Squared Error');
xlabel('Epoch');
W=net.IW;
B=net.b;
L=net.LW;
save parmreze6.mat W B L minp maxp mint maxt;
clear all;

% Turb 2 Source Terms
%n1=number of variables (independent + dependent)
%n2=number of rows in a file
n1=4;
 fid=fopen('reordered.source7','r');
 n2=fscanf(fid,'%f',[1]);
 a=zeros(4,n2);
 p=zeros(3,n2);
 t=zeros(1,n2);
 a=fscanf(fid,'%f %f',[n1 n2]);
 st=fclose(fid);
 for i=1:n2
     p(1,i)=a(1,i);
     p(2,i)=a(2,i);
     p(3,i)=a(3,i);
     t(i)=a(4,i);
 end
 clear a;

% Normalize data so that all variables are in [-1;1] range
 [pn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
 iiival=ceil(3/4*n2)+1:n2;
 iitr=1:ceil(3/4*n2);
 v.P=pn(:,iiival);
 v.T=tn(:,iiival);
 ptr=pn(:,iitr);
 ttr=tn(:,iitr);
 net=newff([-1 1; -1 1; -1 1], [50,1], {'tansig','purelin'},'trainlm');
 net.trainParam.show=10;
 %net.trainParam.lr=0.005;
 net.trainParam.mem_reduc=10;
 net.trainParam.epochs=3000;
 net.trainParam.min_grad=1e-10;
 net=init(net);
 %load parmreze7 W B L minp maxp mint maxt;
 %net.IW=W;
 %net.LW=L;
 %net.b=B;
 [net,tr]=train(net,ptr,ttr,[],[],v);
 plot(tr.epoch,tr.perf,tr.epoch,tr.vperf,tr.epoch,tr.tperf)
 legend('Training','Validation','Test',-1);
 ylabel('Squared Error');
 xlabel('Epoch');
 W=net.IW;
 B=net.b;
 L=net.LW;
save parmreze7.mat W B L minp maxp mint maxt;
clear all;

% Test the trained Neural Network for M = 0.45

load parmreze1.mat W B L minp maxp mint maxt;
nl=10;
counter=0;
 fid=fopen('fixed.net','r');
b=fscanf(fid,'%f %f',[1 3]);
 for i=counter+1:counter+b(2)
   a(:,i)=fscanf(fid,'%f %f',[n1 1]);
 end
st=fclose(fid);
% Read one fixed file for geometry information
'REad fixed-0.3.net'
 for i=1:b(2)
   p(1,i)=a(1,i);
   p(2,i)=a(2,i);
   p(3,i)=0.45;
 end
 clear a;
 [pn]=tramnmnx(p,minp,maxp);
 net=newff([-1 1; -1 1; -1 1], [50,1], {'tansig','purelin'},'trainlm');
 net=init(net);
 net.IW=W;
 net.b=B;
 net.LW=L;
 ako=sim(net,pn);
 [treba]=postmnmx(ako,mint,maxt);
 fid=fopen('testir1.dat','w');
 for i=1:b(2)
   fprintf(fid,'%12.7f %12.7f %15.7e
',p(1,i),p(2,i),p(3,i),treba(i));
 end
st=fclose(fid);
 clear all;

load parmreze3.mat W B L minp maxp mint maxt;
nl=10;
counter=0;
 fid=fopen('fixed.net','r');
b=fscanf(fid,'%f %f',[1 3]);
 for i=counter+1:counter+b(2)
(674,480),(908,521)
 end
st=fclose(fid);
% Read one fixed file for geometry information
'REad fixed-0.3.net'
 for i=1:b(2)
   p(1,i)=a(1,i);
   p(2,i)=a(2,i);
   p(3,i)=0.45;
clear a;
[pn]=tramnmx(p,minp,maxp);
net=newff([-1 1; -1 1; -1 1], [50,1], {'tansig','purelin'},'trainlm');
net=init(net);
net.IW=W;
net.b=B;
net.LW=L;
ako=sim(net,pn);
[treba]=postmnmx(ako,mint,maxt);
end

fid=fopen('testir3.dat','w');
for i=1:b(2)
    fprintf(fid,'%12.7f %12.7f
%15.7e\n',p(1,i),p(2,i),p(3,i),treba(i));
end
st=fclose(fid);
clear all;

load parmreze4.mat W B L minp maxp mint maxt;
n1=10;
counter=0;
 fid=fopen('fixed.net','r');
b=fscanf(fid,'%f %f',[1 3]);
for i=counter+1:counter+b(2)
    a(:,i)=fscanf(fid,'%f %f',[n1 1]);
end
st=fclose(fid);
% Read one fixed file for geometry information
'Read fixed-0.3.net' for i=1:b(2)
    p(1,i)=a(1,i);
    p(2,i)=a(2,i);
    p(3,i)=0.45;
end
clear a;
[pn]=tramnmx(p,minp,maxp);
net=newff([-1 1; -1 1; -1 1], [50,1], {'tansig','purelin'},'trainlm');
net=init(net);
net.IW=W;
net.b=B;
net.LW=L;
ako=sim(net,pn);
[treba]=postmnmx(ako,mint,maxt);
 fid=fopen('testir4.dat','w');
for i=1:b(2)
    fprintf(fid,'%12.7f %12.7f
%15.7e\n',p(1,i),p(2,i),p(3,i),treba(i));
end
st=fclose(fid);
clear all;

load parmreze6.mat W B L minp maxp mint maxt;
n1=10;
counter=0;
 fid=fopen('fixed.net','r');
b=fscanf(fid,'%f %f',[1 3]);
 for i=counter+1:counter+b(2)
   a(:,i)=fscanf(fid,'%f %f',[n1 1]);
 end
 st=fclose(fid);
% Read one fixed file for geometry information
'Read fixed-0.3.net'
 for i=1:b(2)
   p(1,i)=a(1,i);
   p(2,i)=a(2,i);
   p(3,i)=0.45;
 end
 clear a;
 [pn]=tramnnmx(p,minp,maxp);
 net=newff([-1 1; -1 1; -1 1], [50,1], {'tansig','purelin'},'trainlm');
 net=init(net);
 net.IW=W;
 net.b=B;
 net.LW=L;
 ako=sim(net,pn);
 [treba]=postmnmx(ako,mint,maxt);
 fid=fopen('testir6.dat','w');
 for i=1:b(2)
   fprintf(fid,'%12.7f %12.7f
%15.7e
',p(1,i),p(2,i),p(3,i),treba(i));
 end
 st=fclose(fid);
 clear all;

load parmreze7.mat W B L minp maxp mint maxt;
nl=10;
counter=0;
 fid=fopen('fixed.net','r');
b=fscanf(fid,'%f %f',[1 3]);
 for i=counter+1:counter+b(2)
   a(:,i)=fscanf(fid,'%f %f',[n1 1]);
 end
 st=fclose(fid);
% Read one fixed file for geometry information
'Read fixed-0.3.net'
 for i=1:b(2)
   p(1,i)=a(1,i);
   p(2,i)=a(2,i);
   p(3,i)=0.45;
 end
 clear a;
 [pn]=tramnnmx(p,minp,maxp);
 net=newff([-1 1; -1 1; -1 1], [50,1], {'tansig','purelin'},'trainlm');
 net=init(net);
net.IW=W;
net.b=B;
net.LW=L;
ako=sim(net,pn);
[treba]=postmnmx(ako,mint,maxt);
fid=fopen('testir7.dat','w');
for i=1:b(2)
    fprintf(fid,'%12.7f %12.7f %15.7e
',p(1,i),p(2,i),p(3,i),treba(i));
end
st=fclose(fid);
clear all;

% Generate one source term file for CFD++
fid=fopen('testirl1.dat','r');
fid3=fopen('testir3.dat','r');
fid4=fopen('testir4.dat','r');
%fid5=fopen('testir5.dat','r');
fid6=fopen('testir6.dat','r');
fid7=fopen('testir7.dat','r');
fid8=fopen('fixed-0.3.net','r');
fid9=fopen('rhvecout.gen','w');

fiktc1=fscanf(fid8,'%d',[1]);
fiktc2=fscanf(fid8,'%d',[1]);
fiktc3=fscanf(fid8,'%d',[1]);
fprintf(fid9,'%12.7d
',fiktc1);
fprintf(fid9,'%12.7d
',fiktc2);
fprintf(fid9,'%12.7d
',fiktc3);
for i=1:fiktc2
    a=fscanf(fid,'%f %f %f',[1 3]);
    source1=a(3);
    source2=0.0;
    a=fscanf(fid3,'%f %f %f',[1 3]);
    source3=a(3);
    a=fscanf(fid4,'%f %f %f',[1 3]);
    source4=a(3);
    % a=fscanf(fid5,'%f %f %f',[1 3]);
    % source5=a(3);
    % source5=0.0;
    a=fscanf(fid6,'%f %f %f',[1 3]);
    source6=a(3);
    a=fscanf(fid7,'%f %f %f',[1 3]);
    source7=a(3);
    b=fscanf(fid8,'%d %d %f %f %f %f %f %f',[1 7]);
    fikt4=b(1);
    fikt5=b(2);
    fprintf(fid9,'%d %d %12.7d %12.7d %12.7d %12.7d %12.7d %12.7d %12.7d %12.7d %12.7d %12.7d
',fikt4,fikt5,source1,...
        source2,source3,source4,source5,source6,source7);
end

num1=c fscanf(fid8,'%f %d',[1 2]);
num1=c(1);
num2=c(2);
fprintf(fid9,'%12.7d %d',num1,num2);
close('all');

% Read All Boundary Condition Data
ald=1.0;
fid4=fopen('interface.net','w');
fid2=fopen('nodesin.net','r');
b2=fscanf(fid2,'%d %d',[1 3]);
for i=1:b(2)
    a2(:,i)=fscanf(fid2,'%d %f %f',[3 1]);
end
counter=0;
Mach=0.3;
fid=fopen('cdepsaverage-0.3.net','r');
fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
    a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f',[9 1]);
    a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
    xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,2,a3(5,i))/4;
    yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i))/4;
    if((xco >= 5) & (xco < 5+ald*2) & (yco > 2-0.00001) & (yco <
        2+0.00001))
        counter=counter+1;
    fprintf(fid4,'%f %f %f %f %f %f %f %f %f
',xco,yco,Mach,a(3,i),...
        a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
end

close(fid);
close(fid3);
clear a,a3,b,b3;
'Read cdepsaverage-0.3.net'
Mach=0.4;
fid=fopen('cdepsaverage-0.4.net','r');
fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
    a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f',[9 1]);
    a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
    xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,2,a3(5,i))/4;
    yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i))/4;
    if((xco >= 5) & (xco < 5+ald*2) & (yco > 2-0.00001) & (yco <
        2+0.00001))
        counter=counter+1;
    fprintf(fid4,'%f %f %f %f %f %f %f %f %f
',xco,yco,Mach,a(3,i),...
        a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
end

close(fid);
close(fid3);
clear a,a3,b,b3;
'Read cdepsaveaverage-0.4.net'
Mach=0.5;

fid=fopen('cdepsaveaverage-0.5.net','r');
"fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
    a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f %f',[9 1]);
    a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
    xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,a3(5,i))/4;
    yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i)))/4;
    if((xco >= 5) & (xco < 5+ald*2) & (yco > 2-0.00001) & (yco < 2+0.00001))
        counter=counter+1
        fprintf(fid4,'%f %f %f %f %f %f %f %f
          xco,yco,Mach,a(3,i),...
          a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
    end
end

close(fid);
clear a,a3,b,b3;
' Read cdepsaveaverage-0.5.net' 
Mach=0.6;

fid=fopen('cdepsaveaverage-0.6.net','r');
"fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
    a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f %f',[9 1]);
    a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
    xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,a3(5,i))/4;
    yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i)))/4;
    if((xco >= 5) & (xco < 5+ald*2) & (yco > 2-0.00001) & (yco < 2+0.00001))
        counter=counter+1
        fprintf(fid4,'%f %f %f %f %f %f %f %f
          xco,yco,Mach,a(3,i),...
          a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
    end
end

close(fid);
clear a,a3,b,b3;
' Read cdepsaveaverage-0.6.net' 
Mach=0.7;

fid=fopen('cdepsaveaverage-0.7.net','r');
"fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
    a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f %f',[9 1]);
}
a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,a3(5,i))/4;
yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i))/4;
if((xco >= 5) \& (xco < 5+ald*2) \& (yco > 2-0.00001) \& (yco < 
2+0.00001))
counter=counter+1;
fprintf(fid4,'%f %f %f %f %f %f %f %f %f 
',xco,yco,Mach,a(3,i),... 
a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
end
close(fid);
close(fid3);
clear a,a3,b,b3;
'Read cdepsaverage-0.7.net'
Mach=0.8;
fid=fopen('cdepsaverage-0.8.net','r');
 fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f %f %f',[9 1]);
a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,a3(5,i))/4;
yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i))/4;
if((xco >= 5) \& (xco < 5+ald*2) \& (yco > 2-0.00001) \& (yco < 
2+0.00001))
counter=counter+1;
fprintf(fid4,'%f %f %f %f %f %f %f %f %f 
',xco,yco,Mach,a(3,i),... 
a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
end
close(fid);
close(fid3);
clear a,a3,b,b3;
'Read cdepsaverage-0.8.net'
Mach=0.9;
 fid=fopen('cdepsaverage-0.9.net','r');
 fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f %f %f',[9 1]);
a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,a3(5,i))/4;
yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i))/4;
if((xco >= 5) \& (xco < 5+ald*2) \& (yco > 2-0.00001) \& (yco < 
2+0.00001))
counter=counter+1;
fprintf(fid4,'%f %f %f %f %f %f %f %f %f 
',xco,yco,Mach,a(3,i),... 
a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
end
close(fid);
close(fid3);
clear a,a3,b,b3;
'Read cdepsaverage-0.9.net'
Mach=1.0;
fid=fopen('cdepsaverage-1.0.net','r');
fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
    a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f %f',[9 1]);
    a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
    xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,a3(5,i))/4;
    yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i))/4;
    if((xco >= 5) & (xco < 5+ald*2) & (yco > 2-0.00001) & (yco <
        2+0.00001))
        counter=counter+1;
        fprintf(fid4,'%f %f %f %f %f %f %f %f %f
',xco,yco,Mach,a(3,i),...
    a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
end
close(fid);
close(fid3);
clear a,a3,b,b3;
'Read cdepsaverage-1.0.net'
Mach=1.1;
fid=fopen('cdepsaverage-1.1.net','r');
fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
    a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f %f',[9 1]);
    a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
    xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,a3(5,i))/4;
    yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i))/4;
    if((xco >= 5) & (xco < 5+ald*2) & (yco > 2-0.00001) & (yco <
        2+0.00001))
        counter=counter+1;
        fprintf(fid4,'%f %f %f %f %f %f %f %f %f
',xco,yco,Mach,a(3,i),...
    a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
end
close(fid);
close(fid3);
clear a,a3,b,b3;
'Read cdepsaverage-1.1.net'
Mach=1.2;
fid=fopen('cdepsaverage-1.2.net','r');
fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
% Start NN training for BCs
% Primitive Variable 1

a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f ',[9 1]);
a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,a3(5,i))/4;
yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i))/4;
if((xco >= 5) & (xco < 5+ald*2) & (yco > 2-0.00001) & (yco < 2+0.00001))
    counter=counter+1;
    fprintf(fid4,'%f %f %f %f %f %f %f %f %f
,%xco,yco,Mach,a(3,i),...
a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
end
close(fid);
close(fid3);
clear a,a3,b,b3;
'REad cdepsaverage-1.2.net'
Mach=1.3;
fid=fopen('cdepsaverage-1.3.net','r');
fid3=fopen('cellsin.net','r');
b=fscanf(fid,'%d %d',[1 3]);
b3=fscanf(fid3,'%d %d',[1 3]);
for i=1:b3(2)
    a(:,i)=fscanf(fid,'%d %d %f %f %f %f %f %f %f',[9 1]);
    a3(:,i)=fscanf(fid3,'%d %d %d %d %d',[5 1]);
    xco=(a2(2,a3(2,i)+a2(2,a3(3,i)+a2(2,a3(4,i)+a2(2,a3(5,i))/4;
    yco=(a2(3,a3(2,i)+a2(3,a3(3,i)+a2(3,a3(4,i)+a2(3,a3(5,i))/4;
    if((xco >= 5) & (xco < 5+ald*2) & (yco > 2-0.00001) & (yco < 2+0.00001))
        counter=counter+1;
        fprintf(fid4,'%f %f %f %f %f %f %f %f %f
,%xco,yco,Mach,a(3,i),...
        a(4,i),a(5,i),a(6,i),a(7,i),a(8,i),a(9,i));
    end
    close(fid);
close(fid3);
clear a,a3,b,b3;
'REad cdepsaverage-1.3.net'
clear b2,a2,a;
close(fid4);
% Start NN training for BCs
% Primitive Variable 1
fid4=fopen('interface.net','r');
a=zeros(12,counter);
p=zeros(3,n2);
t=zeros(1,n2);
for i=1:counter
    a=fscanf(fid4,'%f %f %f %f %f %f %f %f %f %f %f',[10 1]);
    p(1,i)=a(1,i);
    p(2,i)=a(2,i);
    p(3,i)=a(3,i);
    t(i)=a(4,i);
end
close(fid4);
% Normalize data so that all variables are in [-1;1] range
[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
iiival=ceil(3/4*n2)+1:n2;
iiitr=1:ceil(3/4*n2);
v.P=pn(:,iiival);
v.T=tn(:,iiival);
ptr=pn(:,iiitr);
ttr=tn(:,iiitr);
net=newff([-1 1; -1 1; -1 1], [70,1], {'tansig','purelin'},'trainlm');
net.trainParam.show=10;
net.trainParam.lr=0.005;
net.trainParam.mem_reduc=10;
net.trainParam.epochs=3000;
net.trainParam.min_grad=1e-10;
net=init(net);
%load parmreze1 W B L minp maxp mint maxt;
%net.IW=W;
%net.LW=L;
%net.b=B;
[net,tr]=train(net,ptr,ttr,[],[],v);
plot(tr.epoch,tr.perf,tr.epoch,tr.vperf,tr.epoch,tr.tperf)
legend('Training','Validation','Test','-1);
ylabel('Squared Error');
xlabel('Epoch');
W=net.IW;
B=net.b;
L=net.LW;
save parmreze1BC.mat W B L minp maxp mint maxt;
clear W,B,L,p,t;
% Primitive Variable 2
for i=1:counter
    p(1,i)=a(1,i);
    p(2,i)=a(2,i);
    p(3,i)=a(3,i);
    t(i)=a(5,i);
end
% Normalize data so that all variables are in [-1;1] range
[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
iiival=ceil(3/4*n2)+1:n2;
iiitr=1:ceil(3/4*n2);
v.P=pn(:,iiival);
v.T=tn(:,iiival);
ptr=pn(:,iiitr);
ttr=tn(:,iiitr);
net=newff([-1 1; -1 1; -1 1], [70,1], {'tansig','purelin'},'trainlm');
net.trainParam.show=10;
%net.trainParam.lr=0.005;
net.trainParam.mem_reduc=10;
net.trainParam.epochs=3000;
net.trainParam.min_grad=1e-10;
net=init(net);
%load parmreze2 W B L minp maxp mint maxt;
%net.IW=W;  
%net.LW=L;  
%net.b=B;  
[net, tr] = train(net, ptr, ttr, [], [], v);  
plot(tr.epoch, tr.perf, tr.epoch, tr.vperf, tr.epoch, tr.tperf)  
legend('Training', 'Validation', 'Test', '-1');  
ylabel('Squared Error');  
xlabel('Epoch');  
W = net.IW;  
B = net.b;  
L = net.LW;  
save parmreze2BC.mat W B L minp maxp mint maxt;  
clear W, B, L, p, t;  
% Primitive Variable 3  
for i = 1:counter  
    p(1, i) = a(1, i);  
    p(2, i) = a(2, i);  
    p(3, i) = a(3, i);  
    t(i) = a(6, i);  
end  
% Normalize data so that all variables are in [-1;1] range  
[pn, minp, maxp, tn, mint, maxt] = premnmx(p, t);  
iival = ceil(3/4*n2) + 1:n2;  
iitr = 1:ceil(3/4*n2);  
V.P = pn(:, iival);  
V.T = tn(:, iival);  
ptr = pn(:, iitr);  
ttr = tn(:, iitr);  
net = newff([-1 1; -1 1; -1 1], [70, 1], {'tansig', 'purelin'}, 'trainlm');  
net.trainParam.show = 10;  
%net.trainParam.lr = 0.005;  
net.trainParam.mem_reduc = 10;  
net.trainParam.epochs = 3000;  
net.trainParam.min_grad = 1e-10;  
net = init(net);  
%load parmreze3 W B L minp maxp mint maxt;  
%net.IW=W;  
%net.LW=L;  
%net.b=B;  
[net, tr] = train(net, ptr, ttr, [], [], v);  
plot(tr.epoch, tr.perf, tr.epoch, tr.vperf, tr.epoch, tr.tperf)  
legend('Training', 'Validation', 'Test', '-1');  
ylabel('Squared Error');  
xlabel('Epoch');  
W = net.IW;  
B = net.b;  
L = net.LW;  
save parmreze3BC.mat W B L minp maxp mint maxt;  
clear W, B, L, p, t;  
% Primitive Variable 4  
for i = 1:counter  
    p(1, i) = a(1, i);  
end
p(2,i)=a(2,i);
p(3,i)=a(3,i);
t(i)=a(7,i);

end

% Normalize data so that all variables are in [-1;1] range
[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
iival=ceil(3/4*n2)+1:n2;
iitr=1:ceil(3/4*n2);
v.P=pn(:,iival);
v.T=tn(:,iival);
ptr=pn(:,iitr);
ttr=tn(:,iitr);
net=newff([-1 1; -1 1; -1 1], [70,1], {'tansig','purelin'},'trainlm');
net.trainParam.show=10;
%net.trainParam.lr=0.005;
net.trainParam.mem_reduc=10;
net.trainParam.epochs=3000;
net.trainParam.min_grad=1e-10;
net=init(net);
%load parmreze4 W B L minp maxp mint maxt;
%net.IW=W;
%net.LW=L;
%net.b=B;
[net, tr] = train(net, ptr, ttr, [], [], v);
plot(tr.epoch, tr.perf, tr.epoch, tr.vperf, tr.epoch, tr.tperf)
legend('Training', 'Validation', 'Test', '-1');
ylabel('Squared Error');
xlabel('Epoch');
W=net.IW;
B=net.b;
L=net.LW;
save parmreze4BC.mat W B L minp maxp mint maxt;
clear W,B,L,p,t;

% Primitive Variable 5
%for i=1:counter
%   p(1,i)=a(1,i);
%   p(2,i)=a(2,i);
%   p(3,i)=a(3,i);
%   t(i)=a(8,i);
%end

% Normalize data so that all variables are in [-1;1] range
[pn, minp, maxp, tn, mint, maxt] = premnmx(p, t);
iival = ceil(3/4*n2)+1:n2;
iitr = 1:ceil(3/4*n2);
v.P = pn(:, iival);
v.T = tn(:, iival);
ptr = pn(:, iitr);
ttr = tn(:, iitr);
net = newff([-1 1; -1 1; -1 1], [70,1], {'tansig','purelin'},'trainlm');
net.trainParam.show = 10;
%net.trainParam.lr = 0.005;
%net.trainParam.mem_reduc=10;
%net.trainParam.epochs=3000;
%net.trainParam.min_grad=1e-10;
%net=init(net);
%load parmreze5 W B L minp maxp mint maxt;
%net.IW=W;
%net.LW=L;
%net.b=B;
%[net, tr] = train(net, ptr, ttr, [], [], v);
%plot(tr.epoch, tr.perf, tr.epoch, tr.vperf, tr.epoch, tr.tperf)
%legend('Training', 'Validation', 'Test', -1);
%ylabel('Squared Error');
%xlabel('Epoch');
%W = net.IW;
%B = net.b;
%L = net.LW;
%save parmreze5BC.mat W B L minp maxp mint maxt;
%clear W, B, L, p, t;
% Primitive Variable 6
for i = 1: counter
    p(1, i) = a(1, i);
    p(2, i) = a(2, i);
    p(3, i) = a(3, i);
    t(i) = a(9, i);
end;

% Normalize data so that all variables are in [-1;1] range
[pn, minp, maxp, tn, mint, maxt] = premnmx(p, t);
iival = ceil(3/4 * n2) + 1:n2;
iitr = 1: ceil(3/4 * n2);
v.P = pn(:, iival);
v.T = tn(:, iival);
ptr = pn(:, iitr);
ttr = tn(:, iitr);
net = newff([-1 1; -1 1; -1 1], [70, 1], {'tansig', 'purelin'}, 'trainlm');
net.trainParam.show = 10;
%net.trainParam.lr = 0.005;
%net.trainParam.mem_reduc = 10;
%net.trainParam.epochs = 3000;
%net.trainParam.min_grad = 1e-10;
%net = init(net);
%load parmreze6 W B L minp maxp mint maxt;
%net.IW = W;
%net.LW = L;
%net.b = B;
%[net, tr] = train(net, ptr, ttr, [], [], v);
%plot(tr.epoch, tr.perf, tr.epoch, tr.vperf, tr.epoch, tr.tperf)
%legend('Training', 'Validation', 'Test', -1);
%ylabel('Squared Error');
xlabel('Epoch');
%W = net.IW;
%B = net.b;
%L = net.LW;
save parmreze6BC.mat W B L minp maxp mint maxt;
clear W, B, L, p, t;

% Primitive Variable 7
for i=1:counter
    p(1, i) = a(1, i);
    p(2, i) = a(2, i);
    p(3, i) = a(3, i);
    t(i) = a(10, i);
end

% Normalize data so that all variables are in [-1;1] range
[pn, minp, maxp, tn, mint, maxt] = premnmx(p, t);
iival = ceil(3/4 * n2) + 1:n2;
iitr = 1:ceil(3/4 * n2);
v_P = pn(:, iival);
v_T = tn(:, iival);
ptr = pn(:, iitr);
ttr = tn(:, iitr);
net = newff([-1 1; -1 1; -1 1], [70, 1], {'tansig', 'purelin'}, 'trainlm');
net.trainParam.show = 10;
% net.trainParam.lr = 0.005;
net.trainParam.mem_reduc = 10;
net.trainParam.epochs = 3000;
net.trainParam.min_grad = 1e-10;
net = init(net);

% load parmreze7 W B L minp maxp mint maxt;
% net.IW = W;
% net.LW = L;
[net, tr] = train(net, ptr, ttr, [], [], v);
plot(tr.epoch, tr.perf, tr.epoch, tr.vperf, tr.epoch, tr.tperf)
legend('Training', 'Validation', 'Test', '-l');
ylabel('Squared Error');
xlabel('Epoch');
W = net.IW;
B = net.b;
L = net.LW;
save parmreze7BC.mat W B L minp maxp mint maxt;
clear W, B, L, p, t;

% Test the trained Neural Network for M = 0.45
load parmreze1BC.mat W B L minp maxp mint maxt;

% Test the trained Neural Network for M = 0.45
load parmreze1BC.mat W B L minp maxp mint maxt;

fid = fopen('interface.net', 'r');
for i = 1:counter
    a(:, i) = fscanf(fid, '%f %f %f %f %f %f %f %f %f', [10 1]);
end
st = fclose(fid);
for i = 1:b(2)
    p(1, i) = a(1, i);
    p(2, i) = a(2, i);
    p(3, i) = 0.45;
end
[pn] = tranmnmx(p, minp, maxp);
net=newff([-1 1; -1 1; -1 1], [70,1], {'tansig','purelin'},'trainlm');
net=init(net);
net.IW=W;
net.b=B;
net.LW=L;
ako=sim(net,pn);
[treba]=postmnmx(ako,mint,maxt);
fid=fopen('testir1BC.dat','w');
for i=1:b(2)
    fprintf(fid,'%12.7f %12.7f %15.7e
',p(1,i),p(2,i),p(3,i),treba(i));
end
st=fclose(fid);
clear all;
load parmreze2BC.mat W B L minp maxp mint maxt;
 fid=fopen('interface.net','r');
 for i=1:counter
     a(:,i)=fscanf(fid,'%f %f %f %f %f %f %f %f %f %f',[10 1]);
 end
st=fclose(fid);
for i=1:b(2)
    p(1,i)=a(1,i);
    p(2,i)=a(2,i);
    p(3,i)=0.45;
end
[pn]=tramnmx(p,minp,maxp);
net=newff([-1 1; -1 1; -1 1], [70,1], {'tansig','purelin'},'trainlm');
net=init(net);
net.IW=W;
net.b=B;
net.LW=L;
ako=sim(net,pn);
[treba]=postmnmx(ako,mint,maxt);
 fid=fopen('testir2BC.dat','w');
for i=1:b(2)
    fprintf(fid,'%12.7f %12.7f %15.7e
',p(1,i),p(2,i),p(3,i),treba(i));
end
st=fclose(fid);
clear all;
load parmreze3BC.mat W B L minp maxp mint maxt;
 fid=fopen('interface.net','r');
 for i=1:counter
     a(:,i)=fscanf(fid,'%f %f %f %f %f %f %f %f %f %f',[10 1]);
 end
st=fclose(fid);
for i=1:b(2)
    p(1,i)=a(1,i);
    p(2,i)=a(2,i);
    p(3,i)=0.45;
end
[pn]=tramnmx(p,minp,maxp);
net=newff([-1 1; -1 1; -1 1], [70,1], {'tansig','purelin'},'trainlm');
net=init(net);
net.IW=W;
net.b=B;
net.LW=L;
ako=sim(net,pn);
[treba]=postmnmx(ako,mint,maxt);
 fid=fopen('testir3BC.dat','w');
 for i=1:b(2)
    fprintf(fid,'%12.7f %12.7f
%15.7e\n',p(1,i),p(2,i),p(3,i),treba(i));
 end
 st=fclose(fid);
clear all;
load parmreze4BC.mat W B L minp maxp mint maxt;
 fid=fopen('interface.net','r');
 for i=1:counter
    a(:,i)=fscanf(fid,'%f %f %f %f %f %f %f %f %f',[10 1]);
 end
 st=fclose(fid);
for i=1:b(2)
    p(1,i)=a(1,i);
    p(2,i)=a(2,i);
    p(3,i)=0.45;
end
 [pn]=tramnmx(p,minp,maxp);
net=newff([-1 1; -1 1; -1 1], [70,1], {'tansig','purelin'},'trainlm');
net=init(net);
net.IW=W;
net.b=B;
net.LW=L;
ako=sim(net,pn);
[treba]=postmnmx(ako,mint,maxt);
 fid=fopen('testir4BC.dat','w');
 for i=1:b(2)
    fprintf(fid,'%12.7f %12.7f
%15.7e\n',p(1,i),p(2,i),p(3,i),treba(i));
 end
 st=fclose(fid);
clear all;
%load parmreze5BC.mat W B L minp maxp mint maxt;
%fid=fopen('interface.net','r');
% for i=1:counter
%    a(:,i)=fscanf(fid,'%f %f %f %f %f %f %f %f %f %f',[10 1]);
% end
% st=fclose(fid);
% for i=1:b(2)
%    p(1,i)=a(1,i);
%    p(2,i)=a(2,i);
%    p(3,i)=0.45;
% end
% [pn]=tramnmx(p,minp,maxp);
%net=newff([-1 1; -1 1; -1 1], [70,1],
{'tansig', 'purelin'}, 'trainlm');
%net=init(net);
%net.IW=W;
%net.b=B;
%net.LW=L;
ako=sim(net, pn);
%[treba]=postmnmx(ako, mint, maxt);
%fid=fopen('testir5BC.dat', 'w');
%for i=1:b(2)
% fprintf(fid, '12.7f %12.7f %15.7e
', p(1,i), p(2,i), p(3,i), treba(i));
%end
%st=fclose(fid);
%clear all;
load parmreze6BC.mat W B L minp maxp mint maxt;
fid=fopen('interface.net', 'r');
for i=1:counter
    a(:, i)=fscanf(fid, '%f %f %f %f %f %f %f %f %f %f', [10 1]);
end
st=fclose(fid);
for i=1:b(2)
    p(1, i)=a(1, i);
    p(2, i)=a(2, i);
    p(3, i)=0.45;
end
[PN]=tramnxm(p, minp, maxp);
net=newff([-1 1; -1 1; -1 1], [70,1], {'tansig', 'purelin'}, 'trainlm');
net=init(net);
net.IW=W;
net.b=B;
net.LW=L;
ako=sim(net, pn);
[treba]=postmnmx(ako, mint, maxt);
%fid=fopen('testir6BC.dat', 'w');
%for i=1:b(2)
% fprintf(fid, '12.7f %12.7f %15.7e
', p(1,i), p(2,i), p(3,i), treba(i));
%end
%st=fclose(fid);
%clear all;
load parmreze7BC.mat W B L minp maxp mint maxt;
fid=fopen('interface.net', 'r');
for i=1:counter
    a(:, i)=fscanf(fid, '%f %f %f %f %f %f %f %f %f %f', [10 1]);
end
st=fclose(fid);
for i=1:b(2)
    p(1, i)=a(1, i);
    p(2, i)=a(2, i);
    p(3, i)=0.45;
end
[pn] = tramnmx(p, minp, maxp);
net = newff([-1 1; -1 1; -1 1], [70, 1], {'tansig', 'purelin'}, 'trainlm');
net = init(net);
net.IW = W;
net.b = B;
net.LW = L;
ako = sim(net, pn);
[treba] = postmnmx(ako, mint, maxt);
fid = fopen('testir7BC.dat', 'w');
for i = 1:b(2)
    fprintf(fid, '%12.7f %12.7f %15.7e
', p(1, i), p(2, i), p(3, i), treba(i));
end
st = fclose(fid);
clear all;

% Generate one BCs file for CFD++
fid = fopen('testir1BC.dat', 'r');
fid2 = fopen('testir2BC.dat', 'r');
fid3 = fopen('testir3BC.dat', 'r');
fid4 = fopen('testir4BC.dat', 'r');
% fid5 = fopen('testir5BC.dat', 'r');
fid6 = fopen('testir6BC.dat', 'r');
fid7 = fopen('testir7BC.dat', 'r');
fid8 = fopen('cdepsaverage-0.3.net', 'r');
fid9 = fopen('interface.gen', 'w');
fid10 = fopen('nodesin.net', 'r');
fid11 = fopen('cellsin.net', 'r');
fiktc1 = fscanf(fid8, '%d', [1]);
fiktc2 = fscanf(fid8, '%d', [1]);
fiktc3 = fscanf(fid8, '%d', [1]);
fprintf(fid9, '%12.7d\n', fiktc1);
fprintf(fid9, '%12.7d\n', fiktc2);
fprintf(fid9, '%12.7d\n', fiktc3);
b2 = fscanf(fid10, '%d %d', [1 3]);
for i = 1:b2(2)
    a2(:, i) = fscanf(fid10, '%d %f %f', [3 1]);
end
b3 = fscanf(fid11, '%d %d', [1 3]);
ald = 1.0;
for i = 1:b3(2)
    a(:, i) = fscanf(fid8, '%d %d %f %f %f %f %f %f %f', [9 1]);
a3(:, i) = fscanf(fid11, '%d %d %d %d %d', [5 1]);
xco = (a2(2, a3(2, i) + a2(2, a3(3, i) + a2(2, a3(4, i) + a2(2, a3(5, i)))/4;
yco = (a2(3, a3(2, i) + a2(3, a3(3, i) + a2(3, a3(4, i) + a2(3, a3(5, i)))/4;
if ((xco >= 5) & (xco < 5 + ald * 2) & (yco > 2 - 0.00001) & (yco < 2 + 0.00001))
    m = fscanf(fid, '%12.7f %12.7f %12.7f %15.7e', [4 1]);
m2 = fscanf(fid2, '%12.7f %12.7f %12.7f %15.7e', [4 1]);
m3 = fscanf(fid3, '%12.7f %12.7f %12.7f %15.7e', [4 1]);
m4 = fscanf(fid4, '%12.7f %12.7f %12.7f %15.7e', [4 1]);
% m5 = fscanf(fid5, '%12.7f %12.7f %12.7f %15.7e', [4 1]);
m5 = 0.0;
m6 = fscanf(fid6, '%12.7f %12.7f %15.7e', [4 1]);
m7 = fscanf(fid7, '%12.7f %12.7f %15.7e', [4 1]);
fprintf(fid9, '%d %d %f %f %f %f %f %f
', a(1, i), a(2, i), m1, m2, m3, m4, m5, m6, m7);
else
    fprintf(fid9, '%d %d %f %f %f %f %f %f
', a(1, i), a(2, i), a(3, i), a(4, i), a(5, i), a(6, i), a(7, i), a(8, i), a(9, i));
end
close(fid);
c = fscanf(fid8, '%f %d', [1 2]);
num1 = c(1);
num2 = c(2);
fprintf(fid9, '%12.7d %d
', num1, num2);
close('all');
Bibliography


23. Kim, I. and Chokani, N.: “Navier-Stokes Study of Supersonic Cavity Flowfield with

Aeroacoustics with Suppression Devices”, Journal of Vibration and Acoustics, Vol. 116,
1994, pp. 105-112.


