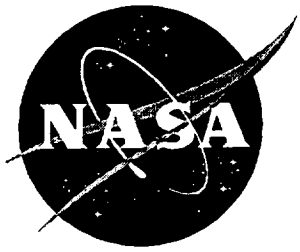
NASA/TM-1998-112233



Two-Dimensional High-Lift Aerodynamic Optimization Using Neural Networks

Roxana M. Greenman
Ames Research Center, Moffett Field, California

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

June 1998

Acknowledgments

I wish to express my appreciation to Dr. Karlin R. Roth at NASA Ames Research Center for her valuable advice and guidance throughout this study. I would also like to extend my appreciation to Dr. James C. Ross, Dr. Stuart E. Rogers, and Dr. Charles C. Jorgensen at NASA Ames Research Center. This work would not have been possible without their helpful discussions and suggestions.

Available from the following:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

Contents

List of Tables

List of Figures

Nomenclature

ROMAN

a	speed of sound
$\mathbf{A}$	linear constraint matrix
$\mathbf{A}$	Jacobian matrix
$\mathbf{A}^T$	transpose of matrix $\mathbf{A}$
c	chord, inches
$\mathbf{c}$	nonlinear constraint matrix
c_p	specific heat at constant pressure
c_v	specific heat at constant volume
C_d	drag coefficient, $C_d \equiv \frac{D}{q_\infty S}$
C_l	lift coefficient, $C_l \equiv \frac{L}{q_\infty S}$
C_m	moment coefficient, $C_m \equiv \frac{M}{q_\infty S c}$
C_p	pressure coefficient, $C_p \equiv \frac{p - p_\infty}{q_\infty}$
CPU	central processing unit
d	desired output
d	distance
$\boldsymbol{d}$	search direction

deg.	degree
D	drag force, lbs.
DV	design variable
e	total energy per unit volume
e_I	internal energy per unit mass
exp	experiment
E	error function
$\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{G}$	inviscid flux vectors
$f(\mathbf{x})$	objective function
$F(\mathbf{x})$	objective function
$\mathbf{g}(\mathbf{x})$	gradient vector, $\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x})$
$\boldsymbol{G}(\mathbf{x})$	Hessian matrix, second derivative matrix, $\mathbf{G}(\mathbf{x}) = \nabla^2 f(\mathbf{x})$
$\boldsymbol{H}(\mathbf{x})$	Hessian matrix, second derivative matrix, $\boldsymbol{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x})$
$\boldsymbol{H}_Q$	transformed Hessian
i	discrete spacial index
$\boldsymbol{I}$	Identity matrix
J	Jacobian
k	coefficient of thermal conductivity
K	number of input pairs
$\boldsymbol{l}$	lower bound vector
l	layer
lbs.	pounds

L	total number of layers
L	lift force, lbs.
L	reference length, inches
L/D	lift-to-drag ratio
mod	modified
M	number of outputs
M	pitching moment, lbs./inches
n	pseudo-time level
nn	neural network
NP	nonlinear programming
ol	overlap
orig	original
p	pressure, lbs./inches^2
p	search direction
PDE	partial differential equation
PDR	pressure difference rule
PE	processing element
Pr	Prandtl number, $Pr \equiv \frac{\mu c_p}{k}$
$q(\mathbf{x})$	quadratic function
q_∞	freestream dynamic pressure, $q_\infty \equiv \frac{1}{2}\rho_\infty V_\infty^2$
Q	vector of conserved mass, momentum, and energy
QP	quadratic programming

rms	root-mean-square
$\boldsymbol{R}$	residual vector
Re	Reynolds number based on freestream conditions and reference length, $Re = \frac{\rho_\infty V_\infty L}{\mu_\infty}$
s	change in $\boldsymbol{x}$, $s = \bar{x} - x$
S	reference area, inches^2
S	magnitude of vorticity, radians/seconds
t	time, seconds
$\boldsymbol{u}$	upper bound vector
u, v, w	velocity components in the x, y, z directions, feet/second
U, V, W	contravariant velocity components, feet/second
V	freestream velocity, feet/second
$\mathbf{x}$	vector of design variables in an optimization problem
x^*	first-order Kuhn-Tucker point
$\bar{x}$	new iterate, $\bar{x} = x + \alpha p$
$\mathbf{x}^{(k)}$ $k=1, 2, \ldots$	iterates in an iterative method
x, y, z	Cartesian coordinates
x_i	inputs
y	actual output
w_i	connection weights of processing element

GREEK

α	angle of attack, degrees
α	step length
β	artificial compressibility factor
γ	ratio of specific heats, $\gamma \equiv \frac{c_p}{c_v}$
δ	central-difference operator
δ	deflection angle, degrees
$\boldsymbol{\delta}$, $\boldsymbol{\delta}^{(k)}$	correction to $\mathbf{x}^{(k)}$
$\Delta C_{p_{diff}}$	pressure difference, $\Delta C_{p_{diff}} = C_{p_{peak}} - C_{p_{te}}$
Θ_i^l	threshold of the *ith* processing element in the *lth* layer
λ	coefficient of bulk viscosity
$\boldsymbol{\lambda}$	Lagrange multiplier vector for general constraints
μ	coefficient of viscosity
μ_τ	turbulent eddy viscosity
ν	scalar
ν	weighting parameter
$\boldsymbol{\xi}$	Lagrange multiplier vector for the bounds
ξ, η, ζ	transformed coordinates
ρ	density, slugs/feet^3
$\tilde{\rho}$	artificial density
σ	nonnegative step length
τ	computational time

τ	shear stress
χ	non-linear eddy viscosity
ω	wall vorticity, radians/second
$\Omega^{(k)}$	neighborhood of x

Subscripts

f	flap
max	maximum
nn	neural network prediction
ref	reference
s	slat
t	transition point
te	trailing-edge
v	viscous
∞	freestream values

Superscripts

k	iteration level
n	time level
$^\circ$	degrees

TWO-DIMENSIONAL HIGH-LIFT AERODYNAMIC OPTIMIZATION USING NEURAL NETWORKS

Roxana M. Greenman
Ames Research Center

ABSTRACT

Artificial neural networks were successfully used to minimize the amount of data required to completely define the aerodynamics of a three-element airfoil. The ability of the neural nets to accurately predict the aerodynamic coefficients (lift, drag, and moment coefficients), for any high-lift flap deflection, gap, and overlap, was demonstrated for both computational and experimental training data sets. Multiple input, single output networks were trained using the NASA Ames variation of the Levenberg-Marquardt algorithm for each of the aerodynamic coefficients. The computational data set was generated using a two-dimensional incompressible Navier-Stokes algorithm with the Spalart-Allmaras turbulence model. In high-lift aerodynamics, both experimentally and computationally, it is difficult to predict the maximum lift, and at which angle of attack it occurs. In order to accurately predict the maximum lift in the computational data set, a maximum lift criteria was needed. The "pressure difference rule," which states that there exists a certain pressure difference between the peak suction pressure and the pressure at the trailing edge of the element at the maximum lift condition, was applied to all three elements. In this study it was found that only the pressure difference on the slat element was needed to predict maximum lift. The neural nets were trained with only three different values of each of the parameters stated at various angles of attack. The entire computational data set was thus sparse and yet by using only 55 - 70% of the computed data, the trained neural networks predicted the aerodynamic coefficients within an acceptable accuracy defined to be the experimental error.

A high-lift optimization study was conducted by using neural nets that are trained with computational data. Artificial neural networks have been successfully integrated with a gradient based optimizer to minimize the amount of data required to completely define the design space of a three-element airfoil. This design process successfully optimized flap deflection, gap, overlap, and angle of attack to maximize lift. Once the neural nets were trained and integrated with the optimizer, minimal additional computer resources are required to perform optimization runs with different initial conditions and parameters. Neural networks "within the process" reduced the amount of computational time and resources needed in high-lift rigging optimization.

Chapter 1

Introduction

The design of an aircraft's high-lift system is a crucial part of the design phase of commercial and military airplanes since this system controls the takeoff and landing performance. The importance of a well designed high-lift system is seen by increased payloads which also increase the operational flexibility by extending ranges and by decreasing take-off and landing distances. Traditionally, high-lift designs have been accomplished by extensive wind tunnel and flight test programs which are expensive and difficult due to the extremely complex flow interactions. Recently, computational fluid dynamics (CFD) has been incorporated in high-lift design [1]. For high-lift applications, CFD can also be expensive because the entire design space is large, grids must be generated around geometrically-complex high-lift devices, and complex flow phenomena must be resolved. The complexity of the high-lift system of a typical civil transport airplane is shown in Figure 1.1. In order to achieve optimum rapid designs, new tools for speedy and efficient analysis of high-lift configurations are required. For these tools to be effective, they need to be functional in all areas of design including wind tunnel, CFD, and flight.

Figure 1.1 Typical civil transport with complex high-lift system.

1.1 Background

Artificial neural networks are a collection (or network) of simple computational devices which are modeled after the architecture of biological nervous systems. The ability of neural networks to accurately learn and predict nonlinear multiple input and output relationships makes them a promising technique in modeling nonlinear aerodynamic data. Computational fluid dynamics in conjunction with neural networks and optimization may help reduce the time and resources needed to accurately define the optimal aerodynamics of an aircraft including high-lift. Essentially, the neural networks will reduce the amount of data required to define the aerodynamic characteristics of an aircraft while the optimizer will allow the design space to be easily searched for extremas. Figure 1.2 shows a visual depiction of the agile artificial intelligence (AI) enhanced design space capture and smart surfing process that is developed. The design space data source is represented by black dots. In this study, computational fluid dynamics will be used to generate the data. The entire design space analyzed is shown in the figure with a carpet map. The neural network will be able to capture the design space with the small amount of data that is generated. Next, the optimizer will be able to locate extremas in the design space by using the captured design space to calculate the path that must be followed to reach a maxima. The agile artificial intelligence (AI) design space capture and surfing process is shown in Figure 1.2.

Recently, neural networks have been applied to a wide range of problems in the aerospace industry. For example, neural networks have been used in aerodynamic performance optimization of rotor blade design [2]. The study demonstrated that for several rotor blade designs, neural networks were advantageous in reducing the time required for the optimization. Faller and Schreck [3] successfully used neural networks to predict real-time three-dimensional unsteady separated flowfields and aerodynamic coefficients of a pitching wing. It has also been demonstrated that neural networks are capable of predicting measured data with sufficient accuracy to enable identification of instrumentation system degradation [4]. Steck and Rokhsaz [5] demon-

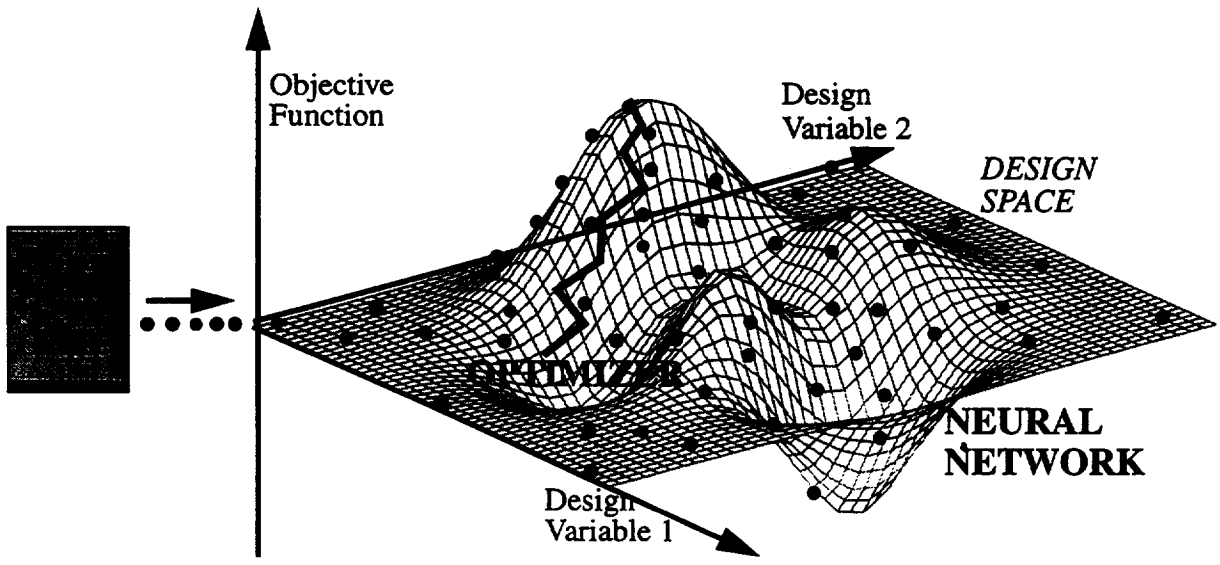


Figure 1.2 Agile AI-enhanced design space capture and smart surfing.

strated that neural networks can be successfully trained to predict aerodynamic forces with sufficient accuracy for design and modeling. Rai and Madavan [6] demonstrated the feasibility of applying neural networks to aerodynamic design of turbomachinery airfoils.

Neural networks have been used at NASA Ames Research Center to minimize the amount of data required to define the aerodynamic performance characteristics of a wind tunnel model [7-8]. It was shown that when only 50% of the data acquired from the wind tunnel test was used to train neural nets, the results had a predictive accuracy equal to or better than the experimental data. The success of the NASA Ames neural network application for wind tunnel data prompted this current study to use neural networks to minimize the amount of computational data required to accurately train neural networks to predict high-lift aerodynamics of a multi-element airfoil.

1.2 Agile AI-Enhanced Design Process

This paper describes a process which allows CFD to impact high-lift design. This process has three phases: 1) generation of the training database using CFD; 2) training of the neural networks; and 3) integration of the trained neural networks with an optimizer to capture and surf (search) the high-lift design space (refer to Figure 1.3). In this study, an incompressible two-dimensional Navier-Stokes solver is used to compute the flowfield about the three-element airfoil shown in Figure 1.4. The selected airfoil is a cross-section of the Flap-Edge model [9] that was tested in the 7- by 10-Foot Wind Tunnel No. 1 at the NASA Ames Research Center. Extensive wind-tunnel investigations [9] have been carried out for the Flap-Edge geometry shown in Figure 1.4. The model is a three-element unswept wing consisting of a 12%c LB-546 slat, NACA 63_2-215 Mod B main element and a 30%c Fowler flap where c is chord and is equal to $c = 30.0$ inches for the undeflected (clean, all high-lift components stowed) airfoil section.

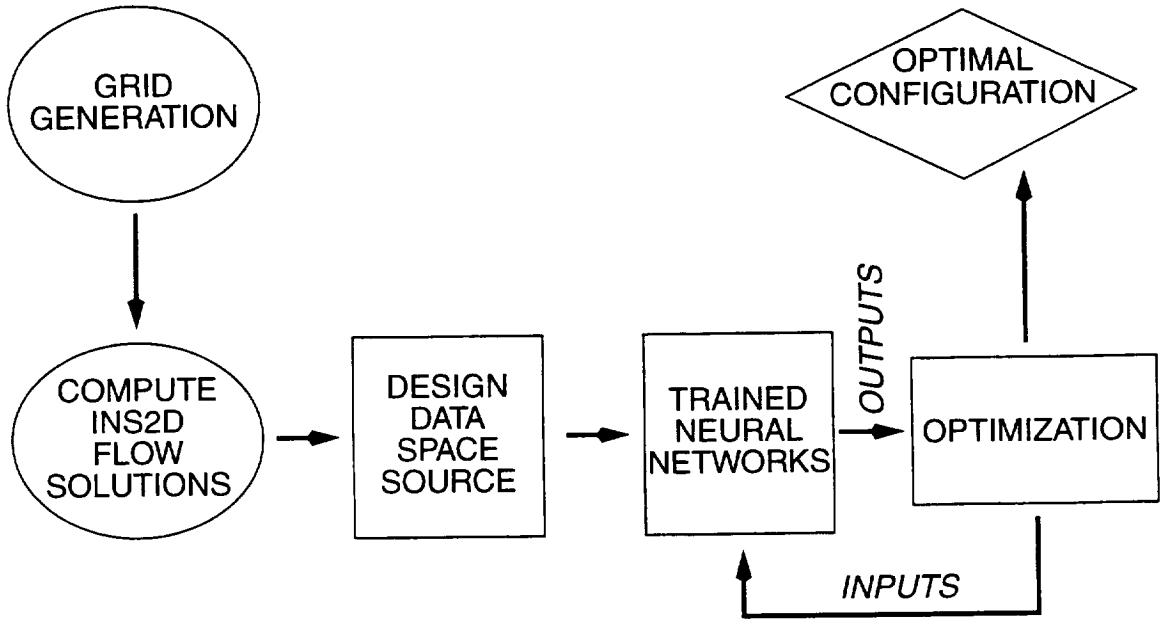


Figure 1.3 Illustration of AI-enhanced design process.

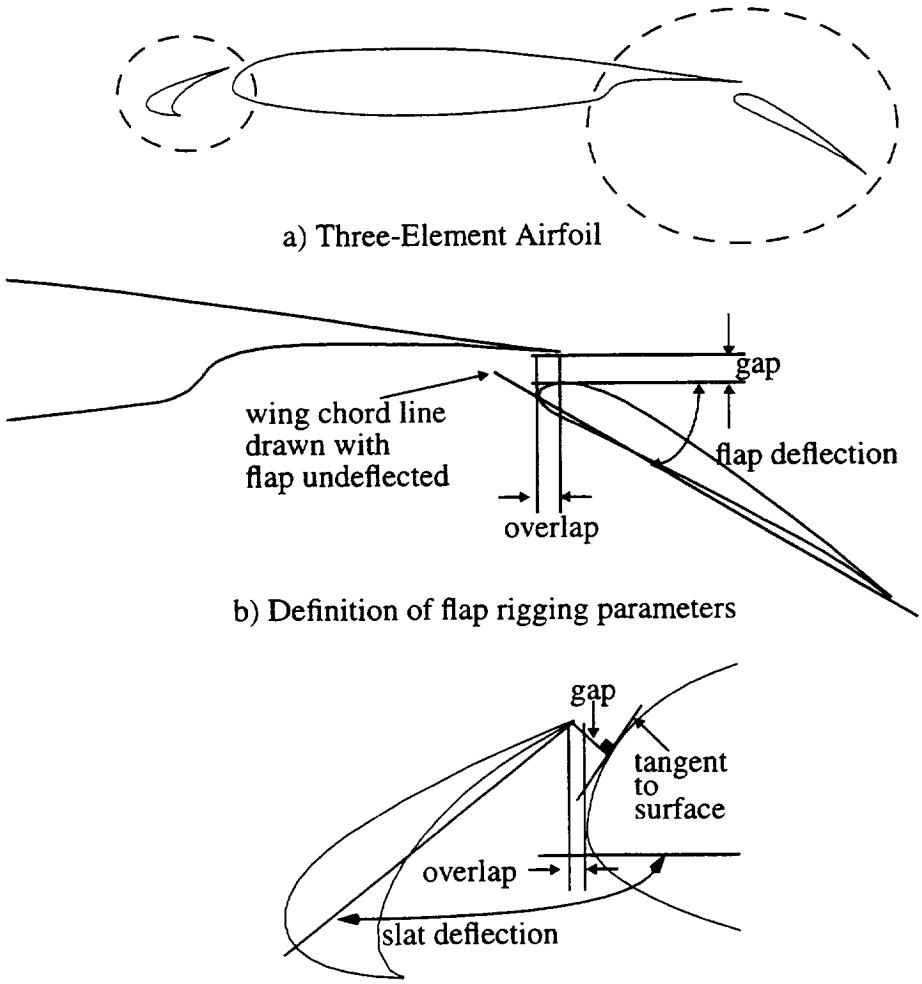


Figure 1.4 Flap-Edge Geometry and definition of flap and slat high-lift rigging.

Within the CFD database for this flap optimization problem, there are two different slat deflection settings, six and twenty-six degrees, and for each, 27 different flap riggings (refer to Figure 1.4b) are computed for ten different angles of attack. Each slat has a gap of gap_s = *2.0%c* and an overlap of ol_s = *-0.05%c*. The slat gap is measured from the slat trailing edge to the point of the main element whose tangent to the surface is perpendicular to the line of measurement. All gap and overlap values in this paper are expressed in terms of percent chord, *%c*. The neural networks are trained by using the flap riggings and angles of attack as the inputs and the aerodynamic forces as the outputs. The neural networks are defined to be successfully trained to predict the aerodynamic coefficients when given a set of inputs that are not in the training set, the outputs are predicted within the experimental error. The experimental error of the total lift coefficient (C_l) is ±0.02 for $C_l \leq 0.95 C_{l_{max}}$ and ±0.06 for $C_l \geq 0.95 C_{l_{max}}$. Finally, the trained neural networks are integrated with the optimizer to allow the design space to be easily searched for points of interest. It will be shown that this agile, artificial intelligence enhanced design process minimizes the cost and time required to accurately optimize the high-lift flap rigging.

This work is presented in the following chapters. The governing equations are presented in Chapter 2. A description of the numerical approach is presented in Chapter 3, including grid generation and boundary conditions. Next, the neural networks are discussed in Chapter 4. The optimizer and the optimization process are described in Chapter 5. Chapter 6 analyzes the results of the neural networks' ability to learn and predict the aerodynamic performance characteristics and the results of the AI optimization process. Finally, Chapter 7 summarizes the results and discussion and also presents recommendations.

Chapter 2

Governing Equations

The governing equations of fluid dynamics are derived from three conservation laws of mass, momentum, and energy. The Navier-Stokes equations were derived independently by Navier and Stokes in the mid-nineteenth century. Although they were originally derived to include only the conservation of momentum, it is now customary to include the conservation of mass and energy in the complete set of the Navier-Stokes equations. The present discussion begins with a description of the Navier-Stokes equations. Next, a coordinate transformation of the governing equations is discussed. The Reynolds-Averaged Navier-Stokes Equations, a special form of the governing equations, are then presented. Lastly, a one-equation turbulence model that is used to model the turbulent eddy viscosity is presented.

2.1 The Navier-Stokes Equations

The universal laws of the conservation of mass, momentum, and energy are the basis of the fundamental equations of fluid dynamics. These conservation laws are used to compose the three-dimensional Navier-Stokes equations which are the governing equations for a Newtonian fluid. A Newtonian fluid is a fluid where the stress is linearly dependent on the rate of strain. The majority of aerodynamic applications deal with air, other gases, or water which are Newtonian fluids. The Navier-Stokes equations are a set of five coupled, nonlinear partial differential equations which are the foundation of the science of viscous flow theory [10]. Upon assuming that body forces and the addition of external heat are negligible the Navier-Stokes equations can be written in nondimensional conservation law form as

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} = \frac{1}{Re}\left(\frac{\partial \mathbf{E}_v}{\partial x} + \frac{\partial \mathbf{F}_v}{\partial y} + \frac{\partial \mathbf{G}_v}{\partial z}\right) \tag{2.1}$$

where $\mathbf{Q}$ is the vector of conserved mass, momentum, and energy given by

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix} \tag{2.2}$$

The inviscid flux vectors, **E**, **F**, and **G**, are defined as

$$\mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (e+p)u \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ (e+p)v \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ (e+p)w \end{bmatrix} \tag{2.3}$$

where ρ is density, u, v, and w are the x, y, and z velocity components, respectively, p is pressure, and e is the total energy per unit volume. In equation (2.1), the Reynolds number, Re, is defined as

$$Re = \frac{\rho_\infty a_\infty L}{\mu_\infty} \tag{2.4}$$

Here, a is the speed of sound, L is a reference length, μ is the coefficient of viscosity, and the subscript ∞ denotes freestream values. The Reynolds number indicates the relative importance of inertial and viscous effects in fluid motion. The viscous flux vectors, $\mathbf{E}_v$, $\mathbf{F}_v$, and $\mathbf{G}_v$, are defined as

$$\mathbf{E}_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ \beta_x \end{bmatrix} \qquad \mathbf{F}_v = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ \beta_y \end{bmatrix} \qquad \mathbf{G}_v = \begin{bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ \beta_z \end{bmatrix} \tag{2.5}$$

where

$$
\begin{aligned}
\tau_{xx} &= \lambda(u_x + v_y + w_z) + 2\mu u_x \\
\tau_{yy} &= \lambda(u_x + v_y + w_z) + 2\mu v_y \\
\tau_{zz} &= \lambda(u_x + v_y + w_z) + 2\mu w_z \\
\tau_{xy} &= \tau_{yx} = \mu(u_y + v_x) \\
\tau_{xz} &= \tau_{zx} = \mu(u_z + w_x) \\
\tau_{yz} &= \tau_{zy} = \mu(v_z + w_y) \\
\beta_x &= \gamma k Pr^{-1} \partial_x e_I + u\tau_{xx} + v\tau_{xy} + w\tau_{xz} \\
\beta_y &= \gamma k Pr^{-1} \partial_y e_I + u\tau_{yx} + v\tau_{yy} + w\tau_{yz} \\
\beta_z &= \gamma k Pr^{-1} \partial_z e_I + u\tau_{zx} + v\tau_{zy} + w\tau_{zz}
\end{aligned}
\tag{2.6}
$$

The Prandtl number, Pr, is defined as

$$
Pr = \frac{\mu c_p}{k_\infty} \tag{2.7}
$$

where c_p is the specific heat at constant pressure, and k is the coefficient of thermal conductivity. The Prandtl number is indicative of the relative ability of the fluid to diffuse momentum and internal energy by molecular mechanisms.

The internal energy, e_I, and the pressure, p, are given in terms of the other flow variables as

$$
\begin{aligned}
e_I &= \frac{e}{\rho} - 0.5(u^2 + v^2 + w^2) \\
p &= (\gamma - 1)[e - 0.5\rho(u^2 + v^2 + w^2)]
\end{aligned}
\tag{2.8}
$$

In order to nondimensionalize the variables appearing in equations (2.1) through (2.8), the following procedure was followed: the spatial coordinates, (x, y, z), are divided by a reference length, L_{ref}; the velocity is divided by the freestream speed of sound; the density and viscosity are divided by their freestream values; time is divided by L_{ref}/a_∞; and the pressure is normalized by $\rho_\infty a_\infty^2$. Stokes hypothesis is applied, which states that for a gas the coefficient of bulk viscosity, λ, can be related to the coefficient of dynamic viscosity, μ, by the following relationship

$$
\lambda = -\frac{2}{3}\mu \tag{2.9}
$$

For turbulent flows, equation (2.1) can be considered to be the Reynolds-averaged Navier-Stokes equations, where the high frequency fluctuations of the turbulent flowfield are time averaged. For turbulent flows, a turbulence model must be used to specify the coefficients of viscosity and heat conductivity which appear in the viscous terms in equation (2.6). This will be further discussed in Section 2.3. The derivation of the Navier-Stokes equations presented here

is for the three-dimensional formulation only, as the two-dimensional system is an obvious subset of the three-dimensional system.

2.2 Coordinate Transformation

In order to apply the numerical algorithm and boundary conditions easily, the governing equations which are developed in the physical domain or Cartesian coordinates, (x, y, z), must be transformed to the computational domain or generalized coordinates, (ξ, η, ζ), as seen in Figure 2.1 [10]. In this study, ξ, η, and ζ are the coordinates in the axial, circumferential, and radial

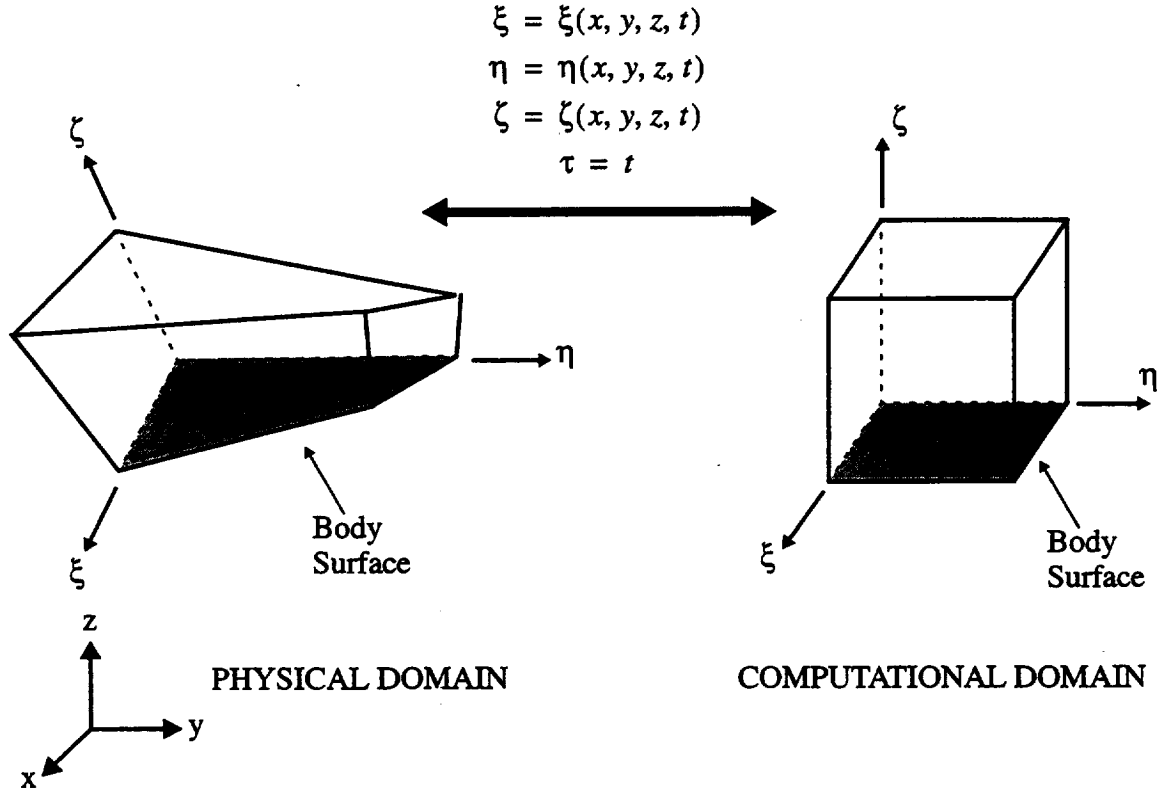


Figure 2.1 Generalized transformation from the physical to the computational domain.

directions, respectively. The general transformation is of the form

$$\begin{aligned} \xi &= \xi(x, y, z, t) \\ \eta &= \eta(x, y, z, t) \\ \zeta &= \zeta(x, y, z, t) \\ \tau &= t \end{aligned} \tag{2.10}$$

and the inverse of the transformation is

$$
\begin{aligned}
x &= x(\xi, \eta, \zeta, \tau) \\
y &= y(\xi, \eta, \zeta, \tau) \\
z &= z(\xi, \eta, \zeta, \tau) \\
t &= \tau
\end{aligned}
\tag{2.11}
$$

The transformation brings the body surface onto one computational plane ($\zeta = 1$). The computational domain is chosen to have equal spacing ($\Delta\xi = \Delta\eta = \Delta\zeta = 1$) to simplify the differencing. By using the chain rule of partial differentiation, the partial derivatives in the physical domain become

$$
\begin{aligned}
\frac{\partial}{\partial x} &= \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} + \zeta_x \frac{\partial}{\partial \zeta} \\
\frac{\partial}{\partial y} &= \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta} + \zeta_y \frac{\partial}{\partial \zeta} \\
\frac{\partial}{\partial z} &= \xi_z \frac{\partial}{\partial \xi} + \eta_z \frac{\partial}{\partial \eta} + \zeta_z \frac{\partial}{\partial \zeta} \\
\frac{\partial}{\partial t} &= \xi_t \frac{\partial}{\partial \xi} + \eta_t \frac{\partial}{\partial \eta} + \zeta_t \frac{\partial}{\partial \zeta} + \frac{\partial}{\partial \tau}
\end{aligned}
\tag{2.12}
$$

where $\tau_t = 1$ and the metrics τ_x, τ_y, and τ_z are equal to zero. The metrics (ξ_x, η_x, ζ_x, ξ_y, η_y, ζ_y, ξ_z, η_z, ζ_z, ξ_t, η_t, ζ_t) that appear in equations (2.12) are obtained in the following manner. The differential expressions are

$$
\begin{aligned}
d\xi &= \xi_x dx + \xi_y dy + \xi_z dz + \xi_t dt \\
d\eta &= \eta_x dx + \eta_y dy + \eta_z dz + \eta_t dt \\
d\zeta &= \zeta_x dx + \zeta_y dy + \zeta_z dz + \zeta_t dt \\
d\tau &= dt
\end{aligned}
\tag{2.13}
$$

which can be written in matrix form as

$$
\begin{bmatrix} d\xi \\ d\eta \\ d\zeta \\ d\tau \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y & \xi_z & \xi_t \\ \eta_x & \eta_y & \eta_z & \eta_t \\ \zeta_x & \zeta_y & \zeta_z & \zeta_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \\ dt \end{bmatrix}
\tag{2.14}
$$

Similarly,

$$\begin{bmatrix} dx \\ dy \\ dz \\ dt \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta & x_\tau \\ y_\xi & y_\eta & y_\zeta & y_\tau \\ z_\xi & z_\eta & z_\zeta & z_\tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \\ d\zeta \\ d\tau \end{bmatrix} \tag{2.15}$$

Therefore

$$\begin{bmatrix} \xi_x & \xi_y & \xi_z & \xi_t \\ \eta_x & \eta_y & \eta_z & \eta_t \\ \zeta_x & \zeta_y & \zeta_z & \zeta_t \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta & x_\tau \\ y_\xi & y_\eta & y_\zeta & y_\tau \\ z_\xi & z_\eta & z_\zeta & z_\tau \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \tag{2.16}$$

Thus, the transformation metrics are

$$\begin{aligned} \xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta) \\ \xi_y &= -J(x_\eta z_\zeta - x_\zeta z_\eta) \\ \xi_z &= J(x_\eta y_\zeta - x_\zeta y_\eta) \\ \eta_x &= -J(y_\xi z_\zeta - y_\zeta z_\xi) \\ \eta_y &= J(x_\xi z_\zeta - x_\zeta z_\xi) \\ \eta_z &= -J(x_\xi y_\zeta - x_\zeta y_\xi) \\ \zeta_x &= J(y_\xi z_\eta - y_\eta z_\xi) \\ \zeta_y &= -J(x_\xi z_\eta - x_\eta z_\xi) \\ \zeta_z &= J(x_\xi y_\eta - x_\eta y_\xi) \\ \xi_t &= -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z \\ \eta_t &= -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z \\ \zeta_t &= -x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z \end{aligned} \tag{2.17}$$

where J is the Jacobian of the transformation, defined as

$$J = \frac{\partial(\xi, \eta, \zeta, \tau)}{\partial(x, y, z, t)} = \begin{vmatrix} \xi_x & \xi_y & \xi_z & \xi_t \\ \eta_x & \eta_y & \eta_z & \eta_t \\ \zeta_x & \zeta_y & \zeta_z & \zeta_t \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{2.18}$$

This may be simplified to

$$J = \frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)} = \begin{vmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{vmatrix} \tag{2.19}$$

which can be evaluated in the following manner

$$J = \frac{1}{J^{-1}} = \frac{1}{\dfrac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)}} = \begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix}^{-1} \tag{2.20}$$

$$= [x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi)]^{-1}$$

The metrics can be determined by using a finite difference scheme in the computational domain.

Applying this generalized transformation to the Navier-Stokes equations (2.1), the following transformed equations are obtained

$$\frac{\partial \hat{\mathbf{Q}}}{\partial \tau} + \frac{\partial \hat{\mathbf{E}}}{\partial \xi} + \frac{\partial \hat{\mathbf{F}}}{\partial \eta} + \frac{\partial \hat{\mathbf{G}}}{\partial \zeta} = \frac{1}{Re}\left(\frac{\partial \hat{\mathbf{E}}_v}{\partial \xi} + \frac{\partial \hat{\mathbf{F}}_v}{\partial \eta} + \frac{\partial \hat{\mathbf{G}}_v}{\partial \zeta}\right) \tag{2.21}$$

where the inviscid flux terms are

$$\hat{\mathbf{Q}} = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix} \qquad \hat{\mathbf{E}} = J^{-1}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ (e+p)U - \xi_t p \end{bmatrix}$$

$$\hat{\mathbf{F}} = J^{-1}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ (e+p)V - \eta_t p \end{bmatrix} \qquad \hat{\mathbf{G}} = J^{-1}\begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ (e+p)W - \zeta_t p \end{bmatrix} \tag{2.22}$$

while the viscous flux terms are given by

$$\hat{\mathbf{E}}_v = J^{-1}\begin{bmatrix} 0 \\ \xi_x\tau_{xx} + \xi_y\tau_{xy} + \xi_z\tau_{xz} \\ \xi_x\tau_{yx} + \xi_y\tau_{yy} + \xi_z\tau_{yz} \\ \xi_x\tau_{zx} + \xi_y\tau_{zy} + \xi_z\tau_{zz} \\ \xi_x\beta_x + \xi_y\beta_y + \xi_z\beta_z \end{bmatrix}$$

$$\hat{\mathbf{F}}_v = J^{-1}\begin{bmatrix} 0 \\ \eta_x\tau_{xx} + \eta_y\tau_{xy} + \eta_z\tau_{xz} \\ \eta_x\tau_{yx} + \eta_y\tau_{yy} + \eta_z\tau_{yz} \\ \eta_x\tau_{zx} + \eta_y\tau_{zy} + \eta_z\tau_{zz} \\ \eta_x\beta_x + \eta_y\beta_y + \eta_z\beta_z \end{bmatrix} \tag{2.23}$$

$$\hat{\mathbf{G}}_v = J^{-1}\begin{bmatrix} 0 \\ \zeta_x\tau_{xx} + \zeta_y\tau_{xy} + \zeta_z\tau_{xz} \\ \zeta_x\tau_{yx} + \zeta_y\tau_{yy} + \zeta_z\tau_{yz} \\ \zeta_x\tau_{zx} + \zeta_y\tau_{zy} + \zeta_z\tau_{zz} \\ \zeta_x\beta_x + \zeta_y\beta_y + \zeta_z\beta_z \end{bmatrix}$$

In equation (2.22) U, V, and W are the contravariant velocity components defined as

$$\begin{aligned} U &= \xi_t + \xi_x u + \xi_y v + \xi_z w \\ V &= \eta_t + \eta_x u + \eta_y v + \eta_z w \\ W &= \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w \end{aligned} \tag{2.24}$$

2.3 Turbulence Modeling

In order to predict turbulent flows solving the Navier-Stokes equations, closure assumptions must be made about the apparent turbulent stress and heat-flux quantities. The Boussinesq approximation, that the apparent turbulent shearing stresses might be related to the rate of mean strain through an apparent scalar turbulent or eddy viscosity, is used.

The prediction of high-lift aerodynamics is currently a difficult challenge for CFD and especially the turbulence modeling. Even in two-dimensions, the flow about a multi-element airfoil is naturally complex. Even though high-lift devices work by manipulating the inviscid flow, viscous effects are important in predicting the flow field [11].

2.3.1 Spalart-Allmaras Turbulence Model

The Spalart-Allmaras turbulence model [12] has been found to be robust enough to run on these complex problems and within numerical assumptions, it currently appears to be the best choice

for this high-lift application [11], [13-16]. The Spalart-Allmaras model is a one-equation model. The Spalart-Allmaras model has a favorable feature that it is "local". The solution at one point does not depend on the solutions of other points [17]. The Spalart-Allmaras turbulence model has the advantage that it does not need as fine grid spacing near the surface of the body as the two-equation models do [13].

The Spalart-Allmaras turbulence model solves one transport equation for a non-linear eddy viscosity variable χ. In the Spalart-Allmaras turbulence model, the eddy viscosity is defined by

$$\nu_t = \nu\chi f_{v1} \tag{2.25}$$

where

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \tag{2.26}$$

The transport equation for χ is given by

$$\begin{aligned} \frac{D\chi}{Dt} &= c_{b1}[1 - f_{t2}]\tilde{S}\chi + \frac{\nu}{\sigma}[\nabla \cdot ((1+\chi)\nabla\chi) + c_{b2}(\nabla\chi)^2] \\ &\quad - \nu\left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\chi}{d}\right)^2 + \frac{f_{t1}}{\nu}\nabla U^2 \end{aligned} \tag{2.27}$$

Here, ∇ is the gradient operator. $\tilde{S}$ and f_{v2} are defined by

$$\begin{aligned} \tilde{S} &\equiv S + \frac{\nu\chi}{\kappa^2 d^2}f_{v2} \\ f_{v2} &= 1 - \frac{\chi}{1 + \chi f_{v1}} \end{aligned} \tag{2.28}$$

where S is the magnitude of the vorticity and d is the distance to the closest wall. The function f_w is given by

$$f_w = g\left[\frac{1 + c_{w3}^3}{g^6 + c_{w3}^6}\right]^{1/6} \tag{2.29}$$

where

$$g = r + c_{w2}(r^6 - r) \tag{2.30}$$

and

$$r \equiv \frac{\nu\chi}{\tilde{S}\kappa^2 d^2} \tag{2.31}$$

The Spalart-Allmaras turbulence model has a sophisticated transition model which provides a smooth laminar to turbulent transition at points specified by the user. In equation (2.27), the transition functions, f_{t1} and f_{t2} are defined as

$$f_{t1} = c_{t1} g_t \exp\left(-c_{t2}\frac{\omega_t^2}{\Delta U^2}[d^2 + g_t^2 d_t^2]\right) \tag{2.32}$$

$$f_{t2} = c_{t3}\exp(-c_{t4}\chi^2) \tag{2.33}$$

$$g_t \equiv min(0.1, \Delta U/\omega_t \Delta x_t) \tag{2.34}$$

Here, the term d_t is the distance from the field point to the trip point on a wall (which is defined by the user; the term ω_t is the wall vorticity at the transition point; ΔU is the difference between the velocity at the field point and at the transition point; and Δx_t is the grid spacing along the wall at the transition location. In this study, the flow is assumed to be fully turbulent and no transition point is specified. Thus, the transition terms defined in equations (2.32) - (2.34) are set to zero.

The different constants and functions that appear in equations (2.25) through (2.34) in the formulation of the Spalart-Allmaras turbulence model are chosen to produce a model which best simulates the turbulent shear flows. In order to calibrate the turbulence model, empirically derived relationships and numerical simulations of many different shear flows are used. The values of these functions and constants are given below.

$$\begin{aligned} \sigma &= 2/3 \qquad \kappa = 0.41 \\ c_{b1} &= 0.1355 \qquad c_{b2} = 0.622 \\ c_{t1} &= 1.0 \qquad c_{t2} = 2.0 \\ c_{t3} &= 1.2 \qquad c_{t4} = 0.5 \\ c_{w1} &= c_{b1}/\kappa^2 + (1 + c_{b2})/\sigma \\ c_{w2} &= 0.3 \qquad c_{w3} = 2.0 \\ c_{v1} &= 7.1 \end{aligned} \tag{2.35}$$

The boundary conditions and initial values for χ must be set before equation (2.27) can be solved numerically. At a no-slip wall boundary, χ is set to zero. At outflow and wall boundaries the normal derivatives of χ is set to zero. The ideal value of χ in the freestream is zero. Typically the initial value of χ at all field points is set to the freestream value. An implicit solution procedure is used to advance equation (2.27) to the next iteration level. At each iteration, the updates of the velocity field from the Navier-Stokes solution algorithm and the turbulent viscosity from the turbulence model are computed in an uncoupled manner.

Chapter 3

Numerical Methods

In this study the algorithm employed to solve the two-dimensional incompressible Navier-Stokes equations is the INS2D-UP code reported by Rogers and Kwak [18], [19]. The INS2D-UP code is robust in obtaining steady-state and time-dependent solutions to the Reynolds-averaged incompressible Navier-Stokes equations. All the computations presented in this study are performed using the steady-state flow option. INS2D-UP uses the approach of artificial compressibility to formulate the equations into a hyperbolic set of partial differential equations. The convective terms are differenced using an upwind biased flux-difference splitting. INS2D-UP uses an implicit line-relaxation scheme to solve the system of equations.

A brief description of the method of artificial compressibility will be presented in this section. Next, a description of the flux-difference splitting scheme used to compute the convective terms and the viscous flux terms will be discussed. A derivation of the linear system of equations that result from the implicit finite difference algorithm will be performed. Then the computational grid generation method will be discussed. Lastly, the boundary conditions that are applied in this study are presented.

3.1 Artificial Compressibility

The two-dimensional incompressible Navier-Stokes equations are a set of mixed elliptic-parabolic partial differential equations (PDEs). In this type of PDE, a disturbance propagates to all points in the flowfield in a single time step. An iterative solution scheme to solve the equations at each time step must be used due to the elliptic nature of the equations. INS2D takes the approach of recasting the incompressible Navier-Stokes equations to a hyperbolic set of PDEs by using the method of artificial compressibility which was first introduced by Chorin [20].

In the method of artificial compressibility, an artificial compressibility term is added to the continuity equation. This term vanishes when the steady-state solution is reached, and thus it continues to satisfy the requirement of the incompressible continuity equation of a divergence-free velocity field. The modified continuity equation is

$$\frac{\partial \tilde{\rho}}{\partial \tau} + \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3.1}$$

Here, $\tilde{\rho}$ is the artificial density and τ is a pseudo-time for incompressible flow. The modified continuity equation and the momentum equation is marched in pseudo-time until a steady-state solution is reached.

The main advantage of adding the artificial compressibility term to the continuity equation is that now the incompressible Navier-Stokes equations are transposed from an elliptic to a hyperbolic set of partial differential equations. The equations can be marched in pseudo-time versus solving them at each iteration. The hyperbolic equations also allow the convective fluxes to be upwind differenced rather than central differenced. For central difference schemes, artificial dissipation needs to be explicitly added to the central differenced convective fluxes to damp out numerical oscillations resulting from the non-linearity of the convective fluxes. The final solution is affected by the amount of artificial dissipation that is added and thus the correct amount needs to be prescribed and adjusted for each simulation. An upwind scheme is a naturally dissipative scheme which damps out the numerical oscillations caused by the nonlinear convective fluxes. Thus by using upwind differencing on the convective terms, many of the problems associated with central differenced convective terms are avoided. Another advantage of upwind differenced convective fluxes is that the scheme is nearly diagonally dominant since it contributes to items on the diagonal of the Jacobian of the residual. The convergence rate of the algorithm used to solve the system of equations is thus improved.

An artificial equation of state is used to relate the artificial density and the pressure as shown below

$$p = \beta\tilde{\rho} \tag{3.2}$$

where β is the artificial compressibility factor and is analogous to the square of the speed of sound in the physical domain. The artificial compressibility factor determines the rate at which waves propagate. Substituting $\tilde{\rho}$ from Equation (3.2) into Equation (3.1) creates the following modified continuity equation

$$\frac{\partial p}{\partial \tau} + \beta\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \tag{3.3}$$

Combining Equation (3.3) with the momentum equations leads to the following set of equations.

$$\frac{\partial \hat{\mathbf{Q}}}{\partial \tau} + \frac{\partial(\hat{\mathbf{E}} - \hat{\mathbf{E}}_v)}{\partial \xi} + \frac{\partial(\hat{\mathbf{F}} - \hat{\mathbf{F}}_v)}{\partial \eta} = 0 \tag{3.4}$$

where the inviscid flux terms are

$$\hat{\mathbf{Q}} = J^{-1}\begin{bmatrix} p \\ u \\ v \end{bmatrix}$$

$$\hat{\mathbf{E}} = J^{-1}\begin{bmatrix} \beta U \\ \xi_x p + uU \\ \xi_y p + vU \end{bmatrix} \tag{3.5}$$

$$\hat{\mathbf{F}} = J^{-1}\begin{bmatrix} \beta V \\ \eta_x p + uV \\ \eta_y p + vV \end{bmatrix}$$

and the viscous flux terms are given by

$$\hat{\mathbf{E}}_v = J^{-1}\begin{bmatrix} 0 \\ \xi_x \tau_{xx} + \xi_y \tau_{xy} \\ \xi_x \tau_{yx} + \xi_y \tau_{yy} \end{bmatrix}$$

$$\hat{\mathbf{F}}_v = J^{-1}\begin{bmatrix} 0 \\ \eta_x \tau_{xx} + \eta_y \tau_{xy} \\ \eta_x \tau_{yx} + \eta_y \tau_{yy} \end{bmatrix} \tag{3.6}$$

the Jacobian of the transformation, J, in two-dimensions is defined to be

$$J = \frac{1}{\begin{vmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{vmatrix}} = \frac{1}{(x_\xi y_\eta - x_\eta y_\xi)} \tag{3.7}$$

Note that for incompressible flow the conservation of energy equation is dropped from the set and only the equations for mass and momentum are considered because the energy equation does not influence the results for velocity and pressure which are the variables that are of interest when solving an incompressible, viscous fluid.

3.2 Finite Differencing

In the finite difference approach, the continuous problem is discretized so that the dependent variables are considered to exist only at discrete points. Derivatives are approximated by differences resulting in an algebraic representation of the partial differential equations. Thus, the problem involving calculus has now been transformed into an algebraic problem. Most finite-difference approximations of derivatives are based on Taylor's series expansion. After the partial derivatives in Equation (3.4) are approximated by finite differences, the governing equations can then be solved numerically. The following sections will discuss the finite difference

approach that is used in the INS2D-UP code. The approach outlined here follows the development by Rogers [21].

3.2.1 Metric Terms

Finite differencing is used to approximate the partial derivatives in the coordinate transformation metrics that are shown in Equation (2.17). In two-dimensions, the metrics terms are simplified to be

$$\begin{aligned} \xi_x &= J y_\eta \qquad \eta_x = -J y_\xi \\ \xi_y &= -J x_\eta \qquad \eta_y = J x_\xi \end{aligned} \tag{3.8}$$

In order to ensure freestream preservation on a stationary grid, the metrics must be evaluated carefully.

The metric terms can be evaluated as defined above if analytic expressions for the inverse of the transformation ($x = x(\xi, \eta)$ and $y = y(\xi, \eta)$) exist. The metric terms are not evaluated directly using finite difference approximations. Instead, the individual values, x_ξ, x_η, y_ξ, and y_η are evaluated using finite difference approximations. The results are then averaged and substituted into Equations (3.7) and (3.8) to obtain the metric terms. The partial derivatives are represented with a central difference approximation with second-order accuracy

$$(x_\xi)_{i,j} = \left(\frac{\partial x}{\partial \xi}\right)_{i,j} = \frac{1}{2\Delta\xi}(x_{i+1,j} - x_{i-1,j}) \tag{3.9}$$

The metric terms are evaluated at each grid point and are now averaged with the following procedure

$$(\eta_y)_{i,j} = \frac{J}{2}[(x_\xi)_{i+1,j} + (x_\xi)_{i-1,j}] \tag{3.10}$$

The other partial derivatives are computed with similar expressions.

3.2.2 Convective Flux Differencing

INS2D-UP uses upwind differencing to follow the propagation of the artificial waves introduced by the artificial compressibility. The upwind scheme provides implicit dissipation which suppresses any oscillations caused by the nonlinear convective terms. Furthermore, the upwind differenced flux vector will contribute terms to the diagonal of the Jacobian of the residual which will make the implicit scheme nearly diagonally dominant and make the numerical code more robust.

Upwind schemes are one-sided difference operators and are stable only for equations with single-signed eigenvalues. Whereas, central difference operators lead to schemes that are simultaneously positive and negative characteristic speeds or eigenvalues. The governing equa-

tions have eigenvalues of mixed signs in the flow regimes studied and thus the flux vectors must be split prior to using the one-sided spatial difference operators [22]. Although it is more costly to use upwind flux differencing than central differencing, there is a significant decrease in the total computer time requirements because of the speed-up in convergence.

The scheme that is used is developed by Roe [23] which is an approximate Riemann solver for the compressible gas dynamics equations. The upwind scheme is derived from one-dimensional theory and is then applied separately to each of the coordinate directions. Flux-difference splitting is used to structure the differencing stencil based on the sign of the eigenvalues of the convective flux Jacobian.

The derivative of the convective flux in the ξ direction is approximated by

$$\frac{\partial \hat{\mathbf{E}}}{\partial \xi} \approx \frac{\tilde{E}_{i+1/2} - \tilde{E}_{i-1/2}}{\Delta \xi} \tag{3.11}$$

where i is the discrete spatial index for the ξ direction and $\tilde{E}_{i+1/2}$ is a numerical flux given by

$$\tilde{E}_{i+1/2} = \frac{1}{2}[\hat{\mathbf{E}}(Q_{i+1}) + \hat{\mathbf{E}}(Q_i) - \phi_{i+1/2}] \tag{3.12}$$

where the dissipation term is denoted by $\phi_{i+1/2}$. If $\phi_{i+1/2} = 0$ then the differencing represents a second-order central-difference scheme. A first-order upwind scheme results when

$$\phi_{i+1/2} = \Delta E^{+}_{i+1/2} - \Delta E^{-}_{i+1/2} \tag{3.13}$$

where $\Delta E^{\pm}$ is the flux difference across positive and negative traveling waves. The flux difference is computed by the following equation

$$\Delta E^{\pm}_{i+1/2} = \mathbf{A}^{\pm}(\bar{\mathbf{Q}})\Delta Q_{i+1/2} \tag{3.14}$$

Here, the Δ operator is

$$\Delta Q_{i+1/2} = Q_{i+1} - Q_i \tag{3.15}$$

The plus Jacobian matrix has only positive eigenvalues whereas, the negative Jacobian matrix has only negative eigenvalues. They are computed from

$$\mathbf{A}^{\pm} = \mathbf{X}\mathbf{\Lambda}^{\pm}\mathbf{X}^{-1} \tag{3.16}$$

where $\mathbf{A}^{\pm}$ and $\mathbf{\Lambda}^{\pm}$ are the matrices of either positive or negative eigenvalues and eigenvectors, respectively. Now that flux vector splitting has been performed, the appropriate one-sided scheme can be used.

3.2.3 Viscous Flux Differencing

The viscous flux terms in Equation (3.6) contain partial derivatives that need to be approximated with finite differencing. A second-order accurate central difference scheme is used in the INS2D-UP code to approximate the derivatives in the viscous flux terms as shown below

$$\left(\frac{\delta\hat{\mathbf{E}}_v}{\partial\xi}\right)_{i,j} = \frac{(\hat{\mathbf{E}}_v)_{i+1,j} - (\hat{\mathbf{E}}_v)_{i-1,j}}{2\Delta\xi}$$
$$\left(\frac{\delta\hat{\mathbf{F}}_v}{\partial\eta}\right)_{i,j} = \frac{(\hat{\mathbf{F}}_v)_{i,j+1} - (\hat{\mathbf{F}}_v)_{i,j-1}}{2\Delta\eta} \tag{3.17}$$

The turbulent viscosity appearing in the viscous flux vectors must be computed at each grid point and at each step in pseudo-time using the turbulence model.

3.2.4 Pseudo-time derivatives

A marching scheme in pseudo-time is used until a steady-state solution is obtained. Since accuracy is not required in pseudo-time, a first-order implicit Euler differencing scheme can be used to represent the partial derivatives of $\hat{\mathbf{Q}}$ with respect to the pseudo-time τ. Using an implicit differencing scheme eliminates the restriction on step size in pseudo-time that the explicit scheme contains to maintain stability. To begin, Equation (3.4) is rewritten as

$$\frac{\partial\hat{\mathbf{Q}}}{\partial\tau} = -\mathbf{R} \tag{3.18}$$

where $\mathbf{R}$ is the residual vector and is equal to

$$\mathbf{R} = \frac{\partial(\hat{\mathbf{E}} - \hat{\mathbf{E}}_v)}{\partial\xi} + \frac{\partial(\hat{\mathbf{F}} - \hat{\mathbf{F}}_v)}{\partial\eta} + \frac{\partial(\hat{\mathbf{G}} - \hat{\mathbf{G}}_v)}{\partial\zeta} \tag{3.19}$$

Next the implicit Euler scheme is applied to Equation (3.18)

$$\frac{\mathbf{Q}^{n+1} - \mathbf{Q}^n}{J\Delta\tau} = -\mathbf{R}^{n+1} \tag{3.20}$$

where n represents the pseudo-time level and

$$\mathbf{Q} = J\hat{\mathbf{Q}} \tag{3.21}$$

To linearize the right-hand-side of the above equation, a Taylor's series expansion is written and truncated after the first two terms. The chain rule for partial differentiation is also used which yields

$$\mathbf{R}^{n+1} = \mathbf{R}^n + \Delta\tau\left(\frac{\partial \mathbf{R}}{\partial \tau}\right)^n = \mathbf{R}^n + \Delta\tau\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^n\left(\frac{\partial \mathbf{Q}}{\partial \tau}\right) \tag{3.22}$$

$$\mathbf{R}^{n+1} = \mathbf{R}^n + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^n(\mathbf{Q}^{n+1} - \mathbf{Q}^n) \tag{3.23}$$

Substituting Equation (3.23) into Equation (3.19) and rearranging yields the following linear system of equations

$$\left[\frac{1}{J\Delta\tau}\mathbf{I} + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^n\right](\mathbf{Q}^{n+1} - \mathbf{Q}^n) = -\mathbf{R}^n \tag{3.24}$$

INS2D-UP provides many different schemes that can be used to solve the linear system of equations shown in Equation (3.24). The implicit method that is used in the present study is the Generalized Minimal Residual (GMRES) method [24]. The convergence of this method is dependent on the eigenvalue distribution of the matrix being solved. The system of equations must be preconditioned to speed up convergence. The Incomplete Lower-Upper (ILU) factorization scheme with zero additional fill is used. Rogers [24] found that the GMRES with the ILU preconditioner outperformed both the point relaxation and line relaxation schemes.

3.3 Computational Grid

The finite-difference approach requires calculations to be made over a collection of discrete grid points. The arrangement of these discrete points throughout the flow field is called the grid. The composite grid around the three-element airfoil is generated by using OVERMAGG [25] which is an automated script system used to perform overset multi-element airfoil grid generation.

3.3.1 Grid Generation Procedure

The automation process that OVERMAGG uses to generate the volume grids is described here. OVERMAGG takes as input the surface definition of the individual elements of the airfoil. Then it creates a surface grid for each individual element by generating and redistributing points from the given surface definition. It calls the HYPGEN code [26] to generate volume grids about each element. The finite difference volume grid is generated in the normal direction of the surface by solving a set of hyperbolic partial differential equations. OVERMAGG also automatically calls the PEGSUS code [27] to unite the individual volume grids into an overset grid system which is the final output of OVERMAGG. Details of HYPGEN and PEGSUS codes can be found in the References cited.

The PEGSUS code is used to implement the Chimera overset grid method. The Chimera overset grid scheme unites the individual grids into a single multi-zone grid. The overset grid method requires only that neighboring grids overlap each other. Points from the main, slat, or flap grids which were generated independently, may fall inside the body boundary of another element. These points must be removed from the calculation. When this happens, a hole is cut

to remove the points. This creates boundary points (or fringe points) at the edge of holes in addition to the existing individual grid outer boundaries. The fringe points are employed in this study for all grids. Flow variables are updated at hole and outer boundary points by tri-linear interpolation. The PEGSUS code identifies hole boundaries and determines the interpolation stencils. Figure 3.1 shows the grid system that is used for numerical prediction of the flow field about the multi-element airfoil.

3.3.2 Grid Sensitivity Study

A grid resolution study is conducted to determine the grid density required to solve the physical flow features. The grid sensitivity study is conducted for a slat setting of δ_s = *6.0*, gap_s = *2.0%c*, ol_s = *-0.05%c* and a flap setting of δ_f = *40.0*, gap_f = *1.45%c*, ol_f = *1.24%c*. Four different grids are used in the computations and the different grid densities are shown in Table 3.1:. The number of grid points were increased on the bodies, wakes, and coves for each element. For the fine grid system, a total of 121,154 grid points are used consisting of a 242 x 81 C-grid around the slat; a 451 x 131 C-grid around the main element; and a 351 x 121 embedded grid around the flap which is used to help resolve the merging wake in this region. The normal wall spacing for all grids is 5 x 10^{-6} chords.

The computed lift coefficient versus angle of attack for each grid system is shown in Figure 3.2. This figure shows that there is not much difference between the solutions generated on the different grid systems. However, if the lift coefficient is plotted against the drag coefficient as shown in Figure 3.3, there are differences in the solutions. Coarse grid computations predict higher drag coefficients than the other calculations. Medium grid computations under predict the drag coefficients when compared to the intermediate and the fine grid solutions. The differ-

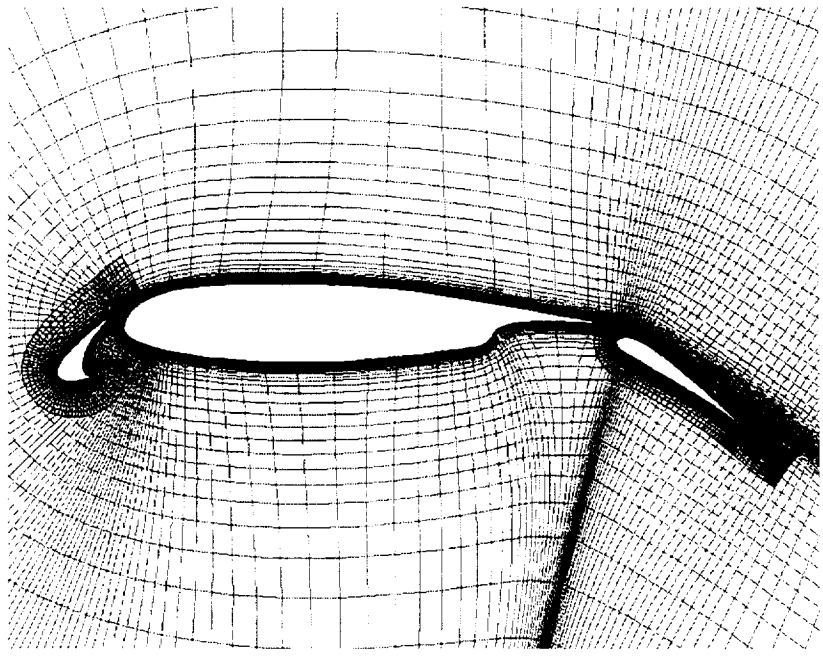


Figure 3.1 Grid around three-element airfoil (every other point shown for clarity).

Table 3.1: Grid dimensions used for grid sensitivity study

Grid Density	Slat		Main		Flap		Total Points
	x	y	x	y	x	y	
Coarse	121	41	401	121	141	51	60,673
Medium	161	61	501	141	311	111	113,153
Intermediate	242	81	401	121	351	121	110,594
Fine	242	81	451	131	351	121	121,154

ence in the grid point distribution between the intermediate and fine grid system is in the main element. It appears from Figure 3.3 that adding the extra points in the main element is beneficial. Similar results were obtained in a grid resolution study conducted to quantify the number of grid points necessary to obtain accurate solutions for a multi-element configuration [13]. The fine grid density is used in the remainder of this study.

3.4 Boundary Conditions

Implicit boundary conditions are used at all of the boundaries, thus allowing the use of large time steps. In this numerical simulation, the boundary conditions applied to all solid surfaces

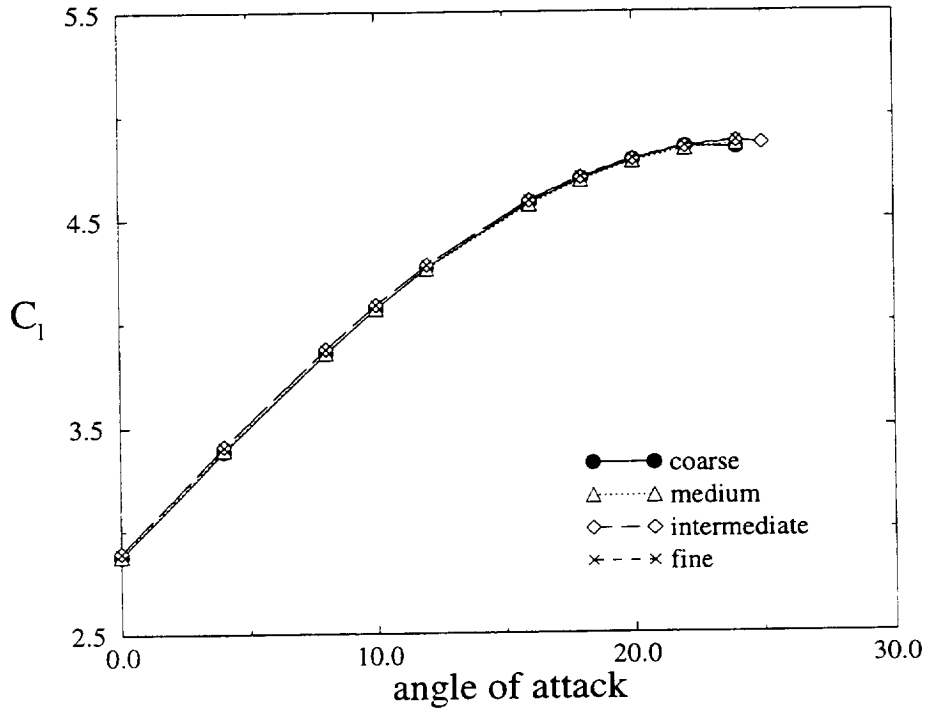


Figure 3.2 Comparison of lift coefficient for grid sensitivity study.

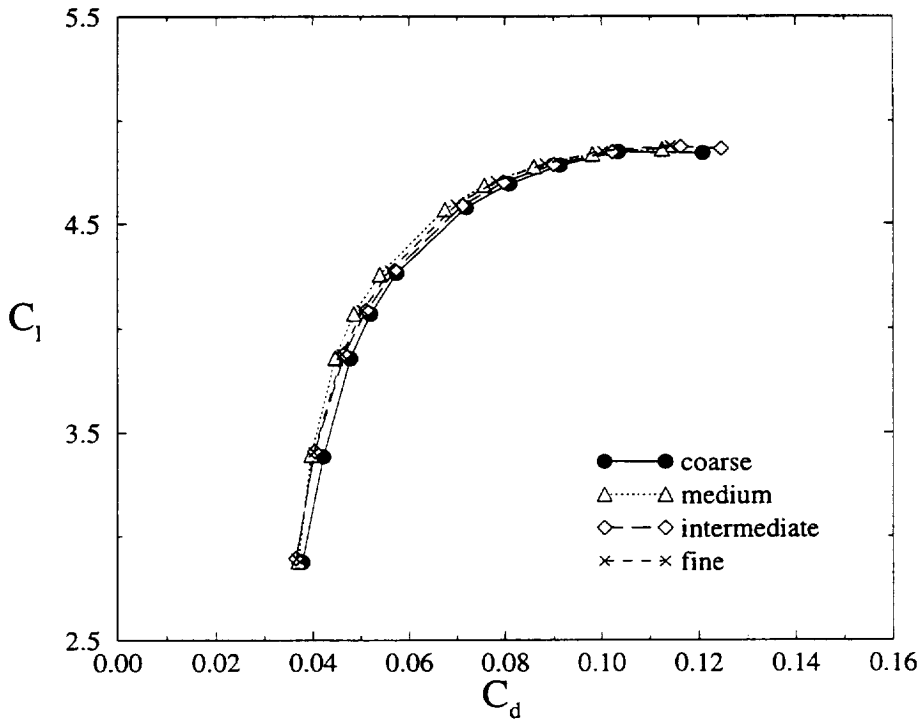


Figure 3.3 Comparison of lift coefficient versus drag coefficient for grid sensitivity study.

are no-slip boundary condition. For a viscous no-slip surface, the velocity is specified to be zero, and the pressure at the boundary is obtained by requiring the pressure gradient normal to the wall to be zero. The boundary conditions used for inflow and outflow regions in INS2D-UP are based on the method of characteristics (refer to Reference 21 for a complete description). The outflow boundary uses extrapolated velocity and constant static pressure. The inflow boundary condition is prescribed with uniform velocity and constant total pressure. Wake-cut boundaries are required since a C-grid topology is used. Wake points are updated by a first order averaging of the points on either side of the wake cut. As mentioned, PEGSUS is used to obtain boundary conditions at grid boundaries that overlap neighboring grids.

Chapter 4

Neural Nets

Although there has been research in developing and applying neural networks in the last fifty years, it is only recently that neural networks have been getting popular for engineering applications. Significant research in neural networks is being conducted in neurobiology, biology, cognitive science, computer science, optics, physics, statistics, and engineering including aerospace engineering.

A discussion of biological neurons is presented in the next section. Next, the artificial neurons are defined and compared to the biological neurons and the artificial neural networks are discussed. A brief history of neural nets is then outlined followed by two current applications of neural networks. In addition, neural network operations are presented. The algorithm that is used in this present study to train the neural nets will be derived including a development of the nonlinear least square optimization of the problem. Lastly, the architecture of the neural networks used in this current study is described.

4.1 Analogy to the Human Brain

Humans and computers are good at completing different tasks. A human can recognize objects, colors, and details, whereas the most powerful computer can not compete with the human performance at this task. Likewise, there are computers that can perform calculations in one second that would take a human 406 years to perform [28]. Computers and humans excel at different tasks because their nervous systems are different. A computer is usually composed of one processor that executes commands in the order written by a programmer. The nervous system of a human being contains over 100 billion (10^{11}) neurons and 10^{14} synapses in the human nervous system [29]. The human brain performs simple computations without the help of a programmer. The human brain neuron's switch time is about a few milliseconds which is about a millionfold times slower than current computer elements, however, it has a thousandfold greater connectivity than today's supercomputers [30].

Neural networks are designed to simulate the human nervous system. They are a collection or network of simple computational devices which are modeled after the architecture of the biological nervous systems. In order to understand the artificial neural nets, the biological nervous system will be briefly described.

The building block of the nervous system and especially the brain is the neuron. A neuron is a microprocessing unit. Each neuron receives and combines signals from many other neurons through structures called dendrites. The dendrite will activate the firing of the neuron if the combined signal is strong enough. The neuron then produces an output signal and the path of the signal is along the axon which is a component of the cell. This simple type of transfer is based on chemical reactions but there are electrical side effects that are measurable [31].

The brain contains over 100 billion neurons densely interconnected. The neurons and the interconnection synapses are the key elements for neural information processing [32]. The axon which is the output path of a neuron splits up and connects to the dendrites which are the input paths of other neurons through a junction referred to as a synapse. Some neurons communicate with only a few other local neurons and on the contrary other neurons communicate with thousands of other neurons which may or may not be closeby. Figure 4.1 shows the basic structure of biological neurons. More precisely, the neuron generates the action potential and propagates this down the branches of axons, where axonal insulators restore and amplify the signal as it propagates until it arrives at a synapse. The transmission across this junction is chemical and the amount of signal that is transferred depends on the amount of neurotransmitters released by the axon and received by the dendrites. This strength referred by the synaptic efficiency is what is modified when the brain learns. The synapse and the processing of information in the neuron produce the basic memory mechanism of the brain.

4.2 Artificial Neural Nets

4.2.1 Basic Structure of Artificial Neurons and Neural Networks

Artificial neural networks simulate human functions such as learning from experience, generalizing from previous to new data, and abstracting essential characteristics from inputs containing irrelevant data [33]. In an artificial neural network, the unit analogous to the biological neuron

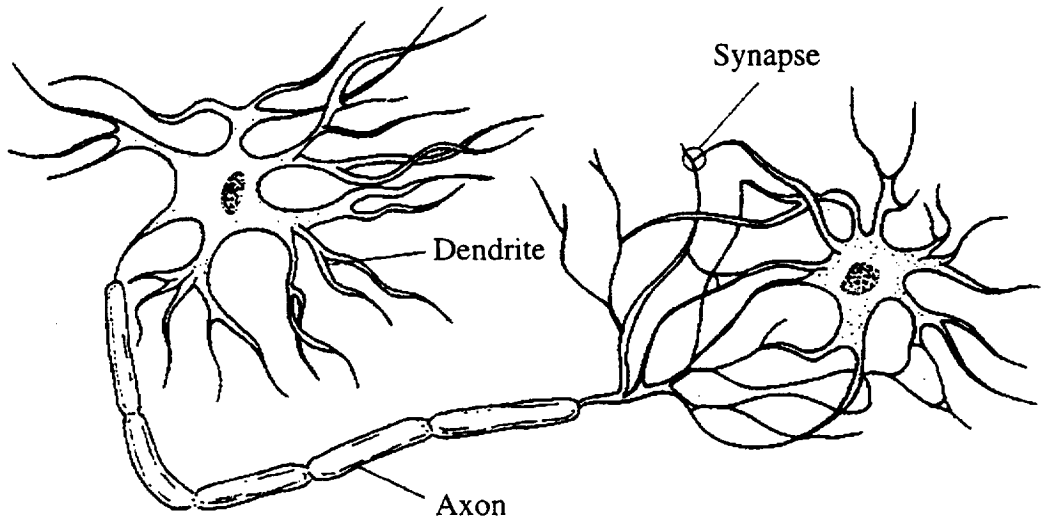


Figure 4.1 Structure of biological neurons [29] (reproduced with permission of Prentice-Hall, Inc., Upper Saddle River, NJ 07458).

is known as a processing element which may be referred to as a PE. A single processing element has many input paths which are comparable to the dendrites in the biological neuron. The processing element combines the values of the input paths by some method. Summations are usually used to combine the input values of the processing element. Once the input values are combined, there is an internal activity level for the PE. The combined input is further modified by a transfer function. One type of transfer function is a threshold function where information only passes along if the combined activity level reaches a specified or given value. A second type of transfer function is a continuous function of the combined input [34]. The output value of the transfer function is generally passed directly to the output paths of the processing element.

The input paths of the processing elements can be connected to output paths of other processing elements through connection weights. The values of the connections weights correspond to the strength of the neural connections. Each connection has a corresponding weight, thus the signals on the input paths to a processing element are modified by the weights before they are combined or summed. Therefore, the summation function is a weighted summation. Figure 4.2 shows a simple artificial neuron model where x_i and w_i are the inputs and connection weights, respectively. In this figure, the analogous biological terms are shown in parentheses.

In the past, neural network research primarily concentrated on simple neural networks with just one layer of output units which where either linear or nonlinear. In the early development of neural networks, researchers understood that these simple neural networks could not accurately predict the complexity of real-world problems. Many researchers such as Widrow and Lehr [35] proposed using more complex neural network architectures in order to overcome the deficiencies of the single-layered neural network. Neural networks became popular after the introduction of multi-layered neural networks [36]. In a multi-layered neural network, there are one or more hidden layers in-between the input layer and output layer. A hidden unit may be

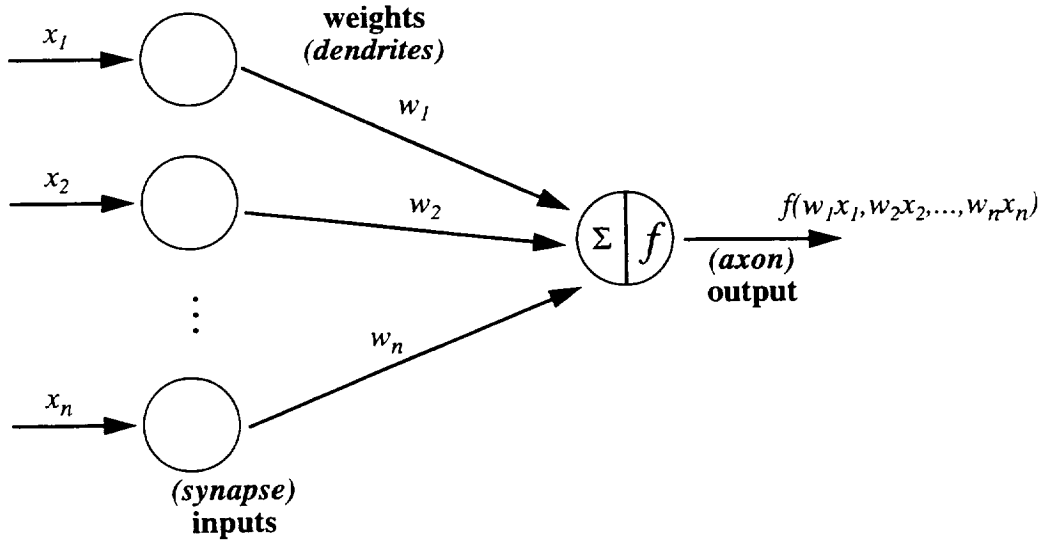


Figure 4.2 Simple artificial neuron model or processing element. The analogous biological neuron terms are shown in parentheses.

connected to an input, output, and/or another hidden unit in a different hidden layer [37]. A fully connected multi-layered neural network with two input neurons, two hidden layers with three neurons in each layer and one output neuron is illustrated in Figure 4.3. In a layered neural network, neurons in every layer are associated with neurons in the previous layer only. Thus, the signal flows from one set of neurons to another set of neurons.

4.2.2 History of Neural Nets

Neural networks can be traced back to neurobiology in the 1800's. For many decades, scientists were interested to find out how the human nervous system and other types of nervous systems function. Researchers were trying to answer how nerves are stimulated, how much current is needed to stimulate nerve cells, and how nerve cells communicate. Not until the mid-twentieth century could scientists hypothesize or answer these questions and many others. Psychologists were also interested in understanding how humans and animals perform certain tasks, such as learning, forgetting, and recognizing. Many psycho-physical experiments helped scientists understand how individual or group of neurons work [30].

A brief outline of the history of neural networks is given here for completeness; further details can be found in the references cited. Rashevsky [38] initiated studies of neurodynamics in 1938. He used differential equations to represent activation and propagation in a neural network. McCulloch and Pitts using binary threshold functions invented the first artificial model of

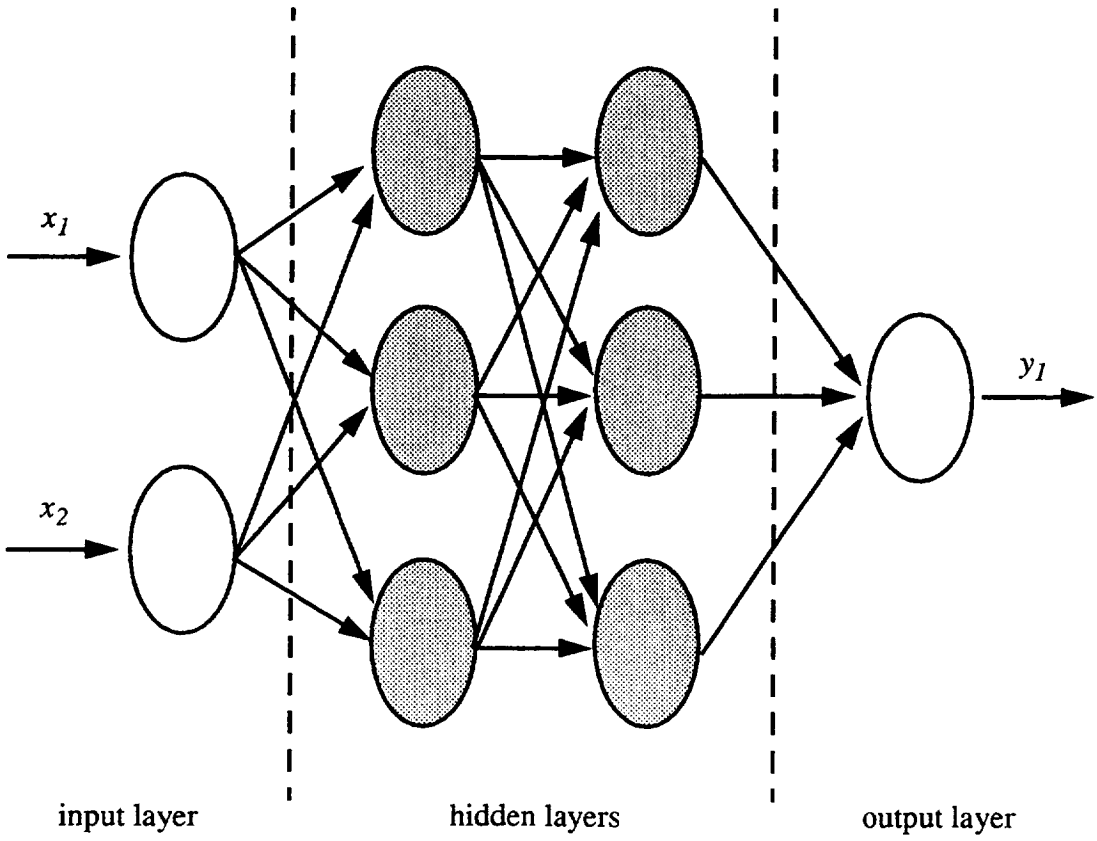


Figure 4.3 A neural network with two hidden layers.

the biological neurons in 1943 [39]. In 1949, Hebb published his very influential book, *The Organization of Behavior* [40], and introduced his famous learning rule which repeated activation of one neuron by another and across a particular synapse. In 1954, Gabor invented the "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 [41]. In 1958, Rosenblatt [42] invented the "perceptron" introducing a learning method for the McCulloch and Pitts [39] neuron model.

Widrow and Hoff [43] in 1960 introduced the Adaline, Adaptive Linear Neuron, which is a simple network trained by a gradient descent rule to minimize the mean squared error. The Adaline and a two-layer neuron called the Madeline, Multiple Adaline, were used for a wide range of applications including speech recognition, character recognition, weather prediction, and adaptive control. Widrow used the adaptive linear element algorithm to create adaptive filters. These filters eliminated echoes and noise on telephone lines. This was the first time that neural computing systems were used to solve a major real-world problem.

Rosenblatt in 1961 proposed the "backpropagation" scheme [44] for training multilayer networks. His attempt was unsuccessful because he used non-differentiable node functions. Marvin Minsky and Seymour Papert from MIT's Research Laboratory of Electronics, began their research on an in depth critique of the perceptron. Misky's and Papert's book, *Perceptron* [45], on the limitations of the simple perceptrons was published in 1969. The work contained a detailed mathematical analysis of an abstract version of Rosenblatt's perceptron. They stated that multi-layer neural networks have the same limitations as single-layer neural networks. This resulted in a drastic reduction in funding and support for neural networks research.

The next two decades led to new research in neural networks where several different types of studies were completed. One of the major accomplishments in this era was the combination of many neurons into neural networks. Also, learning rules applicable to large neural networks which were mostly based on gradient descent were developed. The major contributors were Dryfus in 1962 [46], Bryson and Ho in 1969 [47], Anderson between 1972 and 1977 [48]-[49], Werbos in 1974 [50], Grossberg in 1977 [51], McClelland and Rumelhart in 1986 [52], and Kohonen [53].

Since gradient descent is often not successful in obtaining a desired solution, random, probabilistic, or stochastic methods such as Boltzmann machines were developed by Ackley, Hinton, and Seynowski in 1985 [54]; Kirkpatrick, Gelatt, and Vecchi on 1983 [55]; and many others. Hybrid systems which are a combination of neural networks and non-connectionist components were developed in 1986 by Gallant [56] and again many others contributed to this research.

4.2.3 Current Applications of Neural Networks

Currently, there are many governmental, industrial, and academic research groups performing work in neural networks development and applications. The research is being conducted in a wide range of disciplines. For instance there is current work in neurosciences, cognitive psychology, physics, computer science, mathematics, and engineering. In this section two examples of current applications of neural networks pertinent to aeronautical and aerospace research

are discussed. Several other examples [2-8] were previously discussed in Chapter 1.

First, a simple robot is "taught" the physical characteristics of the human brain with the aid of neural networks [57]. The testbed consists of modified surgical hardware with a robotic multisensor probe, a computer, and neural net software. The probe is equipped with tiny sensors including a pressure sensor. The probe enters the brain carefully and gently locates the edges of tumors while preventing damage to critical arteries. Brain tumors typically have a different density than normal brain tissues. Dr. Robert Mah and his colleagues in their early research successfully used tofu which is a food made from soybeans and with a consistency very similar to brain tissue, to model the brain and to teach the neural nets what a normal brain tissue is. Next, they used a special gel and noodles to simulate brain tissue and blood vessels during their research. In the future, this research will aid surgeons and astronauts to perform surgery in space.

Second, scientists at NASA Ames Research Center and Boeing/McDonnell Douglas Aircraft Corporation are developing neural net software [58] that will allow airplanes that suffer major equipment failures or explosions to be flown and land safely. In a catastrophic problem such as damaged wings, fuselage holes, and sensor failures, the aircraft will handle differently and sometimes the controls will not function properly or at all. Neural net software will allow the aircraft's computer to "relearn" to fly the damaged aircraft correctly in less than one second. Aircraft sensors send velocity, direction, and force data to the computer program. Then the pattern of what is actually occurring is compared with a pattern showing how the aircraft should fly. When there is a mismatch in the patterns, the neural net software, which contains aeronautical stability and control equations, determines the correct pattern that the aircraft should fly under the new conditions and adjusts the how the aircraft should fly.

4.3 Neural Network Operations

In neural networks, there are two main phases in the operation: learning and recall. In most neural networks, these two phases are distinct.

Learning is the first phase and is the process of adapting or modifying the connection weights in response to stimuli from the input layer or optionally the output layer. A stimulus presented at the output layer corresponds to a desired response to a given input. This desired response must be provided by a knowledgeable teacher [31]. This is known as supervised learning. Unsupervised learning is when there is no desired output shown.

Whether supervised or non-supervised learning is used, an essential characteristic of any network is its learning rule. The learning rule determines how weights adapt in response to a learning example. To learn, a network may require many examples to be shown once or many thousands of times. The parameters that govern a learning rule may change over time as the network learns. The long-term control of the learning parameters is referred to as the learning schedule.

The second phase in the network operation is recall. Recall refers to how the network processes a stimulus presented at its input layer and creates a response at the output layer. Often

recall is an integral part of the learning phase. For example, when a desired response of the network must be compared to the actual output of the network to create an error signal.

4.4 Levenberg-Marquardt Algorithm

There are many types of algorithms that can be used to train neural nets. An effective algorithm is the Levenberg-Marquardt algorithm. Levenberg [59] and Marquardt [60] independently developed an elegant algorithm for the numerical solution of finding a local minimum of the non-linear least squares problem. The Levenberg-Marquardt algorithm evolves from a steepest descent algorithm to a quasi-Newton algorithm as optimization proceeds. In the method of steepest descent, the search direction from a point is along the negative gradient direction at the point [61]. In a quasi-Newton method (also referred to as a variable metric method), the Hessian matrix which is the matrix of second partial derivatives and which may be difficult to evaluate, is replaced by a symmetric positive definite matrix which is updated in each iteration without the need for matrix inversion [61]. The Levenberg-Marquardt algorithm has been successfully used to train artificial neural networks [8], [62] - [63]. The Levenberg-Marquardt algorithm is used in this present study to train the artificial neural nets. In order to understand the algorithm, the nonlinear least squares optimization problem will be briefly discussed below, followed by a derivation of the Levenberg-Marquardt algorithm.

4.4.1 Nonlinear Least Squares Optimization Problem

In a neural network, each neuron or processing element produces its output by computing the inner product of its input signal and weight vectors and by passing the result through a nonlinear function.

In training neural networks, the error criteria that is mostly used is the minimization of the sum of squares of the error function. The error function for a network with M outputs and K pairs of input and desired output presented to the network is

$$E_K = \frac{1}{2} \sum_{r=1}^{K} \sum_{m=1}^{M} [d(m) - y(m)]_r^2 \tag{4.1}$$

where the d and y denote the desired and actual outputs of the network, and the subscript r shows their dependence on the rth input presentation.

In a feed-forward network, the input pattern is presented and propagates forward through the network. Each processing element computes its output value using the prespecified set of input weights. For the case of a network consisting of L layers and the lth layer contains N_l processing elements, the Lth layer is the output layer and the zeroth layer is the input layer. Then each PE computes its output according to the following equations for $l = 1, \ldots, L$ and $i = 1, \ldots, N_l$

$$x_i^l = g\left\{ \sum_{n=1}^{N_{l-1}} (w_{ni}^l x_n^{l-1}) + \theta_i^l \right\} \tag{4.2}$$

Here, x_i^l denotes the output of the ith PE belonging to the lth layer and w_{ni}^l is the weight of the connection between the nth PE of the $(l - 1)$ layer and the ith PE of the lth layer. The function g is a nonlinear function, usually sigmoidal. θ_i^l is the threshold of the ith PE of the lth layer. It controls the processing element's output when there are no signals to its input. In the following discussion, the thresholds will be treated as weights that connect an input to the PE and the threshold will always be on, thus its value will be one [62].

Next, a vector, $\boldsymbol{w}_i^l$, will be defined to contain all input weights to the ith PE of the lth layer and the threshold.

$$\boldsymbol{w}_i^l = \left[w_{1i}^l \ldots w_{N_{l-1}i}^l \; \theta_i^l \right]^T \tag{4.3}$$

Likewise, create a vector, $\boldsymbol{x}^l$, which contains the outputs of all PE in the lth layer and an additional element set equal to one.

$$\boldsymbol{x}^l = \left[x_1^l \ldots x_{N_l}^l \; 1 \right]^T \tag{4.4}$$

Now, equation (4.2) can be rewritten as follows

$$x_i^l = g\left\{ (\boldsymbol{w}_i^l)^T (\boldsymbol{x}^{l-1}) \right\} \tag{4.5}$$

Next, define a vector, $\boldsymbol{w}$, consisting of all the weights $\boldsymbol{w}_i^l$ in the network. At a global or local minimum of equation (4.1), the condition that the derivatives of this function with respect to $\boldsymbol{w}$ be zero must be satisfied. Forming these derivatives, the following system of nonlinear equations for $l = 1, \ldots, L$ and $i = 1, \ldots, N_l$

$$\boldsymbol{f}_i^l = \sum_{r=1}^{K} \sum_{m=1}^{M} [d(m) - y(m)]_r^T \left[-\frac{\partial y(m)}{\partial \boldsymbol{w}_i^l} \right]_r = 0 \tag{4.6}$$

where the dimension of vector $\boldsymbol{f}_i^l$ is $(N_{l-1} + 1)$. Now form a global vector $\boldsymbol{F}$, consisting of all vectors $\boldsymbol{f}_i^l$. An iterative technique can be used to solve the system of nonlinear equations shown above in equation (4.6). For example, the Levenberg-Marquardt algorithm is an iterative technique that has been shown to successfully solve nonlinear equations [8], [62] - [63].

4.4.2 Derivation of the Levenberg-Marquardt Algorithm

In the nonlinear least squares problem, the objective function that is being minimized is in the form of a sum of m squared terms

$$f(\mathbf{x}) = \sum_{i=1}^{m} [r_i(\mathbf{x})]^2 = \mathbf{r}^T\mathbf{r} \tag{4.7}$$

where $\mathbf{r} = \mathbf{r}(\mathbf{x})$. In solving this problem, the complicated step of finding the non-positive Hessian matrices that is required in Newton's method is avoided in the Levenberg-Marquardt algorithm.

A quadratic model of the objective function, f, can be obtained from a truncated Taylor Series expansion of $f(\mathbf{x})$ about $(\mathbf{x}^k)$, which can be written as

$$f(\mathbf{x}^k + \boldsymbol{\delta}) \approx q^{(k)}(\boldsymbol{\delta}) = f^{(k)} + \mathbf{g}^{(k)^T}\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T\mathbf{G}^{(k)}\boldsymbol{\delta} \tag{4.8}$$

$q^{(k)}(\boldsymbol{\delta})$ is the resulting quadratic approximation for iteration k. The superscript T denotes the transpose. Here, $\boldsymbol{\delta}$ is the correction to $\mathbf{x}^{(k)}$ defined as

$$\boldsymbol{\delta} = \mathbf{x} - \mathbf{x}^{(k)} \tag{4.9}$$

and $\mathbf{g}(\mathbf{x})$ is the gradient vector

$$\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x}) \tag{4.10}$$

where ∇ is the first derivative operator $(\partial/\partial x_i)$ and $\mathbf{G}(\mathbf{x})$ is the Hessian matrix (second derivative matrix)

$$\mathbf{G}(\mathbf{x}) = \nabla^2 f(\mathbf{x}) \tag{4.11}$$

Next, assume that some neighborhood $\Omega^{(k)}$ of $\mathbf{x}^{(k)}$ is defined where $q^{(k)}(\boldsymbol{\delta})$ agrees with $f(\mathbf{x}^k + \boldsymbol{\delta})$. Then it would be appropriate to choose

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}^{(k)} \tag{4.12}$$

where the correction $\boldsymbol{\delta}^{(k)}$ minimizes $q^{(k)}(\boldsymbol{\delta})$ for all $\mathbf{x}^{(k)} + \boldsymbol{\delta}$ in $\Omega^{(k)}$. This type of method is referred to as a restricted step method since the step is restricted by the region of validity of the Taylor Series. $\Omega^{(k)}$ can not be defined in a general manner. Thus, it is convenient to consider the case

$$\Omega^{(k)} = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^{(k)}\| \leq h^{(k)}\} \tag{4.13}$$

and to find the solution $\boldsymbol{\delta}^{(k)}$ of the resulting subproblem

$$\text{minimize } q^{(k)}(\boldsymbol{\delta}) \text{ subject to } \|\boldsymbol{\delta}\| \leq h^{(k)} \tag{4.14}$$

which can be solved for certain types of norms [64].

When the restricted step method of equation (4.14) is defined in terms of the L_2 norm, the method is characterized by solving a system

$$(\boldsymbol{G}^{(k)} + \nu\mathbf{I})\boldsymbol{\delta}^{(k)} = -\boldsymbol{g}^{(k)}, \quad \nu \geq 0 \tag{4.15}$$

in order to determine the correction $\boldsymbol{\delta}^{(k)}$. Here, $\mathbf{I}$ is the Identity matrix and ν is a scalar. Levenberg in 1944 and Marquardt in 1963 independently developed an algorithm to solve the nonlinear least squares problem by approximating $\mathbf{G}^{(k)}$ by

$$\mathbf{G}(\mathbf{x}) \approx 2\mathbf{A}\mathbf{A}^T \tag{4.16}$$

where

$$\mathbf{A}(\mathbf{x}) = [\nabla\mathbf{r}_1, \nabla\mathbf{r}_2, \ldots, \nabla\mathbf{r}_m] \tag{4.17}$$

is the n x m Jacobian matrix. The columns of $\mathbf{A}$ are the first derivative vectors $\nabla\mathbf{r}_i$ of the components of $\mathbf{r}$ ($A_{ij} = \partial\mathbf{r}_j/\partial x_i$).

The solution of (4.15) using the L_2 norms can be expressed

$$\text{minimize } \boldsymbol{\delta}: \; q^{(k)}(\boldsymbol{\delta}) \equiv \frac{1}{2}\boldsymbol{\delta}^T\mathbf{G}^{(k)}\boldsymbol{\delta} + \mathbf{g}^{(k)T}\boldsymbol{\delta} \tag{4.18}$$

$$\text{subject to } \boldsymbol{\delta}^T\boldsymbol{\delta} \leq h^{(k)^2} \tag{4.19}$$

and it is assumed that $h^{(k)} > 0$. It can be proven that the correction $\boldsymbol{\delta}^{(k)}$ is a global solution of (4.18) if and only if there exists $\nu \geq 0$ such that (4.15) holds. Thus,

$$\nu(h^{(k)^2} - \boldsymbol{\delta}^{(k)^T}\boldsymbol{\delta}^{(k)}) = 0 \tag{4.20}$$

and $\mathbf{G}^{(k)} + \nu\mathbf{I}$ is positive semi-definite. Moreover, if $\mathbf{G}^{(k)} + \nu\mathbf{I}$ is positive definite then $\boldsymbol{\delta}^{(k)}$ is the unique solution of (4.18).

Therefore, the Levenberg-Marquardt algorithm finds a value $\nu \geq 0$ such that $\mathbf{G}^{(k)} + \nu\mathbf{I}$ is

positive definite and solves (4.15) to determine $\boldsymbol{\delta}^{(k)}$. This leads to the following algorithm [64] for the kth iteration:

(i) given $\mathbf{x}^{(k)}$ and $\nu^{(k)}$, calculate $\mathbf{g}^{(k)}$ and $\mathbf{G}^{(k)}$; (4.21)

(ii) factorize $\mathbf{G}^{(k)} + \nu\mathbf{I}$: if not positive definite, reset $\nu^{(k)} = 4\nu^{(k)}$ and repeat;

(iii) solve (4.15) to give $\boldsymbol{\delta}^{(k)}$;

(iv) evaluate $f(\mathbf{x}^{(k)} + \boldsymbol{\delta}^{(k)})$ and hence $\mathbf{r}^{(k)}$;

(v) if $\mathbf{r}^{(k)} < 0.25$ set $\nu^{(k+1)} = 4\nu^{(k)}$

if $\mathbf{r}^{(k)} > 0.75$ set $\nu^{(k+1)} = \nu^{(k)}/2$

otherwise set $\nu^{(k+1)} = \nu^{(k)}$;

(vi) if $\mathbf{r}^{(k)} \leq 0$ set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$ else $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}^{(k)}$

Initially, $\nu^{(1)} > 0$ is chosen arbitrarily. It should be noted that the parameters in step (v) are arbitrary. In the present study, the values that are shown in the above algorithm are used in training the neural networks.

4.5 Architecture of Neural Networks

The architecture of the neural networks in this study is a two-layer network with tangent hyperbolic activation functions in hidden layer units, and a linear transfer function in the output unit [7]. It is found that a two-layer neural network is easier and faster to train than a single-layer network [63]. Individual 4-input, 1-output networks are used to model each of the desired aerodynamic coefficients. A NASA Ames variation of the Levenberg-Marquardt training scheme is used because it provides better accuracy than all schemes tested including the back-propagation training method [63]. The single output networks for each of the aerodynamic coefficients yield more precise modeling than multiple-output networks [4] and [8]. The neural network contains 15 nodes in the hidden layer and Figure 4.4 shows a sketch of the architecture.

The four independent input variables are flap deflection (δ_f), gap, overlap, and angle of attack (α) as illustrated in Figure 4.4. The outputs are lift, drag and moment coefficients (C_l, C_d, and C_m) and the lift-to-drag ratio, L/D. Thus, four different neural networks are used to train and predict each of the outputs.

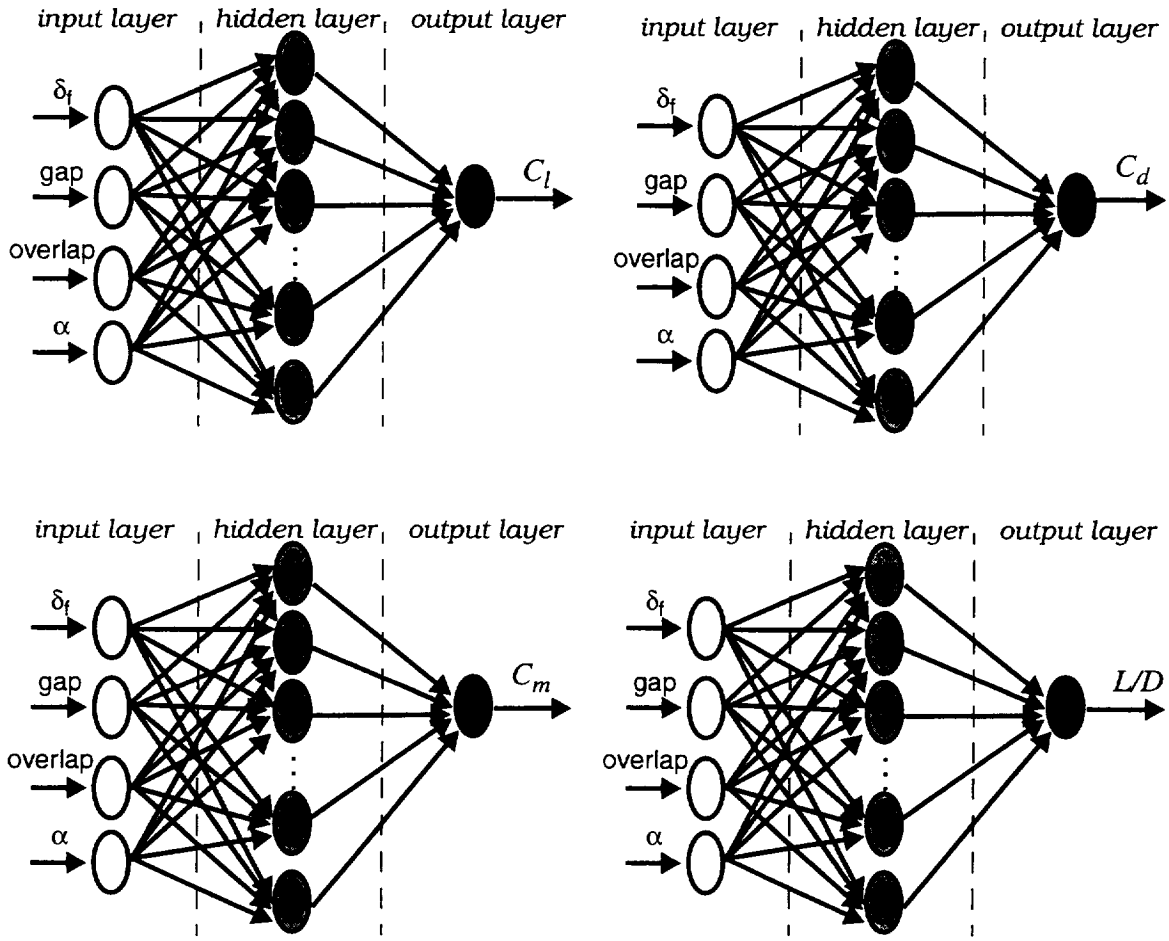


Figure 4.4 Architecture of neural networks with 15 nodes in each hidden layer.

Chapter 5

Optimization

Optimization can be defined as the science of determining the best solutions to certain mathematically defined problems, which are often models of physical reality. Recently, there has been significant increase of interest in the use of formal optimization techniques to improve aeronautical and aerospace vehicles. In order for these optimum designs to be useful, it is necessary to couple advanced and often complex analysis techniques inside the optimization procedure. In this study, computational fluid dynamics and neural networks are used in conjunction with a gradient-based optimizer to improve the high-lift aerodynamics of an airfoil.

A brief description of the different types of optimization methods that are available will be discussed in this chapter. The optimizer which is used in this study will then be presented including a brief description of the algorithm. Lastly, the optimization with neural networks procedure that is applied will be explained.

5.1 Optimization Methods

In the past, there have been many approaches in developing aerodynamic numerical optimization to improve aircraft performance [65]-[66]. Most of these methods involve coupling some type of optimizer with an aerodynamic analysis code. The aerodynamic analysis code can be a computational fluid dynamics flow solver, wind tunnel data analysis tool, or a numerical tool. Most of these approaches can be categorized into two types: direct and indirect numerical optimization methods [67].

In the direct approach, an aerodynamic analysis code is coupled with a numerical optimizer in order to minimize (or maximize) a given aerodynamic objective function. The direct method minimizes a given aerodynamic objective function by iterating directly on the geometry. Often this is accomplished by means of gradient evaluations [68]-[69]. The geometry of the body that is being optimized can be represented by a general function or by parameters that define the body. The desired shape of the body is found by iterating directly on the geometry until a local minimum in the objective function is reached. Two fast methods for performing the gradient evaluations are finite differencing and analytical sensitivity methods. Examples of the analytical methods include the one-shot method [70] and the adjoint method (also called the control theory method) [71]-[72]. The one-shot method uses a multigrid technique to solve for the

unknowns simultaneously, while restricting optimization on a design variable to only grids that produce non-smooth perturbations. On the other hand, the adjoint method solves the adjoint equation of the Navier-Stokes equations in order to obtain the sensitivity direction. Since the analytic methods typically employ modifications to the governing equations in the CFD code, they require recoding of the flow solvers, objective functions, and boundary conditions.

The second class of optimization method is the indirect or inverse approach. The indirect or inverse design method calculates the geometry from the prescribed aerodynamic distribution, usually the pressure distribution. For instance, the target pressure distribution is optimized and the corresponding geometry can be determined by the inverse methods. This process is repeated until the resulting geometry satisfies specific geometry constraints and meets required performance. The quality of the optimized shape depends on how well the distribution is defined. This can lead to problems in translating the design goals into properly defined distributions [73] containing the required aerodynamic characteristics. In addition, the inverse design method does not easily handle geometric constraints.

The direct approach to optimization is used in this study. The flap high-lift settings will be modified until the objective function which is to maximize the lift coefficient is optimum. The neural networks are coupled with an optimizer that uses the finite difference method for the gradient evaluation. An advantage of using the neural networks versus the traditional direct approach is that once the neural networks are trained, no further grid generation and flow solutions are required. In contrast, each time the design variables are modified in the traditional direct methods, a new geometry is required to be configured and the aerodynamic coefficients (required in the objective functions and constraints) to be calculated by whatever numerical tool is used. Further, the design variables, objective function, constraints, and/or initial values can easily be changed resulting in many optimization problems that can be solved with no additional geometry and grid generation, flow solution calculations, or large increase in computer time.

5.2 NPSOL Optimizer

The optimizer that is used in this study is NPSOL [74]. NPSOL is chosen as the optimizer for the following technical reasons: 1) flexibility, especially allowing constraints to be easily coded and applied; 2) ability to handle both linear and non-linear constraints; 3) ability to solve non-linear optimization problem; 4) validated and used by industry; 5) supported by Stanford Department of Operations Research; and 6) simplicity in solving non-linear optimization problem. Further, it was chosen because of past experience [75]. NPSOL is a collection of Fortran subroutines designed to solve the nonlinear programming (NP) problem stated as:

$$\text{minimize } F(\boldsymbol{x})$$

$$\text{subject to } \boldsymbol{l} \le \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{Ax} \\ \boldsymbol{c(x)} \end{bmatrix} \le \boldsymbol{u} \tag{5.1}$$

where $F(\boldsymbol{x})$ is the objective nonlinear function, $\boldsymbol{x}$ is a vector of length n that contains the design

variables, $\boldsymbol{c(x)}$ contains the nonlinear constraint functions, and $\boldsymbol{A}$ is the linear constraint matrix. The variables, $\boldsymbol{l}$ and $\boldsymbol{u}$ are the upper and lower bound vectors that must be specified for each design variable and constraint. The functions $F(x)$ and $\boldsymbol{c(x)}$ are assumed to be smooth which means that they are at least twice-continuously differentiable. The term "programming" is synonymous with "optimization" and was originally used to mean optimization in the sense of optimal planning.

In order to locate the minimum of $\boldsymbol{F(x)}$, the optimizer uses a sequential quadratic programming (SQP) algorithm [74]. The search direction at each iteration is the solution of a quadratic programming problem. Each quadratic programming subproblem is solved by a quasi-Newton algorithm. The optimizer continues this process until it finds a local minimum of $\boldsymbol{F(x)}$. The definition of $\boldsymbol{F(x)}$, $\boldsymbol{A}$, $\boldsymbol{c(x)}$, and their bounds need to be specified as inputs. The initial values of the design variables should also be supplied.

The gradient of the objective function is the vector $\boldsymbol{g(x)}$ defined to be

$$g_j(x) \equiv \partial f(\boldsymbol{x})/\partial x_j \tag{5.2}$$

The gradient of each constraint $c_i(\boldsymbol{x})$ forms the ith row of the Jacobian matrix $\boldsymbol{A(x)}$

$$A_{ij} \equiv \partial c_i(\boldsymbol{x})/\partial x_j \tag{5.3}$$

An important consideration is the difference intervals. It should be noted that NPSOL has an option to calculate the difference interval used in the finite difference approximation of the gradient. However, in this study a common difference interval of 0.002 for all design variables is specified as an input.

5.2.1 Optimization Algorithm

The method of NPSOL is a sequential quadratic programming (SQP) method. NPSOL involves both major and minor iterations. The major iterations generate a sequence of iterates $\{\boldsymbol{x}_k\}$ that converge to x^*, a first-order Kuhn-Tucker point of NP.

A point x is a first-oder Kuhn-Tucker point [64] for NP if the following conditions are true [64]

(i) x is feasible;

(ii) there exist vectors $\boldsymbol{\xi}$ and $\boldsymbol{\lambda}$ (the Lagrange multiplier vectors for the bound and general constraints) such that;

$$\boldsymbol{g} = \boldsymbol{C}^T\boldsymbol{\lambda} + \boldsymbol{\xi} \tag{5.4}$$

where $\xi_j = 0$ if the j-th variable is free

(iii) The Lagrange multiplier corresponding to an inequality constraint that is active at its lower bound must be non-negative, and non-positive for an inequality constraint that is active at its upper bound.

At the major iteration, the new iterate, $\bar{x}$, is defined as

$$\bar{x} = x + \alpha p \tag{5.5}$$

where x is the current iterate, the non-negative scalar α is the step length, and p is the search direction. The search direction p is the solution of a quadratic programming (QP) subproblem of the form

$$\begin{aligned} &\text{minimize} && g^T p + \frac{1}{2} p^T H p \\ &\text{subject to} && \dot{l} \le \begin{Bmatrix} p \\ A_L p \\ A_N p \end{Bmatrix} \le \bar{u} \end{aligned} \tag{5.6}$$

Here, the matrix H is a positive-definite quasi-Newton approximation to the Hessian of the Lagrangian function; and lastly, A_N is the Jacobian matrix of c evaluated at x.

The lower bound vector, l, in NP is partitioned into three sections, l_B, l_L, and l_N, corresponding to the bound, linear, and nonlinear constraints. The vector $\dot{l}$ in Equation (5.6) is similarly partitioned as

$$\begin{aligned} \dot{l}_B &= l_B - x \\ \dot{l}_L &= l_L - x \\ \dot{l}_N &= l_N - x \end{aligned} \tag{5.7}$$

The vector u and $\bar{u}$ are defined in a similar fashion.

In the quadratic programming subproblem, certain matrices are relevant in the major iterations. A "working set" of m_w constraints is identified at each iteration and is updated iteratively until it converges to the optimal QP active set. An important feature of the working-set constraints is that their gradients are linearly independent. This implies that $m_w \le n$. The constraints in the working set can be a simple-bound, linear, or nonlinear. If C denotes the m_w x n matrix of gradients of constraints, then

$$CQ = [0, T] \tag{5.8}$$

where $\boldsymbol{T}$ is a nonsingular m_w x m_w reverse triangular matrix

$$t_{ij} = 0 \text{ if } i > j \tag{5.9}$$

and the nonsingular n x n matrix $\boldsymbol{Q}$ is the product of orthogonal transformations [76]. This comes from the TQ factorization of $\boldsymbol{C}$. Next, the upper-triangular Cholesky factor $\boldsymbol{R}$ of the transformed Hessian matrix is

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{H}_Q = \boldsymbol{Q}^T\bar{\boldsymbol{H}}\boldsymbol{Q} \tag{5.10}$$

where $\bar{\boldsymbol{H}}$ is the approximate Hessian with reordered rows and columns if the columns of $\boldsymbol{Q}$ are partitioned so that

$$\boldsymbol{Q} = \boldsymbol{Z}\boldsymbol{Y} \tag{5.11}$$

The n_z, defined as $n_z \equiv n - m$ columns of $\boldsymbol{Z}$ form a basis for the null space of $\boldsymbol{C}$. The matrix $\boldsymbol{Z}$ is used to compute the reduced gradient $\boldsymbol{Z}^T\boldsymbol{g}$ at the current iterate.

After $\boldsymbol{p}$ has been computed, the major iteration proceeds by determining a step length, α, that produces sufficient decrease in an augmented Lagrangian merit function. Finally, the approximation to the transformed Hessian matrix $\boldsymbol{H}_Q$ is updated using a modified BFGS (Broyden, Fletcher, Goldfarb, and Shanno) [64] quasi-Newton update to incorporate new curvature information obtained in the move from $\boldsymbol{x}$ to $\bar{\boldsymbol{x}}$.

To summarize, NPSOL first determines a point that satisfies both the bounds and constraints. Then, for each iteration a quadratic programming subproblem is solved. This is followed by a line search with an augmented Lagrangian merit function. Lastly, there is a quasi-Newton update of the approximate Hessian of the Lagrangian function. The last three procedures are discussed in greater detailed in the sections below. The following description follows the procedure from Gill et al. (for greater details refer to [74]).

5.2.2 Solution of the Quadratic-Programming Subproblem

The method used to determine the search direction $\boldsymbol{p}$ is a two-phase quadratic programming method. The first phase is finding an initial feasible point by minimizing the sum of infeasibilities and the second phase is minimizing the quadratic objective function within the feasible region. A point is said to be feasible if it satisfies all the constraints in Equation (5.6) and the set of all such points is referred to as the feasible region.

In general, a quadratic problem must be solved in an iterative method. If $\boldsymbol{p}$ denotes the current estimate of the solution of Equation (5.6) then $\bar{\boldsymbol{p}}$ which is the new iterate is defined as

$$\bar{\boldsymbol{p}} = \boldsymbol{p} + \sigma\boldsymbol{d} \tag{5.12}$$

where σ is a nonnegative step length and $\boldsymbol{d}$ is the search direction.

For each iteration, a working set is defined by constraints that are satisfied exactly. The vector $\boldsymbol{d}$ is constructed so that the constraints remain unaltered for all moves along $\boldsymbol{d}$. The matrix $\boldsymbol{C}$ is defined to be the matrix of gradients of constraints in the working set. The constrains will remain unaltered if

$$\boldsymbol{C}\boldsymbol{d} = 0 \tag{5.13}$$

which is equal to

$$\boldsymbol{d} = \boldsymbol{Z}\boldsymbol{d}_z \tag{5.14}$$

for some vector $\boldsymbol{d}_z$. Here $\boldsymbol{Z}$ is the matrix associated with the TQ factorization of $\boldsymbol{C}$. If $\boldsymbol{p}$ is infeasible then $\boldsymbol{d}_z$ is zero except for a component γ in the jth position, where j and γ are chosen to minimize the sum of infeasibilities along $\boldsymbol{d}$. On the other hand, if $\boldsymbol{p}$ is feasible then $\boldsymbol{d}_z$ must satisfy

$$\boldsymbol{R}_z^T\boldsymbol{R}_z\boldsymbol{d}_z = -\boldsymbol{Z}^T\boldsymbol{q} \tag{5.15}$$

where $\boldsymbol{R}_z$ is the Cholesky factor of $\boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z}$. The gradient of the quadratic objective function is $\boldsymbol{q}$ which is defined to be

$$\boldsymbol{q} = \boldsymbol{g} + \boldsymbol{H}\boldsymbol{p} \tag{5.16}$$

With Equation (5.16), $\boldsymbol{p} + \boldsymbol{d}$ is the minimizer of the quadratic objective function subject to treating the constraints in the working set as equalities.

5.2.3 The Merit Function

After computing the search direction as described above, each major iteration proceeds by determining a step length α in Equation (5.5) that produces a sufficient decrease in the augmented Lagrangian merit function

$$\mathrm{L}(\boldsymbol{x}, \lambda, \boldsymbol{s}) = F(\boldsymbol{x}) - \sum_i \lambda_i(c_i(\boldsymbol{x}) - s_i) + \frac{1}{2}\sum_i \rho_i(c_i(\boldsymbol{x}) - s_i)^2 \tag{5.17}$$

where $\boldsymbol{s}$ is the change in $\boldsymbol{x}$

$$\boldsymbol{s} = \bar{\boldsymbol{x}} - \boldsymbol{x} \tag{5.18}$$

Here, $\boldsymbol{x}$, $\boldsymbol{\lambda}$, and $\boldsymbol{s}$ vary during the line search. Only the nonlinear constraints are in the summation terms in Equation (5.17). The vector $\boldsymbol{\lambda}$ is an estimate of the Lagrangian multipliers for the nonlinear constraints of the nonlinear programming problem. The solution of the QP subproblem; Equation (5.6), provides a vector triple that serves as a direction of search for three sets of variables.

5.2.4 Quasi-Newton Update

In Equation (5.6), the matrix $\boldsymbol{H}$ is a quasi-Newton approximation to the Hessian of the Lagrangian function and is positive-definite. A new Hessian approximation, $\bar{\boldsymbol{H}}$, is defined as a rank-two modification of $\boldsymbol{H}$ at the end of a major iteration. In NPSOL, the BFGS quasi-Newton update is used

$$\bar{\boldsymbol{H}} = \boldsymbol{H} - \frac{1}{\boldsymbol{s}^T\boldsymbol{H}\boldsymbol{s}}\boldsymbol{H}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{H} + \frac{1}{\boldsymbol{y}^T\boldsymbol{s}}\boldsymbol{y}\boldsymbol{y}^T \tag{5.19}$$

Here, s is again the change in the new iterate from the old iterate as is defined in Equation (5.18) NPSOL requires the Hessian, $\boldsymbol{H}$, to be a positive definite matrix. If $\boldsymbol{H}$ is positive definite, then $\bar{\boldsymbol{H}}$ will be positive definite if and only if $y^T s$ is positive [77]. In Equation (5.19), y would be set to y_L which is the change in the gradient of the Lagrangian function

$$y_L = \bar{g} - \bar{A}_N^T\mu_N - g + A_N^T\mu_N \tag{5.20}$$

where μ_N denotes the quadratic programming multipliers associated with the nonlinear constraints. NPSOL then makes an attempt to perform the update with a vector y if $y_L^T s$ is not sufficiently positive in the form

$$y = y_L + \sum_{i=1}^{m_N} \omega_i(a_i(\bar{x})c_i(\bar{x}) - a_i(x)c_i(x)) \tag{5.21}$$

where $\omega_i \geq 0$. If no vector can be found that satisfies all the constraints and requirements then a scaled y_L is used to perform the update.

Instead of modifying $\boldsymbol{H}$ itself, the Cholesky factor of the transformed Hessian, $\boldsymbol{H}_Q$ Equation (4) is updated, where $\boldsymbol{Q}$ is the matrix from Equation (5.6) associated with active set of the QP subproblem. The update in Equation (5.19) is equal to the following update to $\boldsymbol{H}_Q$

$$\bar{\boldsymbol{H}}_Q = \boldsymbol{H}_Q - \frac{1}{\boldsymbol{s}_Q^T\boldsymbol{H}_Q\boldsymbol{s}_Q}\boldsymbol{H}_Q\boldsymbol{s}_Q\boldsymbol{s}_Q^T\boldsymbol{H}_Q + \frac{1}{\boldsymbol{y}_Q^T\boldsymbol{s}_Q}\boldsymbol{y}_Q\boldsymbol{y}_Q^T \tag{5.22}$$

where $y_Q = \boldsymbol{Q}^T y$ and $s_Q = \boldsymbol{Q}^T s$.

5.3 Optimization Procedure

The optimization problem in this study is to optimize the high-lift riggings to maximize the lift coefficient. The design variables are flap deflection angle, gap, and overlap and the angle of attack. The objective function which is defined to be the value or function that is being driven to the optimal minimum or maximum is the lift coefficient in this study. The optimizer needs to determine the gradient search direction in order to find the minimum objective function. In this

study, the optimizer is integrated with the trained neural networks so that it can calculate the gradient search direction. In some instances, a constraint is applied to the optimization problem.

The optimization procedure that is used in this study is shown in Figure 5.1. There are three different phases in the optimization procedure: generate the training set, train the neural networks, and optimize. The first two phases need only to be performed once in this process.

The first phase is to generate the training set. In order to train the neural networks to accurately predict the aerodynamic coefficients for a set of inputs, a training data set needs to be created. As discussed in Chapter 4, the training data needs to have different sets of inputs and outputs so that the neural network can learn to predict outputs for a given set of inputs which are not in the training set. The neural networks need to be able to predict within the design space that the optimizer will be searching. Thus, the training set must contain information throughout the design space including the bounds of the design space. An important step is to determine the sufficient number of input/output sets and which sets are required to successfully train the neural networks. The approach that is used to train the neural networks in this study is discussed thoroughly in Chapter 6. For each set of inputs which are the flap deflection, gap, and overlap, a grid is generated as discussed in Section 3.3. For each grid, an angle-of-attack polar is calculated by using INS2D (refer to Chapter 3) to calculate the flow solution. Now, a training data set can be created that includes the various high-lift riggings as the inputs and the aerodynamic coefficients as the outputs. This data set is then used in phase 2 to train the neural networks to accurately predict the aerodynamic coefficients for any set of inputs that is the design space.

The trained neural networks are now used in phase three which is an iterative phase. In this study, only one neural network is actually used because only the lift coefficient is used as the objective function. However, the other aerodynamic coefficients can be easily used as part of the objective function or constraints. The optimization phase begins with the optimizer generating a baseline objective function from the initial values of the design variables. This is accomplished by inputting the initial values of the design variables into the neural network which then will predict the output, that is the lift coefficient, for those sets of design variables. One of the most important advantages of the optimization process is that once the neural networks have been trained, then new designs can be rapidly obtained. Whereas, the traditional optimization process would require function evaluations (CFD simulations) for every new design considered. The next step that the optimizer performs is to perturb each of the design variables to calculate the direction of the gradient using the previously obtained values from the neural net. Figure 5.2 illustrates the details of the optimization process in phase 3. Using the neural network, the optimizer continues to modify the design variables and search for the correct gradient direction until a set of design variables is found with a local minimum objective function which meets all the constraints.

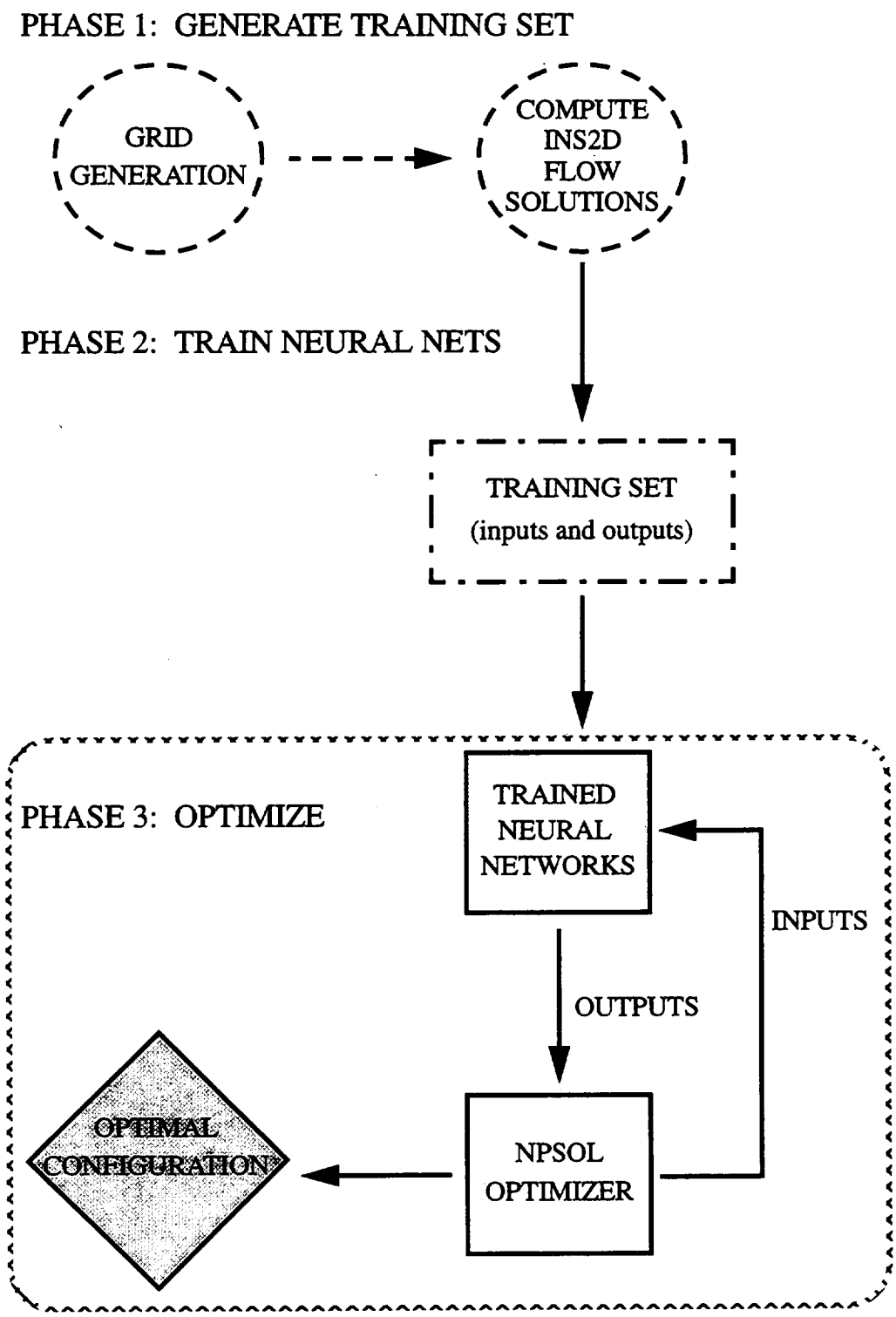


Figure 5.1 The three phases of the neural network optimization procedure.

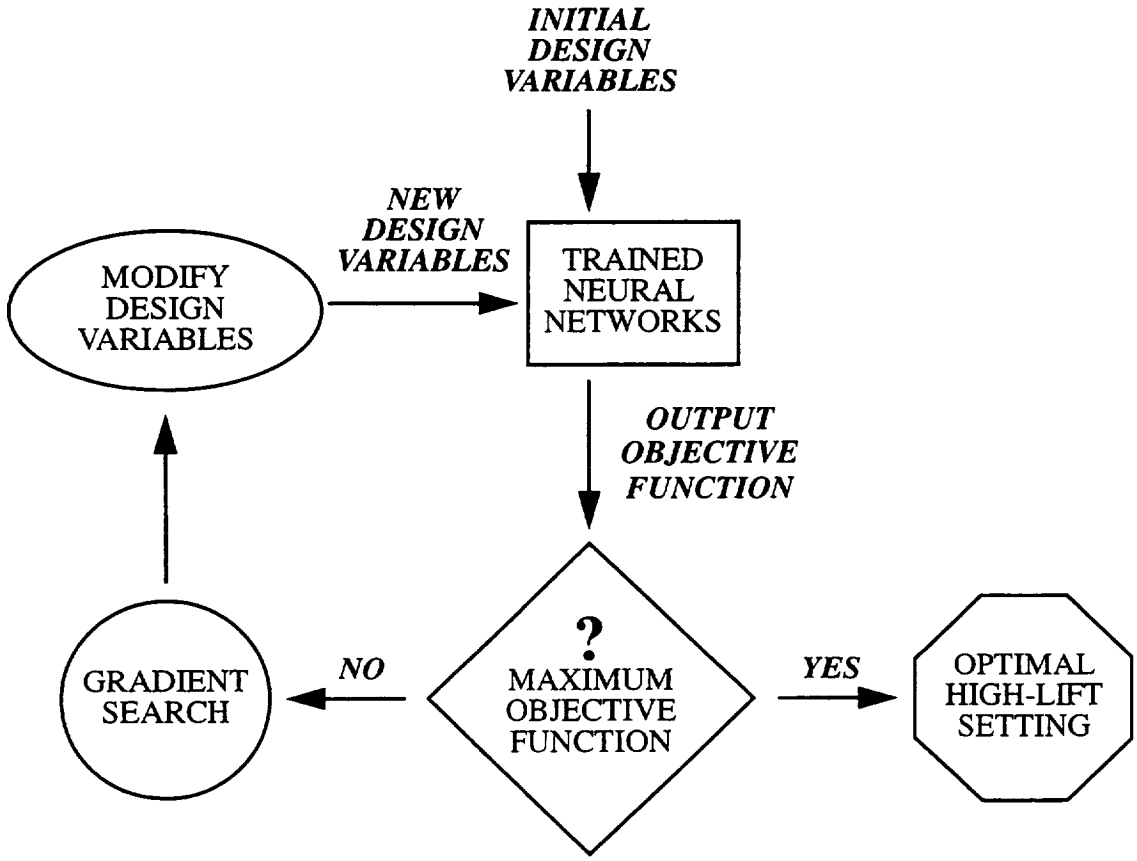


Figure 5.2 Optimization process in phase 3.

Chapter 6

Results and Discussion

As discussed earlier, the overall goal of this research is to develop a method that will allow computational fluid dynamics to impact design. Neural networks are used to create an agile artificial intelligence-enhanced design space capturing method. The neural networks reduce the amount of data that is required to accurately define the aerodynamics of a geometry. The neural networks are trained with both experimental and computational data for the Flap-Edge airfoil in this study. The neural networks which are trained with the computational data set are then integrated with an optimizer to help search the entire design space for certain points of interest, such as extremes or confined subsets of the design space by constraining the search.

The results of this study are presented in the following subsections. The first subsection examines the ability of the neural networks to learn and predict aerodynamic data on an experimental data set. The next subsection discusses the training results for the computational data sets, such as the learning curve of the neural networks and the need for a maximum lift criteria. Next, a discussion on minimizing the amount of data needed for training is presented. The last subsection explains the optimization results including the optimal high-lift riggings and the savings on resources when using neural networks versus the traditional method in the optimization process.

6.1 Experimental Training Set

The Flap-Edge airfoil [9] which was tested in the 7- by 10-Foot Wind Tunnel No. 1 at NASA Ames Research Center will be used to show the validity of the neural networks. Experimental data from the wind tunnel experiment will be used to train the neural networks and to test its accuracy. There is limited wind tunnel data available for the three-element baseline Flap-Edge wing. The data that is available is used to train the neural networks to predict the aerodynamics of the Flap-Edge geometry. The training data consists of five different configurations with eight different angles of attack for each configurations. There are a total of 40 input-output pairs used in the training data set. Four individual neural networks are trained with five-inputs and a single output. The inputs are slat deflection, angle of attack, flap deflection, gap, and overlap. The individual outputs are the lift coefficient for the slat, main, and flap ($C_{l_{slat}}$, $C_{l_{main}}$, and $C_{l_{flap}}$) and the total lift coefficient for the wing ($C_{l_{total}} = C_{l_{slat}} + C_{l_{main}} + C_{l_{flap}}$).

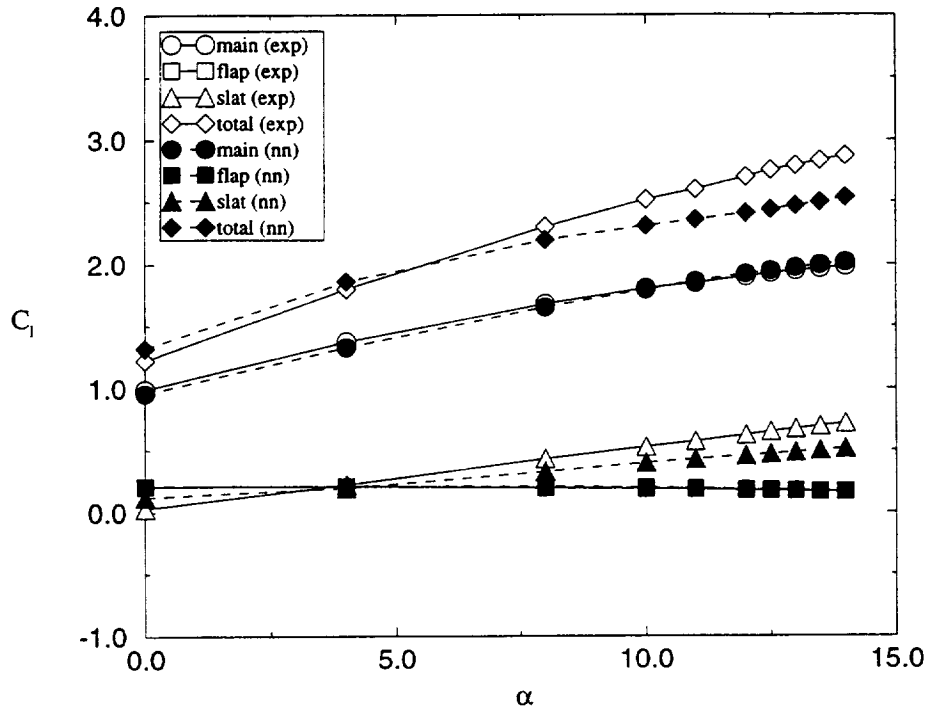


Figure 6.1 Neural network prediction of experimental data for $\delta_s = 6.0°$, $gap_s = 2.0\%c$, $ol_s = -0.05\%c$, $\delta_f = 39.5°$, $gap_f = 2.7\%c$, $ol_f = 1.5\%c$.

The neural networks are trained with the wind-tunnel experimental data for the full-slat configuration. Then the accuracy of the neural networks is tested by predicting the lift coefficients for a configuration that is not included in the training set. The lift coefficients for each element and the total lift coefficient versus angle of attack for $\delta_s = 6.0°$, $gap_s = 2.0\%c$, $ol_s = -0.05\%c$, $\delta_f = 39.5°$, $gap_f = 2.7\%c$, $ol_f = 1.5\%c$ are shown in Figure 6.1. The open symbols represent the experimental data (exp) and the black-filled symbols represent the neural network prediction (nn). The neural networks show good agreement for the main and flap elements, however, there are some noticeable differences in both slope and magnitude in the slat lift coefficient. Since the total lift coefficient is the summation of the individual lift coefficients, as expected the total lift coefficient prediction has noticeable differences as well. This results from the fact that the errors are also summed and amplify in the prediction of the total lift coefficient. The prediction error in $C_{l_{slat}}$ is most likely caused by the sparse training data since there were only five different configurations used to train the neural networks.

6.2 Computational Training Set

All of the computations presented in this study are obtained using the INS2D-UP code in the steady-state mode with the Spalart-Allmaras turbulence model. The flow was treated as fully turbulent for all elements in the computations. The maximum residual in the solution is reduced

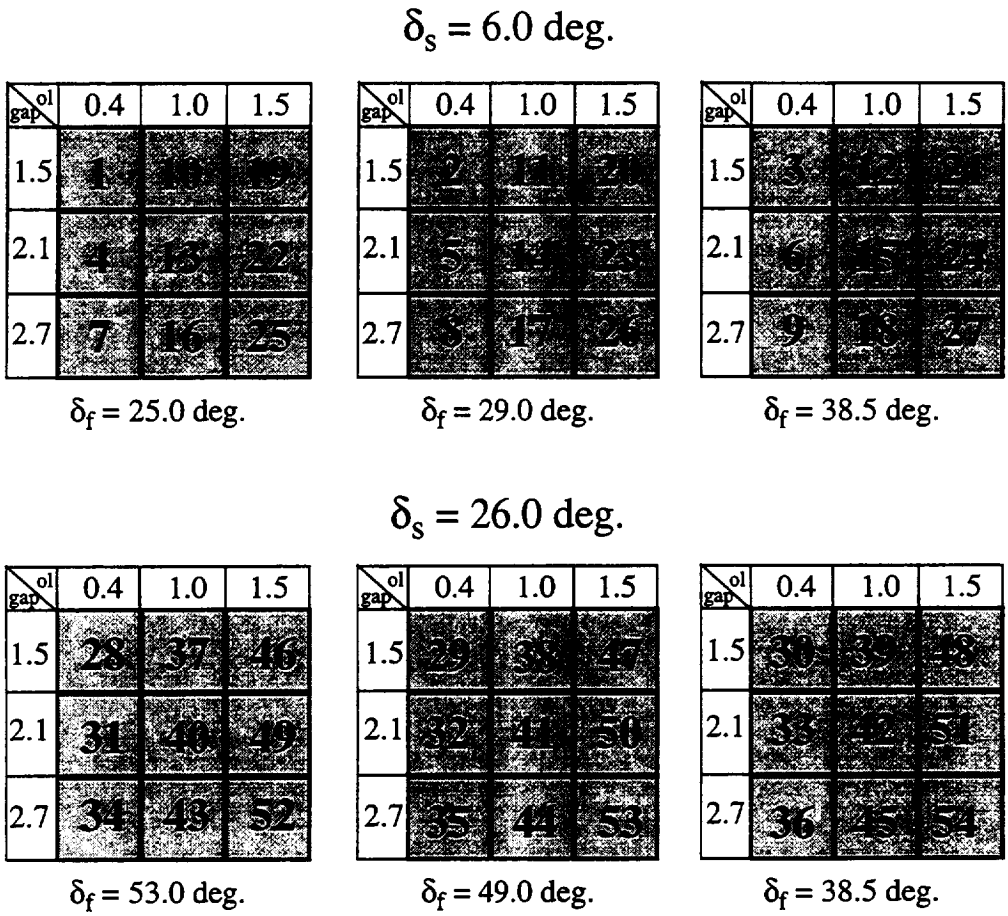


Figure 6.2 Computational cases used to train the neural networks.

by 7 orders of magnitude and the maximum divergence in the converged solution is on the order of 10^{-4} or less. A typical solution for the multi-element airfoil at small angles of attack converged in 200 iterations for a total of 268 seconds (10.6 microseconds/iteration/point) on a CRAY C90 computer. The solutions near maximum lift converged in 800 iterations for a total of 1072 seconds. As the angle of attack increases and approaches the angle where maximum lift occurs, the flow around the airfoil tends to separate from the top surface and create a large wake of separated flow behind the airfoil. Inside this separated region, the flow is recirculating. The flowfield near maximum lift is more complex and thus more time is required for convergence.

The neural networks are trained with computational data consisting of 54 geometric configurations. There are two slat deflection settings (δ_s = 6.0° and δ_s = 26.0°) and each has 27 different flap riggings as shown in Figure 6.2. The computational data is divided into two training sets, one for each slat deflection setting. By only having two slat settings, the data would be represented linearly if the slat setting is used as a training input; this is invalid since the aerodynamic relationship is known to be non-linear. The large numbers in the shaded boxes in Figure 6.2 represent the case number for that particular flap rigging. Cases 1 through 27 are used to train the neural networks for δ_s = 6.0° and likewise, cases 28 through 54 are used to train the δ_s = 26.0° data. Both slat deflections have three flap gap settings of gap_f = *1.5, 2.1,* and

2.7%c and three flap overlap settings of ol_f = *0.4, 1.0,* and *1.5%c*. The flap deflection angles for the six-degree-deflected slat are δ_f = 25.0°, 29.0°, and 38.5°. The twenty-six-degree-deflected slat has flap deflections of δ_f = 38.5°, 49.0°, and 53.0°. Although these flap deflections are higher than what is normally used in flight, they are acceptable for demonstrating the capability of the coupled optimization method. In one optimization study, the additional deflection flaps, δ_f = 49.0° and 53.0°, are added to the training data for δ_s = 6.0° (refer to Section 6.4). The same set of gap and overlap matrix is used. Thus, there are 45 configurations in the training set for this particular optimization run. The range of angle of attack varies from $0.0° < \alpha < 22.0°$ in this study.

6.2.1 Learning Curve

The training data is presented to the neural network for it to learn the relationship between the input and output variables. It is important to know how many times (iterations) to present the data to the neural network. To determine the correct number of iterations that will be used to train the neural networks in this study, the training set for the six-degree-deflected slat is used to train the neural networks with various iterations. The correct number of iterations required to train neural networks is dependent on the inputs and outputs. Thus, a user must determine the required number of iterations for each study performed. The root-mean-square (rms) error of the predicted output and the actual computational value for each aerodynamic coefficient is calculated and is shown in Figure 6.3 for a high-lift setting that was not included in the training set. The case that is used to test the accuracy of the prediction has a flap setting of δ_f = 29.0°, gap_f = *2.1%c,* and ol_f = *1.0%c*. Figure 6.3 shows the rms errors for C_l, C_d, and C_m on the left vertical axis and for *L/D* on the right vertical axis. All four aerodynamic coefficients show the same trend. There is an improvement in the rms error up to a certain iteration number and then there is no further improvement as the iterations to train the neural network increase. For C_d, the rms error continues to drop until 300 iterations and then there is no further improvement in the error. The rms error for C_m drops until about 250 iterations and once again there is no further improvement. The rms error for the lift-to-drag ratio continues to drop until about 450 iterations. In the case for C_l, the rms error drops until about 300 iterations with no additional improvement.

The rms error for the six-degree-deflected slat cases (cases 1 - 27) are calculated to further aid in determining the correct iteration number to use to train the neural networks to predict the aerodynamic coefficients accurately. The average of the rms errors for all twenty-seven cases for each of the different values of the iterations used for training is shown in Figure 6.4. For C_d and C_m, as the iterations increase there is a decrease in rms error until 400 and 300 iterations, respectively. Then the rms errors increase as the iterations increase. For C_l, the rms error is lowest at 250 iterations and then continues to slowly increase as the iterations increase. At 550 iterations, there is a big increase in the rms error. The *L/D* rms error decreases until 450 iterations and then like the other aerodynamic coefficients, the error also increases as the iterations rise.

The neural networks that model the aerodynamic coefficients are overtrained at the higher iterations. This phenomenon has been observed by others [30] where excessive training on the training data sometimes decreases the performance of the networks. The neural networks are starting to memorize the points and go through the points instead of learning the patterns and interpolating. When the data is seen too much by the neural networks, it may lead to spurious

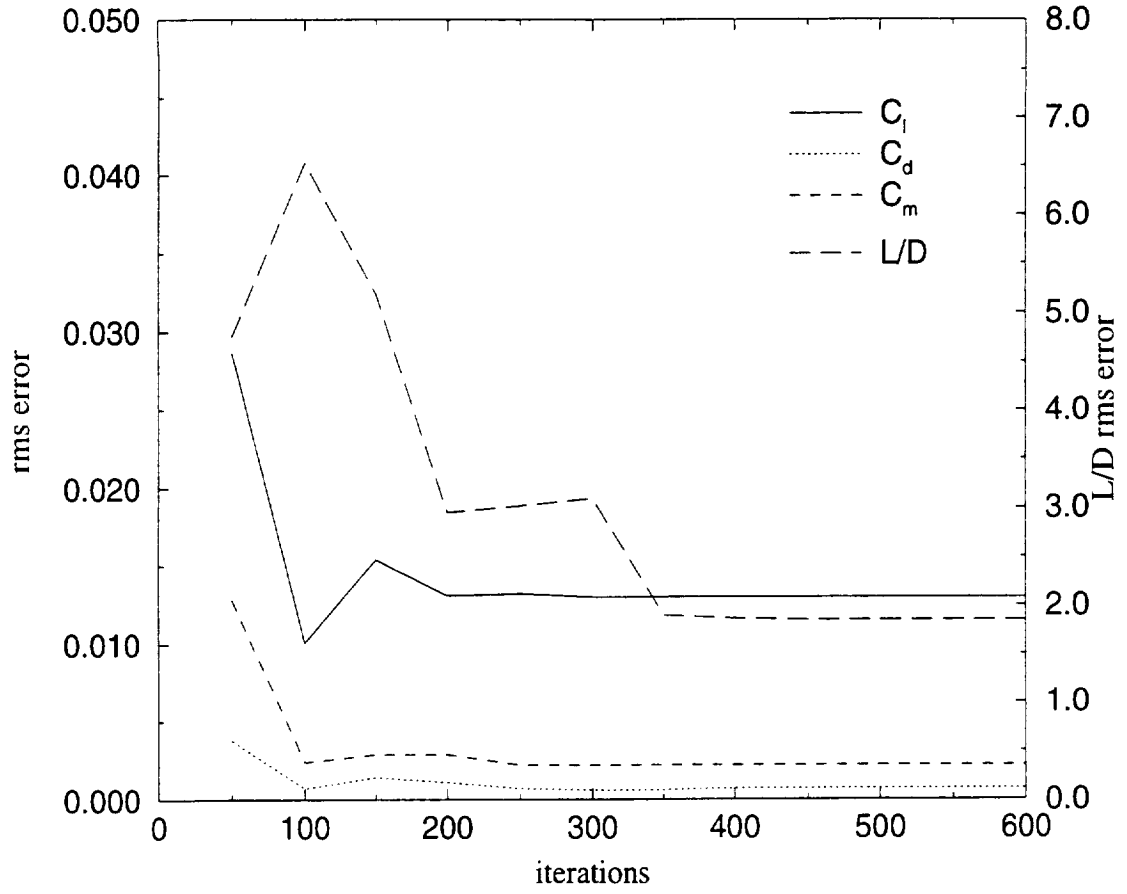


Figure 6.3 Learning curve of the neural networks used to predict the aerodynamics for high-lift setting of δ_s = 6.0°, gap_s = *2.0%c*, ol_s = *-0.05%c*, δ_f = 29.0°, gap_f = *2.1%c*, ol_f = *1.0%c*. This case is not included in the training set.

decision boundaries or cause overfitting of the model and poor interpolation. For these reasons and since these neural networks are going to be used to optimize flap riggings for maximum lift, 250 iterations are used to train the neural networks in this study because C_l showed the best accuracy at this iteration level.

6.2.2 Maximum Lift Criteria

The flowfield about a multi-element airfoil that is designed for high-lift is very complex and is not easily computed by any method that is reported in the literature [11], [78]. Some of the difficult features to capture are trailing viscous wakes whose strength and location vary with angle of attack, merging wakes and boundary layers, different transition phenomena on each of the airfoils elements, boundary-layer separation, and reversed flows in the main element wake. In addition, the airfoil performance varies with Reynolds and Mach number [79], [80]. Lastly, the determination of maximum lift is one of the most important results of any high-lift wing design study. No current fully computational method is able to resolve all these features and accurately predict maximum lift. Thus, an empirically based maximum lift criteria has been implemented.

Figure 6.5 shows a plot for the computed angle-of-attack polar curve for a high-lift setting of δ_s = 6.0°, gap_s = 2.0%c, ol_s = -0.05%c, δ_f = 38.5°, gap_f = 2.7%c, ol_f = 0.4%c. It is shown that the computational curve never bends over (C_l continues to increase with α) and thus does not accurately predict the maximum lift. Previous studies [78] have also shown that state-of-the-art two-dimensional theory can not accurately predict maximum lift. Valarezo and Chin [78] reported a hybrid method that couples cost-effective computational fluid dynamics technology with empirically-observed phenomenon in order to predict maximum lift ($C_{l_{max}}$) for complex multi-element wing geometries. Their semi-empirical $C_{l_{max}}$ criteria for multi-element airfoils or wings, designated the pressure difference rule, is applied to the computational training data set. The pressure difference rule (PDR) states that for a given Reynolds and Mach number combination, there exists a certain difference between the peak suction pressure and the trailing edge pressure at the maximum lift condition

$$\Delta C_{p_{diff}} = C_{p_{peak}} - C_{p_{te}} \tag{6.1}$$

For the flow conditions of this study, the pressure difference value is -13.0. The rule is applied to each element of the airfoil. The slat is the element that has pressure differences greater than the acceptable value, thus this is the critical element in determining the maximum lift for this

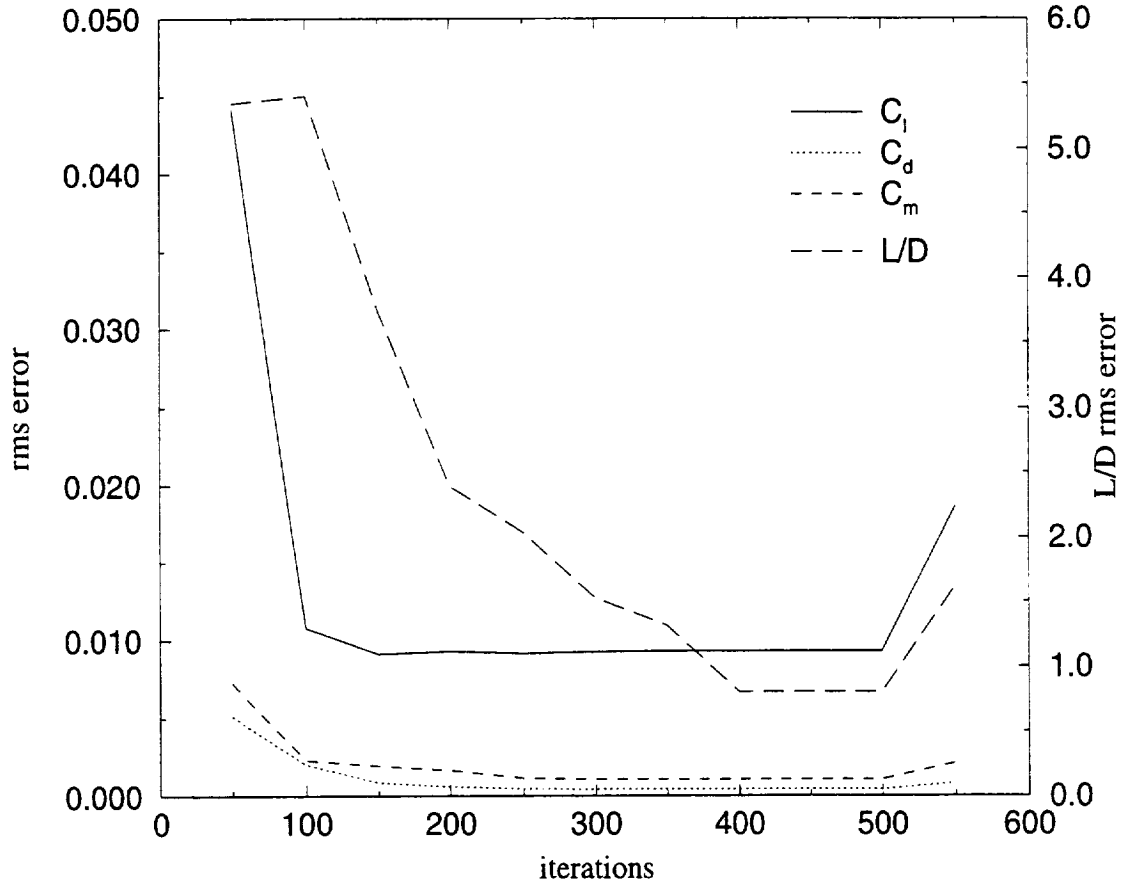


Figure 6.4 Average rms error for cases 1 - 27 for the six-degree slat deflection training set.

configuration.

By applying the pressure difference rule to the training data set, the set is then reduced to include only the data points that are at or below the maximum lift. In order to accurately represent the aerodynamic data, the data points that had a larger angle of attack than where maximum lift is predicted are not used. As Figure 6.5 illustrates for one rigging, the maximum lift occurs near an angle of attack of $\alpha = 12.0°$ for this design space, thus only the data up to and including $\alpha = 12.0°$ is used in the training set for this high-lift rigging. Although this particular configuration was not tested in the wind-tunnel, the angle where maximum lift is predicted is near where the expected experimental value would be by observations of similar configurations.

In order to see if the neural networks would predict the non-linear aerodynamic data better once the pressure difference rule is applied, the neural networks are trained with the entire data set for the six-degree-deflected slat and then also with the processed data set that includes only data up to and containing maximum lift. The comparison of the rms error for each training set is shown in Figure 6.6 for cases one through twenty-seven. For all four outputs, C_l, C_d, C_m, and L/D, the rms error is much lower when the pressure difference rule is applied. As expected, the

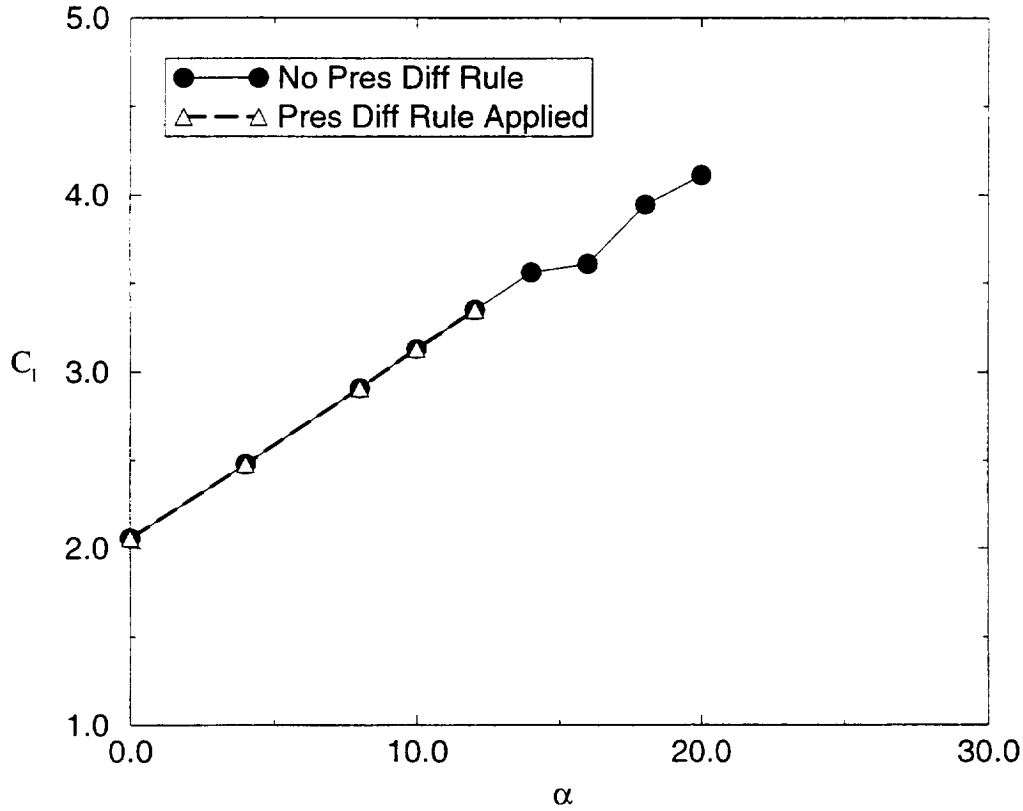
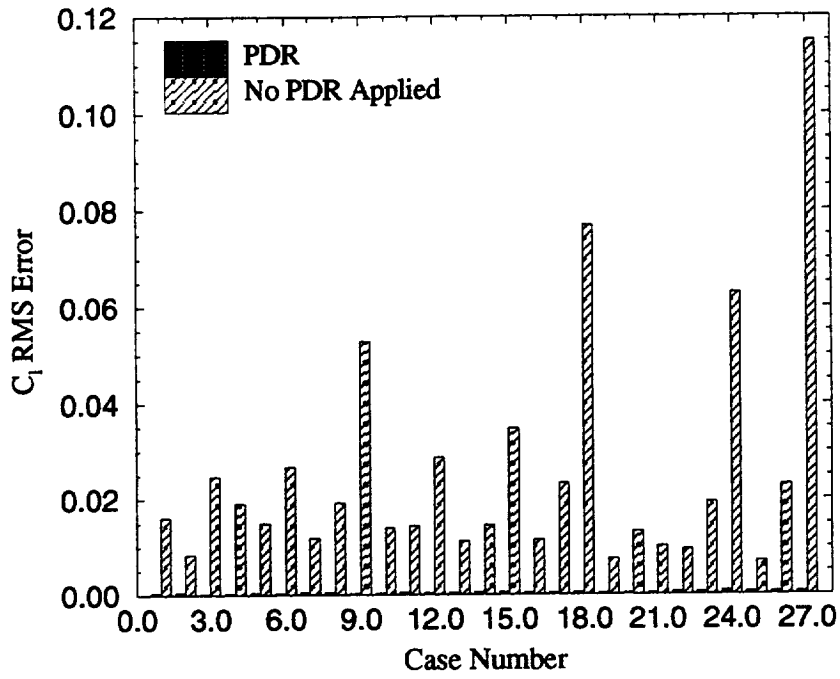
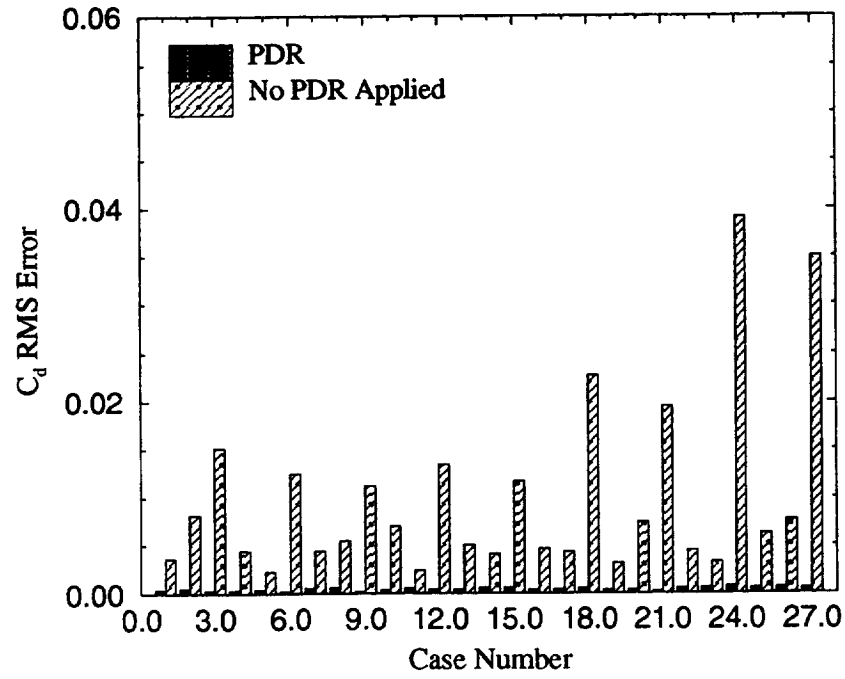


Figure 6.5 Computational lift coefficient versus angle of attack for $\delta_s = 6.0°$, $gap_s = 2.0\%c$, $ol_s = -0.05\%c$, $\delta_f = 38.5°$, $gap_f = 2.7\%c$, $ol_f = 0.4\%c$.

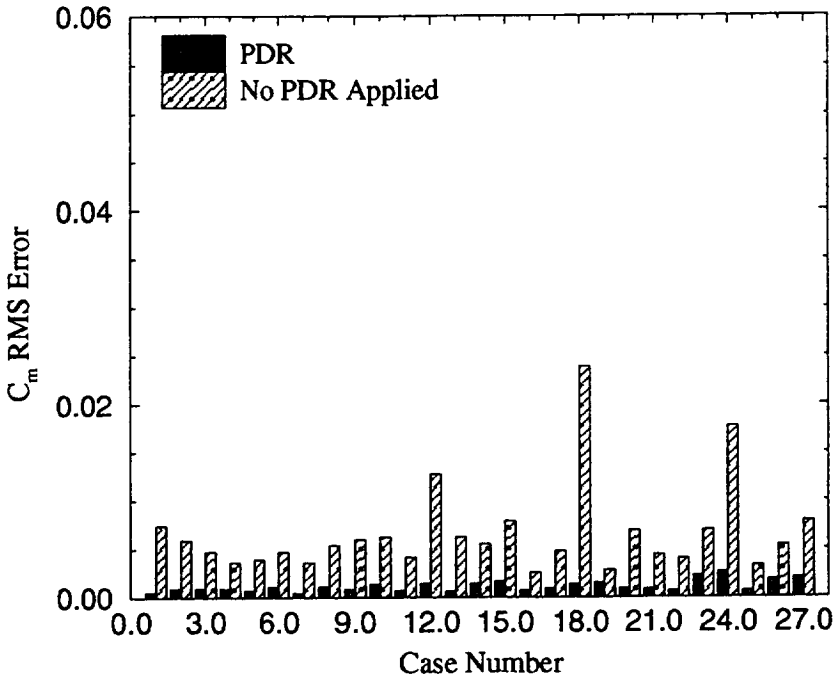


a) C_l rms error

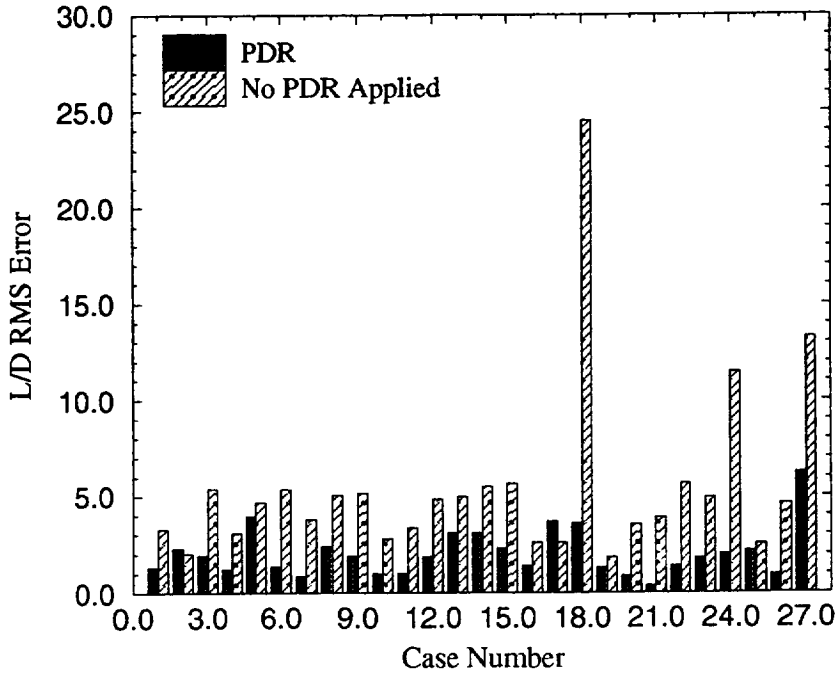


b) C_d rms error

Figure 6.6 Comparison of rms error for maximum lift criteria.



c) C_m rms error



d) *L/D* rms error

Figure 6.6 (continued) Comparison of rms error for maximum lift criteria.

training data is now more accurately representative of the actual aerodynamic data and the neural network is able to predict the aerodynamics more accurately. The lift coefficient versus angle of attack is shown in Figure 6.7 for case 9 which has a high-lift setting of $\delta_s = 6.0°$, $gap_s = 2.0\%c$, $ol_s = -0.05\%c$, $\delta_f = 38.5°$, $gap_f = 2.7\%c$, $ol_f = 0.4\%c$. When the pressure difference rule is not applied to the training data, the neural network prediction is less accurate, particularly at the higher angles of attack, as shown in Figure 6.7a. Here, the black filled circles represent the actual INS2D computational data and the open circles represent the neural networks prediction. Even at $\alpha = 8.0$ and 10.0 degrees, the neural net prediction differs slightly from the computational or actual value. For this case, the rms error for the lift coefficient is bounded by ($0.0070 < C_l$ *rms error* < 0.1149). When the pressure difference rule is applied, however, the neural network's prediction are more accurate and is bounded by ($0.0019 < C_l$ *rms error* < 0.0152). The rms error is decreased by -87% when the pressure difference rule is applied. The neural network does an excellent job at predicting the lift coefficient for all the angles of attack that are within predicted maximum lift as shown in Figure 6.7b.

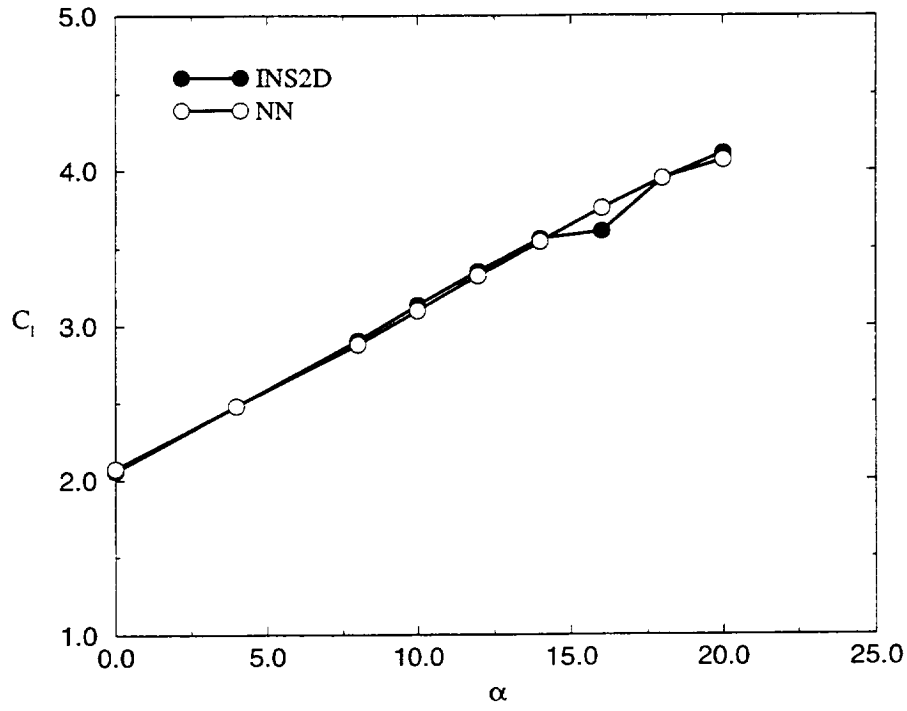
In summary, the computational data in the region beyond the predicted maximum lift seems to be non-physical and is very different for each flap rigging which hinders the ability of the neural networks to learn and to predict the aerodynamics. For the remainder of the study, the neural networks are trained with 250 iterations and with the data set only consisting of data up to and including maximum lift location which is predicted by the pressure difference rule.

6.3 Minimizing Training Data Samples

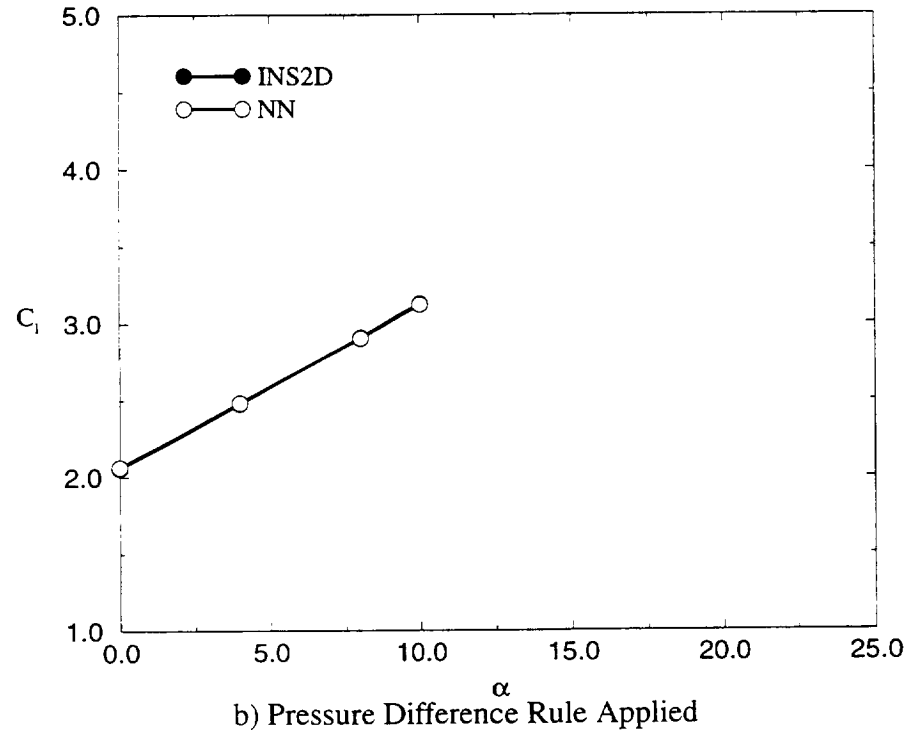
Even though the computational data base that is used for training is sparse, a study is conducted to see how much further the training set can be reduced and still allow the neural networks to predict within the acceptable error. By reducing the training data set further, the required computational resources can be decreased [7].

The six-degree-deflected slat data is used for the majority of the reduction of data study. Several subsets of the computational data are created to train the neural networks and to test the accuracy of the prediction. Each configuration that is generated has its flowfield computed at several angles of attack but not necessarily at the same angles. The number of angles of attack also varies for each configuration. In general, once the grid is generated, it does not acquire extra effort to compute solutions at different angles of attack. The neural networks, are therefore trained using data sets which contain various numbers of configurations which include their individual angle of attack polar. Nine different subsets are created by removing entire configurations from the training set as shown in Figure 6.8. Here, the boxes that are shaded represent the cases that are in the training sets, whereas the numbers in the white boxes and in parentheses are the cases that are omitted from the training sets. The training sets are first created by randomly omitting cases from the entire data set. The results from these sets are then examined which leads to more carefully chosen training sets.

The neural networks are trained with each individual training data set with the flap deflection, gap, and overlap, and angle of attack as the inputs. The outputs for the neural networks are the lift, drag, and moment coefficients and the lift-to-drag ratio. Four identical neural networks are used to model the aerodynamics of the multi-element airfoil. Each network is trained with



a) No Pressure Difference Rule Applied



b) Pressure Difference Rule Applied

Figure 6.7 Neural net prediction with and without the pressure difference rule applied for $\delta_s = 6.0°$, $gap_s = 2.0\%c$, $ol_s = -0.05\%c$, $\delta_f = 38.5°$, $gap_f = 2.7\%c$, $ol_f = 0.4\%c$.

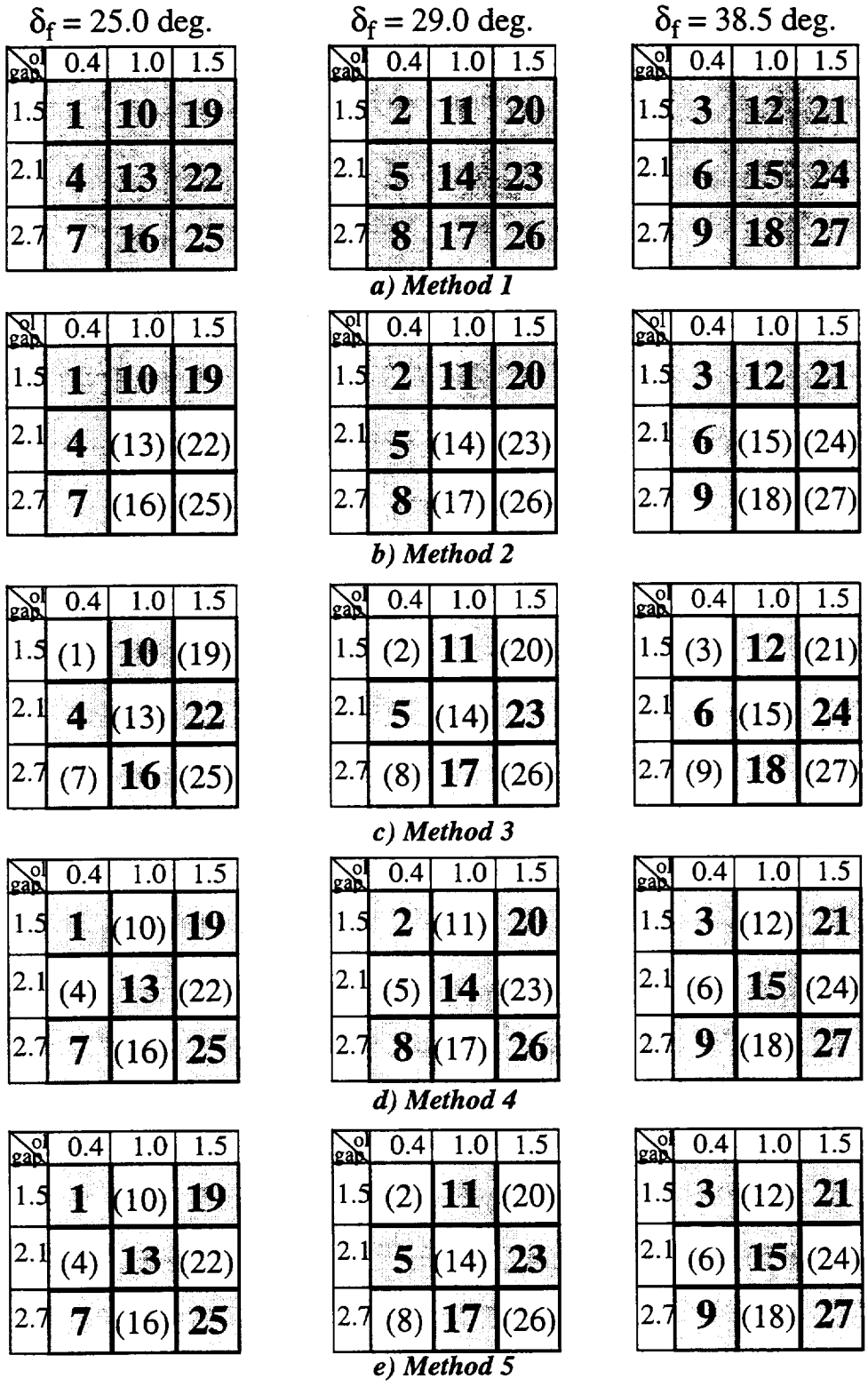


Figure 6.8 Training data subsets $\delta_s = 6°$ (shaded boxes represent cases that are included in the training set whereas the cases in the white boxes and parentheses are omitted).

four inputs and one output. From the previous conclusions, the neural networks are trained with 250 iterations and the pressure difference rule is applied to each training set. Then to test the accuracy of the neural networks ability to predict data, the rms error of the predicted value and the actual computed INS2D value for C_l, C_d, and C_m are calculated for all 27 cases even if they were not included in the training set. This will show the accuracy of each training method to predict cases that and are not included in the training set. The *L/D* rms error is not shown in these plots within the next subsection for clarity since the values are on a different scale. How-

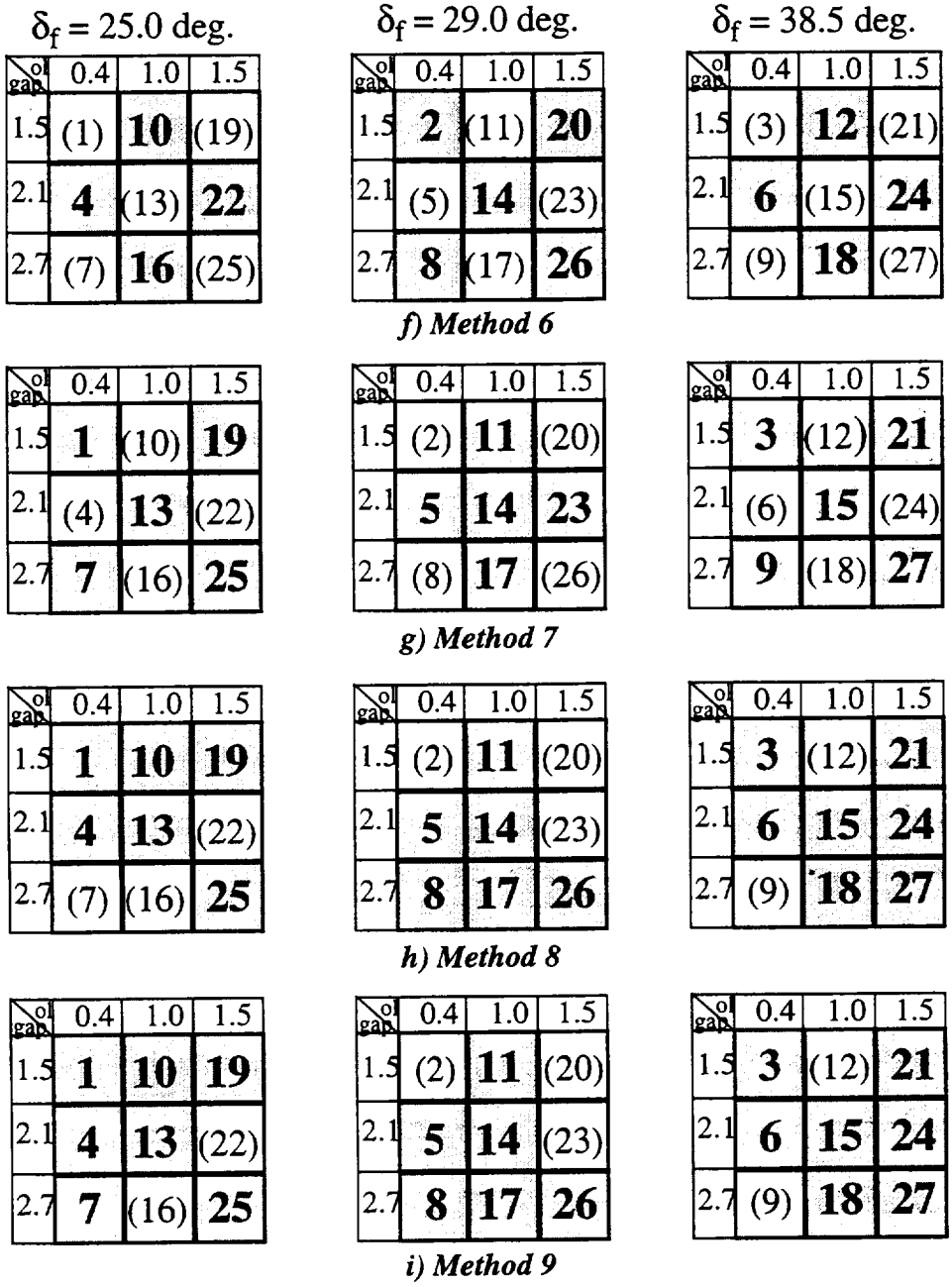


Figure 6.8 (continued) Training data subsets $\delta_s = 6°$ (shaded boxes represent cases that are included in the training set whereas the cases in the white boxes and parentheses are omitted).

ever, the same trends are seen for *L/D* as C_l, C_d, and C_m.

6.3.1 Subsets of Training Data for Six-Degree-Deflected Slat

Method 1 designates the training set which includes all the configurations in the training set. As expected, this has the lowest rms error for all the training methods tested since all the configurations tested are included in the training set. The rms errors for C_l, C_d, and C_m are shown in Figure 6.9. Method 1 represents the training data in a nonlinear fashion for all four inputs (δ_f, gap_f, ol_f, and α). Method 1 predicts C_d and C_m with almost no error for all 27 cases. C_l is predicted within the acceptable error. Case 27 which has a flap setting of $\delta_f = 38.5°$, $gap_f = 2.7\%c$, and $ol_f = 1.5\%c$ has the highest prediction error. Case 27 is the boundary of the design space and this may lead to the problem since there is no data for the neural networks to learn on the outer boundary that has values greater than it.

The next method that is used to train the neural networks is Method 2 which has a nonlinear representation of the flap deflection and angle of attack. The other two inputs, flap gap and overlap, are represented in a linear manner. This method contains 56% of the configurations in the entire training set. The rms errors (Figure 6.10) are low for the cases that are in the training set and high for the cases that are not included in the training. The C_l rms error is greater that what is acceptable for six of the cases. This method was expected to do poorly since it represents the aerodynamic data as linear when it is known to be non-linear.

Another subset tested to train the neural networks is a checkerboard method. Method 3 contains only interior points of the checkerboard as shown in Figure 6.12. This method contains 44% of the total configurations. This method predicts poorly for cases that are not in the training set as shown in Figure 6.11. The prediction level for C_l and C_m are not within the acceptable range. Examining the cases that are in Method 3 shows that this is a bad representation of the data since their is no configuration with the following gap and overlap combinations: $gap_f = 1.5\%c$ with $ol_f = 0.4$ and $1.5\%c$; $gap_f = 2.1\%c$ with $ol_f = 1.0\%c$; and $gap_f = 2.7\%c$ with $ol_f = 0.4$ and $1.5\%c$. Thus, the data is represented in a linear fashion which is incorrect.

Method 4 is another checkerboard subset which contains only the cases excluded from Method 3 in the training set as is illustrated in Figure 6.8. Here, the corners and middle cases are kept and the interior cases are omitted. Method 4 contains 56% of the entire training set. The neural networks when trained with Method 4 predict poorly the cases which are omitted from the training set as was seen previously. Figure 6.12 shows the C_l and C_m prediction error are higher than what was set to be acceptable for the omitted cases. Like, Method 3, there is no representation of the configurations with some particular combinations: $gap_f = 1.5\%c$ with $ol_f = 1.0$; $gap_f = 2.1\%c$ with $ol_f = 0.4$ and $1.5\%c$; and $gap_f = 2.7\%c$ with $ol_f = 1.0\%$c.

The next training set tested, Method 5, is a combination of Methods 3 and 4 as shown in Figure 6.8. For $\delta_f = 25°$ and $\delta_f = 38.5°$, the corner and middle high-lift riggings are used in the training set. In contrast, the interior ones are kept in the training set for $\delta_f = 29.0°$. Thus, there is representation in the training set for every high-lift rigging combination even though they are for various flap deflections. The rms errors (Figure 6.13) show that the prediction for C_l, C_d, and C_m are very good for the cases that are in the training set. Even though there are a few cases which were not included in the training where the C_l rms error is high, the prediction

for several of the cases is quite good. The C_d prediction for most of the cases (included or not in the training set) is good. The moment coefficient prediction is accurate for most of the configurations in the training set. Also, the C_m prediction is good for more than half of the cases omitted in the training set.

The next method that was tested is the reverse of Method 5 as shown in Figure 6.8. Method 6 contains only 48% of the total configurations. This method did a poor job of training the neu-

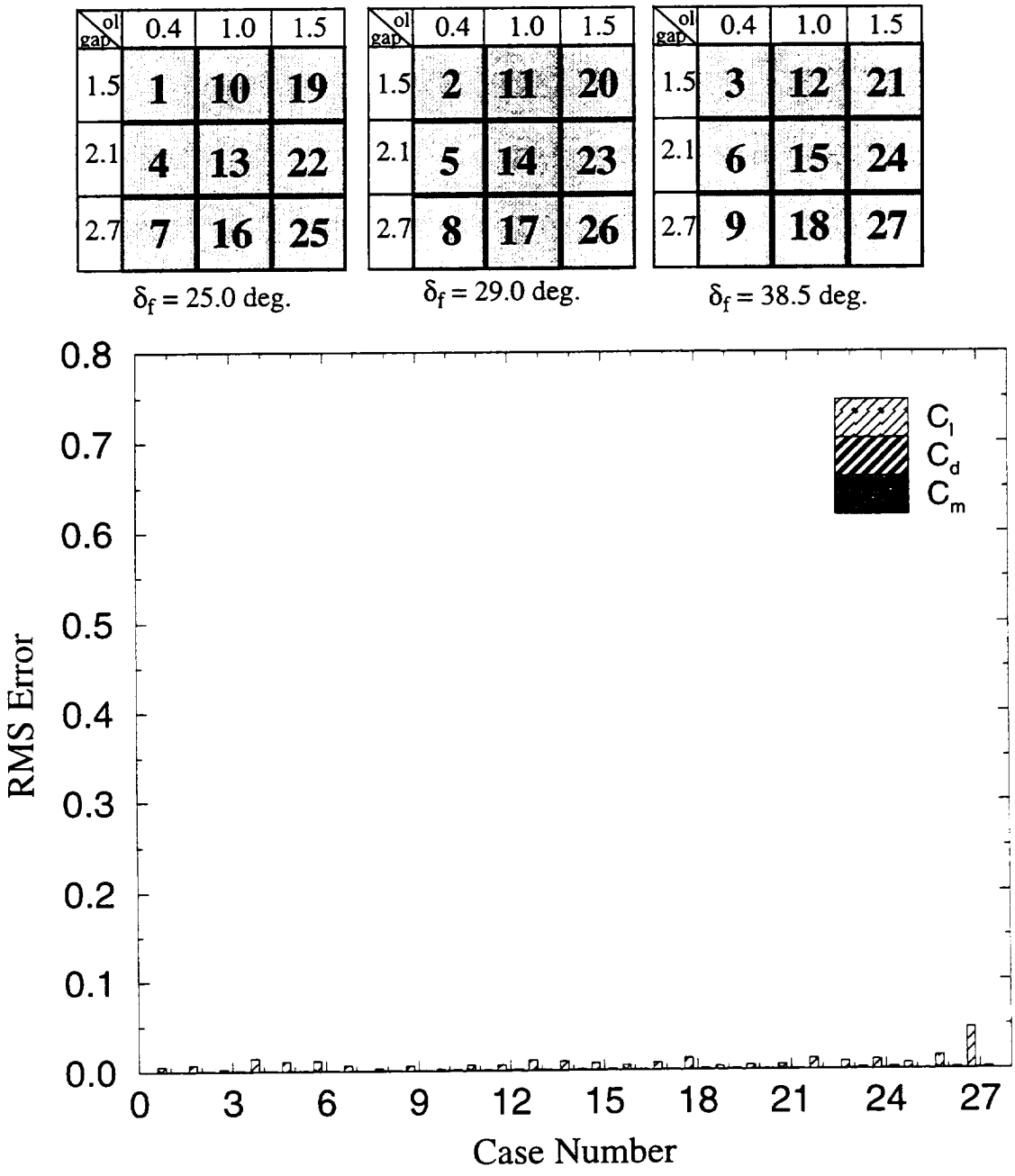


Figure 6.9 Summary of rms error from neural network prediction of aerodynamic coefficients for Method 1. Shaded boxes indicate which flap configurations are contained in the training data set.

ral nets to predict C_l for the cases that are omitted from the training set as shown in Figure 6.14. Surprisingly, the C_d and C_m prediction is good for 26 out of the 27 cases. Case 9 has high C_d and C_m rms errors and is one of the boundary points of the design space. The neural net, for the most part, predicts the lift coefficient accurately for the high-lift settings that are in the training set.

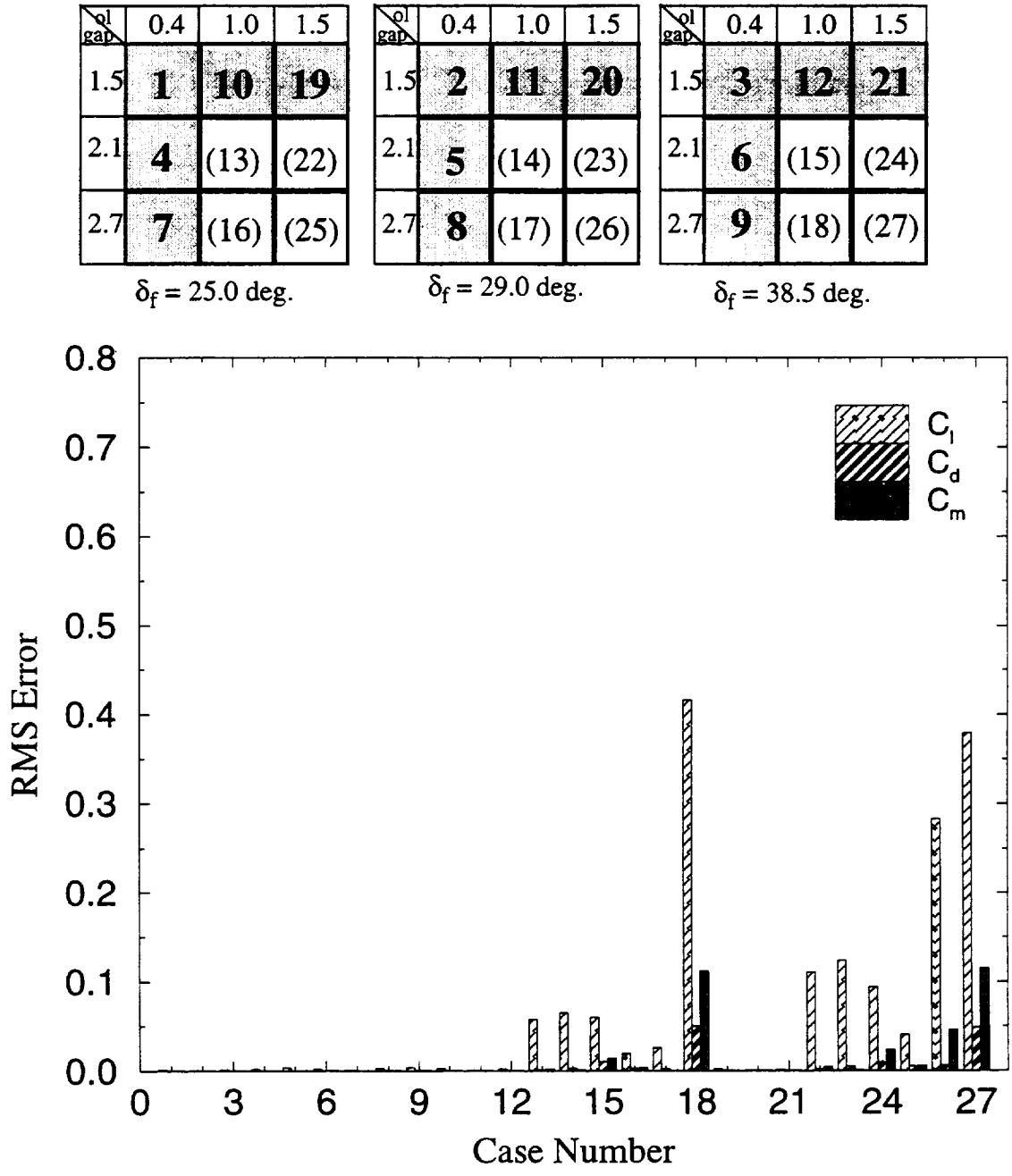


Figure 6.10 Summary of rms error from neural network prediction of aerodynamic coefficients for Method 2. Shaded boxes indicate which flap configurations are contained in the training data set.

Since Method 5 has been found to be the best subset to use to train the neural networks to predict the aerodynamics accurately, the following subsets are created by adding configurations back to Method 5. Method 7 was created by added one more configuration to be included in the training set. Case 14 was added back into the training data set and now Method 7 contains 56% of the entire data. The neural networks predict C_l, C_d, and C_m accurately for all the cases in the training set as is illustrated in Figure 6.15. Now, for the cases which are not included in the

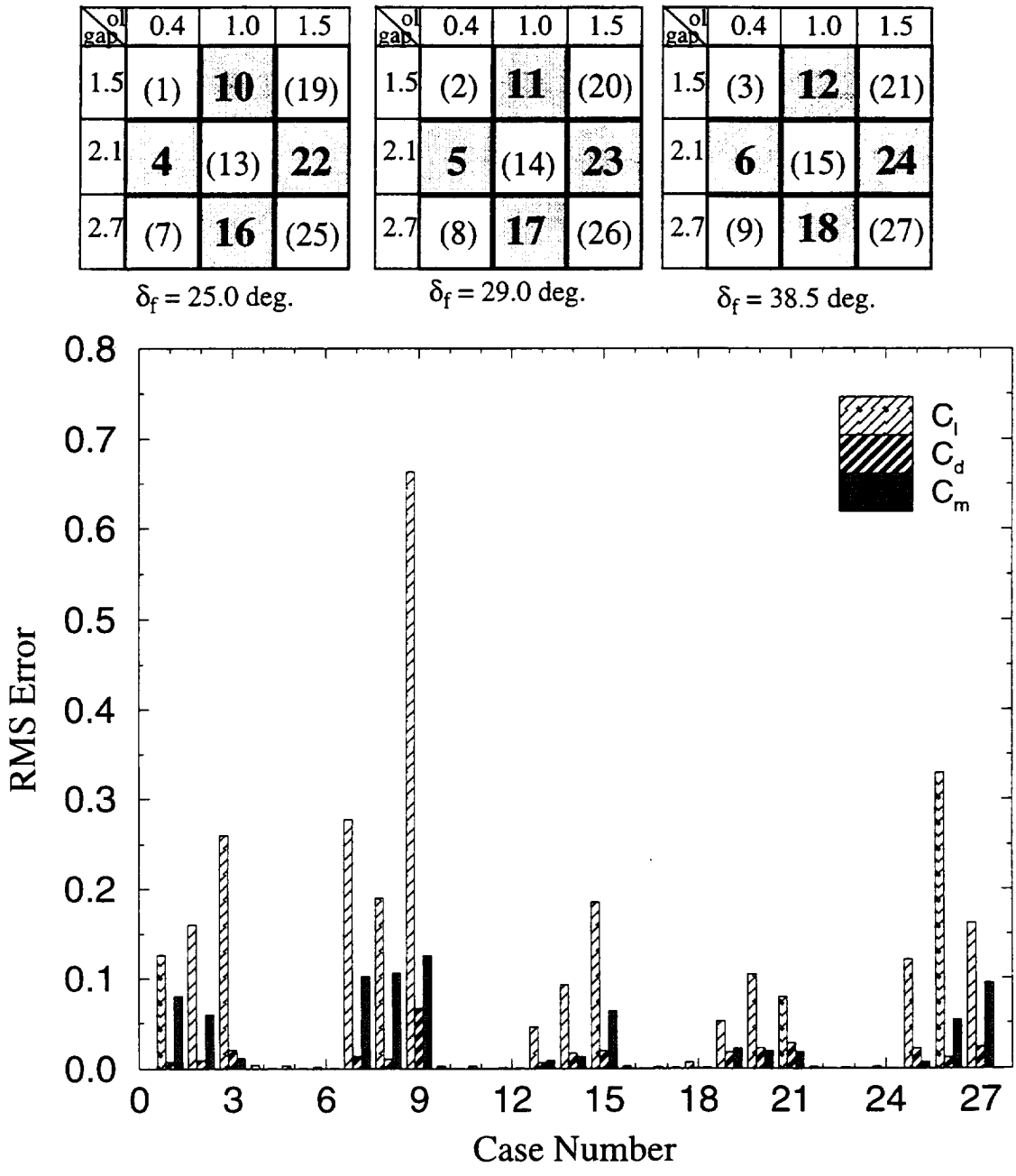
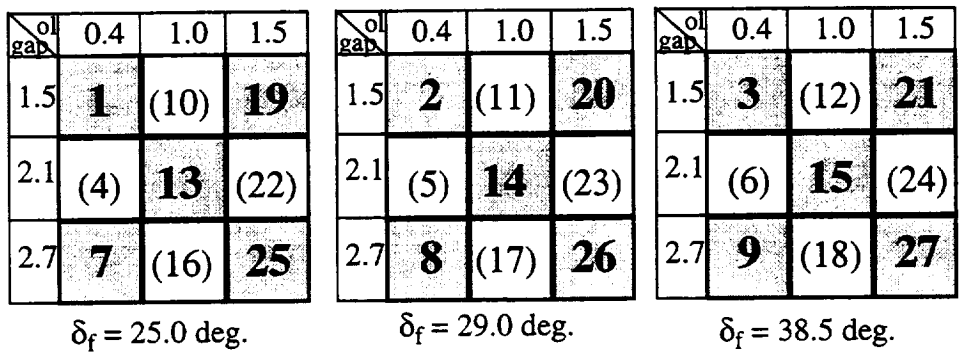


Figure 6.11 Summary of rms error from neural network prediction of aerodynamic coefficients for Method 3. Shaded boxes indicate which flap configurations are contained in the training data set.

training set, Method 7 does a fairly good job training the neural network to predict C_l and C_m. There are a few cases that predict poorly but there also are more cases that do very well. Also, C_d is predicted very accurately for all 27 configurations.

To further improve the accuracy of the prediction while still reducing the number of configurations relative to the full training set, careful selections of the configurations contained in the



δ_f = 25.0 deg. δ_f = 29.0 deg. δ_f = 38.5 deg.

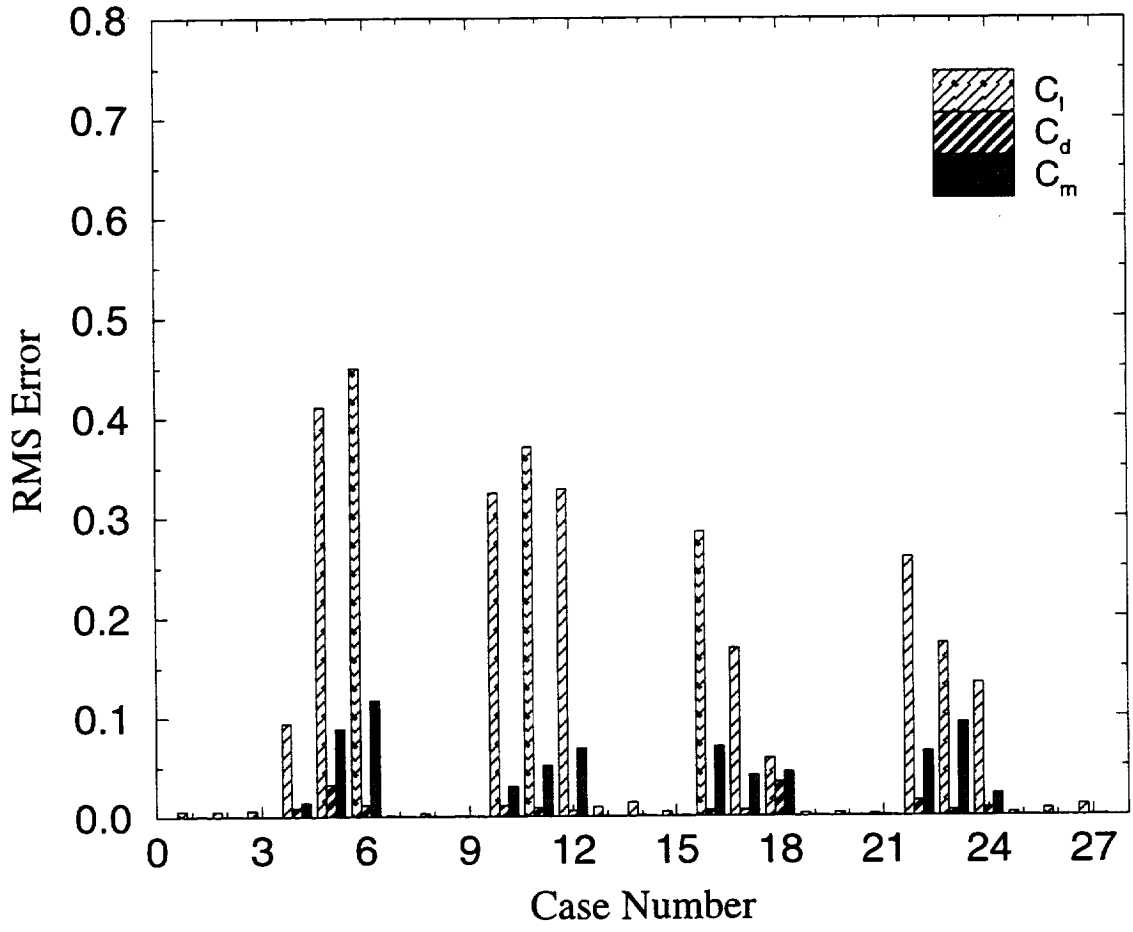
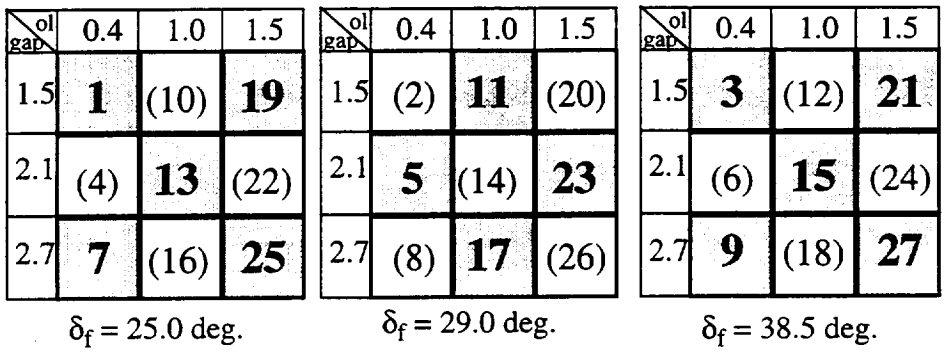


Figure 6.12 Summary of rms error from neural network prediction of aerodynamic coefficients for Method 4. Shaded boxes indicate which flap configurations are contained in the training data set.

training set are created. An example of a subset that is very successful in training the neural networks to predict the aerodynamics of the Flap-Edge geometry is Method 8. Method 8 contains 70% of the data of the full training set. As is apparent by the error bars in Figure 6.16, the error is low for most cases and is well within the acceptable error even for the cases that are not in the training set. The C_l rms errors are exceptionally high for cases 6, 9, and 22. Cases 9 and 22 are not in the training set, however, case 6 is included. Both the drag and moment coefficient pre-



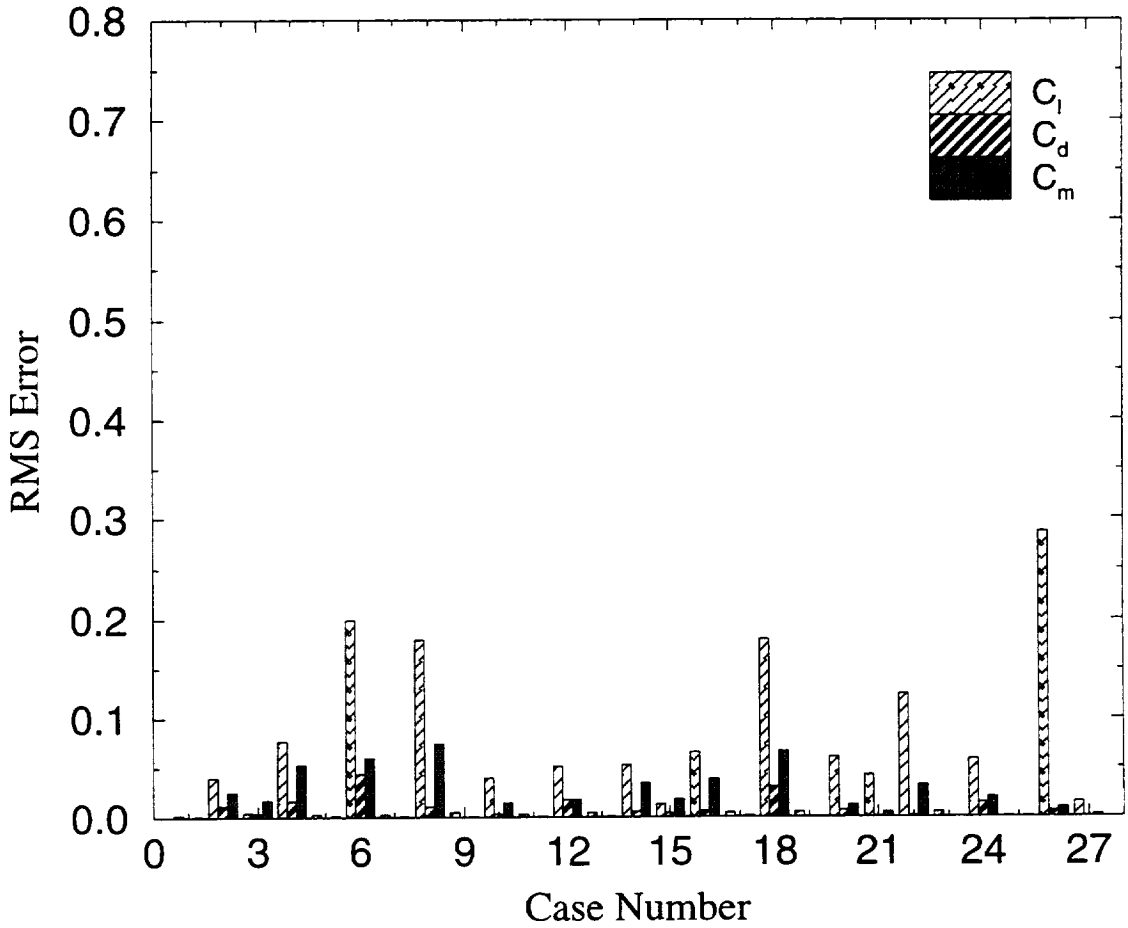


Figure 6.13 Summary of rms error from neural network prediction of aerodynamic coefficients for Method 5. Shaded boxes indicate which flap configurations are contained in the training data set.

dictions are very good and the errors are low. Overall, the prediction is good for this method and if the designer can tolerate a small percent of high error in a few cases in exchange for the computer resources that are being saved, about 30%, this method is well worth it.

The last subset that is tested is Method 9. Method 9 is similar to Method 8 except one more configuration (case 7) is added back to the training set. Method 9 has a total of 74% of the total

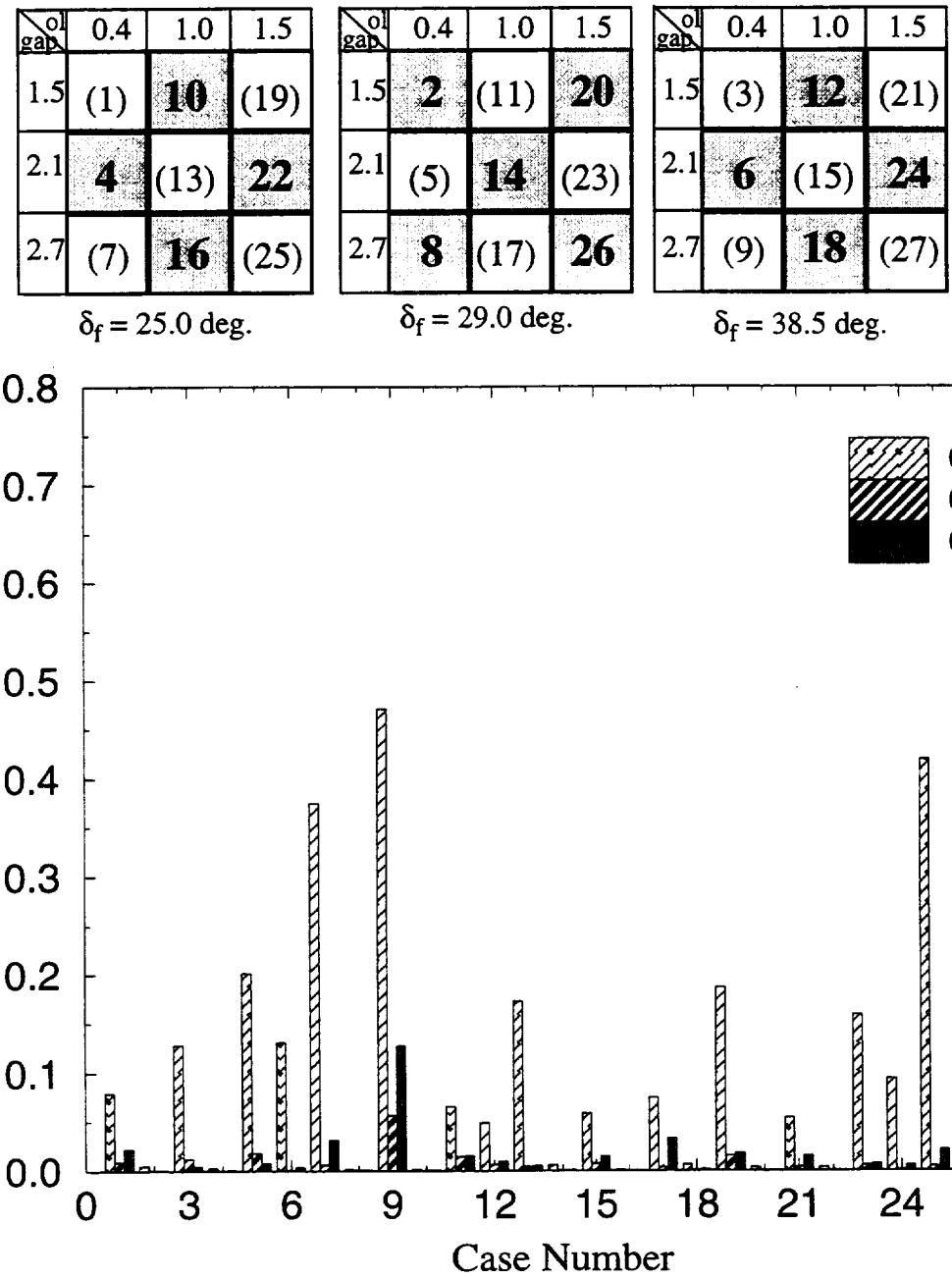


Figure 6.14 Summary of rms error from neural network prediction of aerodynamic coefficients for Method 6. Shaded boxes indicate which flap configurations are contained in the training data set.

configurations in the training set. The rms errors, shown in Figure 6.17, are low and within the acceptable range for most of the configurations. Only three cases (9, 16, and 22) have a C_l rms error above the acceptable error and only case 9 is exceptionally high. These three cases are not included in the training set. The drag prediction is good for all cases. On the other hand, the moment prediction is high for the same three cases as the lift prediction. Once again, by allowing this small error to be acceptable, valuable resources will be saved (26%) by training the

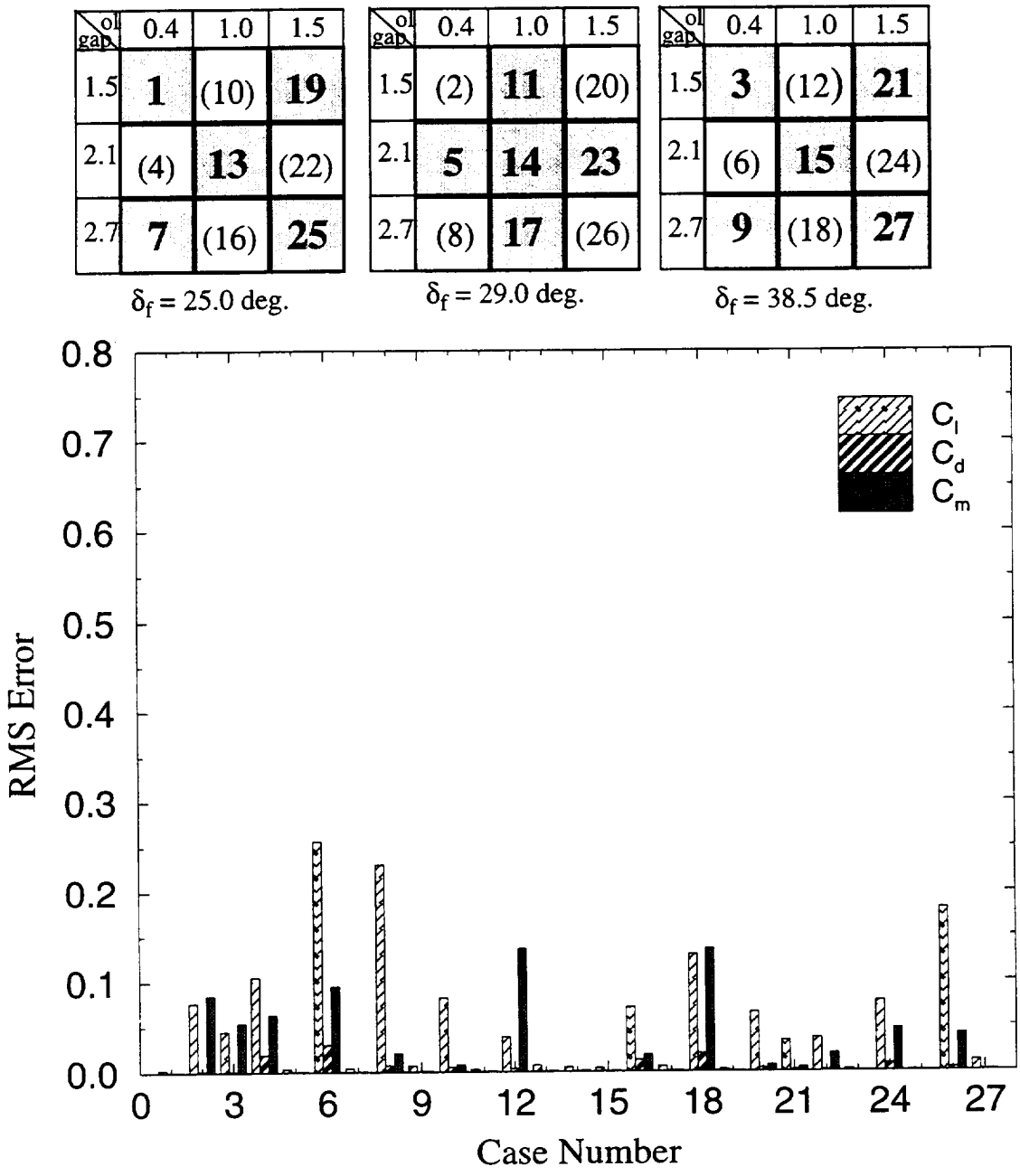


Figure 6.15 Summary of rms error from neural network prediction of aerodynamic coefficients for Method 7. Shaded boxes indicate which flap configurations are contained in the training data set.

neural networks with Method 9 to predict the aerodynamic coefficients.

6.3.2 Mean and Standard Deviation

The mean and standard deviation of each method's rms errors of the lift, drag, and moment coefficients are calculated to evaluate the accuracy of the predictions obtained using the various

gap \ ol	0.4	1.0	1.5
1.5	**1**	**10**	**19**
2.1	**4**	**13**	(22)
2.7	(7)	(16)	**25**

δ_f = 25.0 deg.

gap \ ol	0.4	1.0	1.5
1.5	(2)	**11**	(20)
2.1	**5**	**14**	(23)
2.7	**8**	**17**	**26**

δ_f = 29.0 deg.

gap \ ol	0.4	1.0	1.5
1.5	**3**	(12)	**21**
2.1	**6**	**15**	**24**
2.7	(9)	**18**	**27**

δ_f = 38.5 deg.

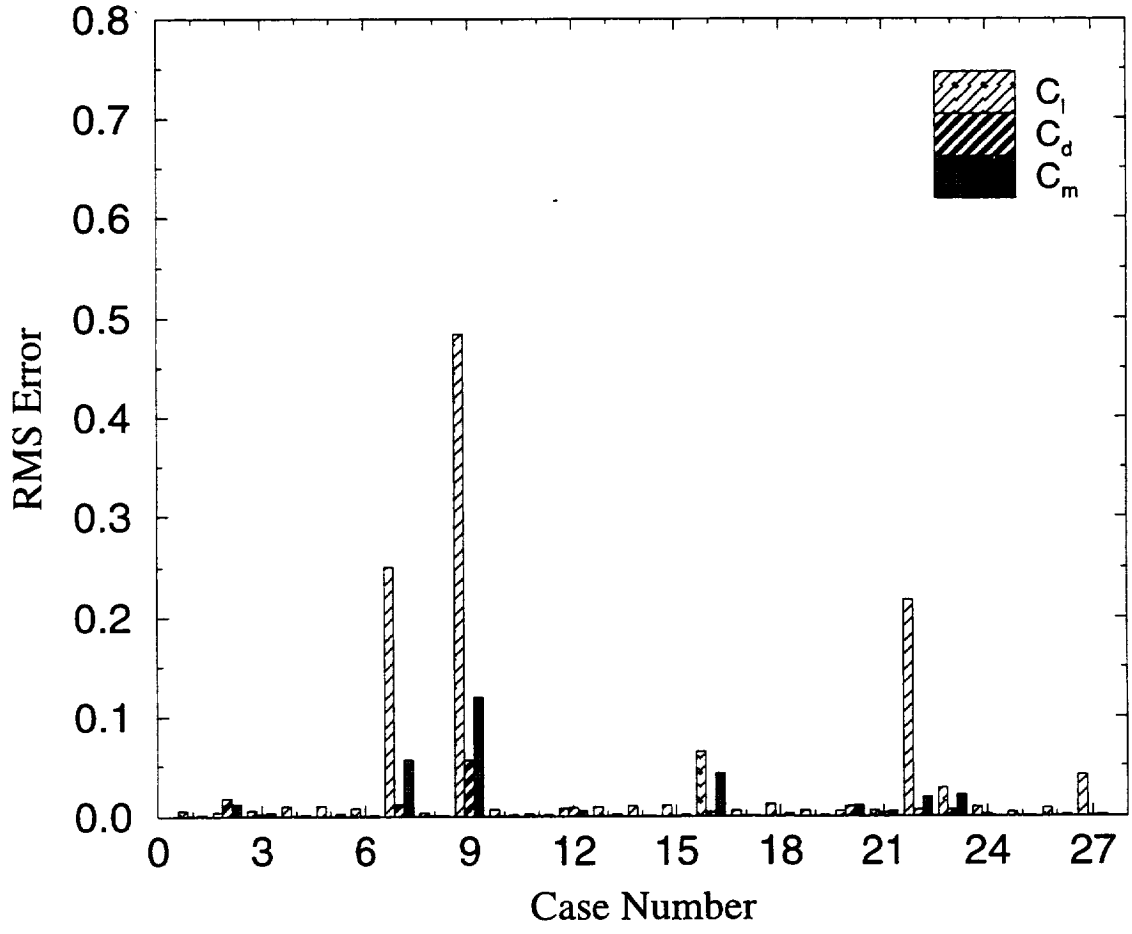


Figure 6.16 Summary of rms error from neural network prediction of aerodynamic coefficients for Method 8. Shaded boxes indicate which flap configurations are contained in the training data set.

training sets. The mean rms errors for Methods 1-9 for the six-degree-deflected slat is shown in Figure 6.18. Likewise, the standard deviation of the rms error for the six-degree-deflected slat is shown in Figure 6.19. Both figures show, as one would anticipate, Method 1 has the lowest mean and standard deviation. This is also clearly seen in Figure 6.9 that showed Method 1 to have almost no errors in predicting the aerodynamic coefficients. This is expected since all cases are included in the training set. The highest rms errors are obtained by Method 4 which

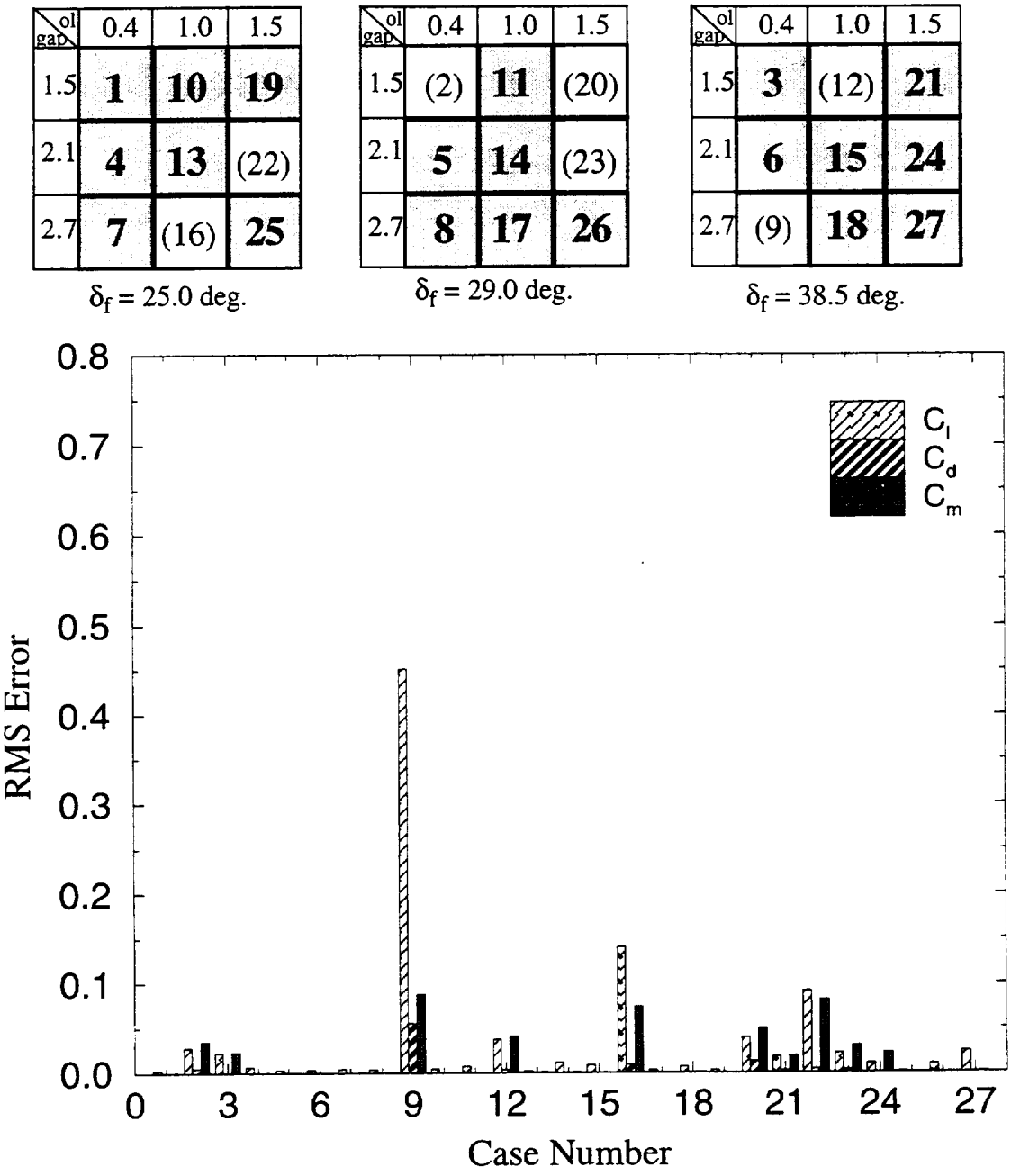


Figure 6.17 Summary of rms error from neural network prediction of aerodynamic coefficients for Method 9. Shaded boxes indicate which flap configurations are contained in the training data set.

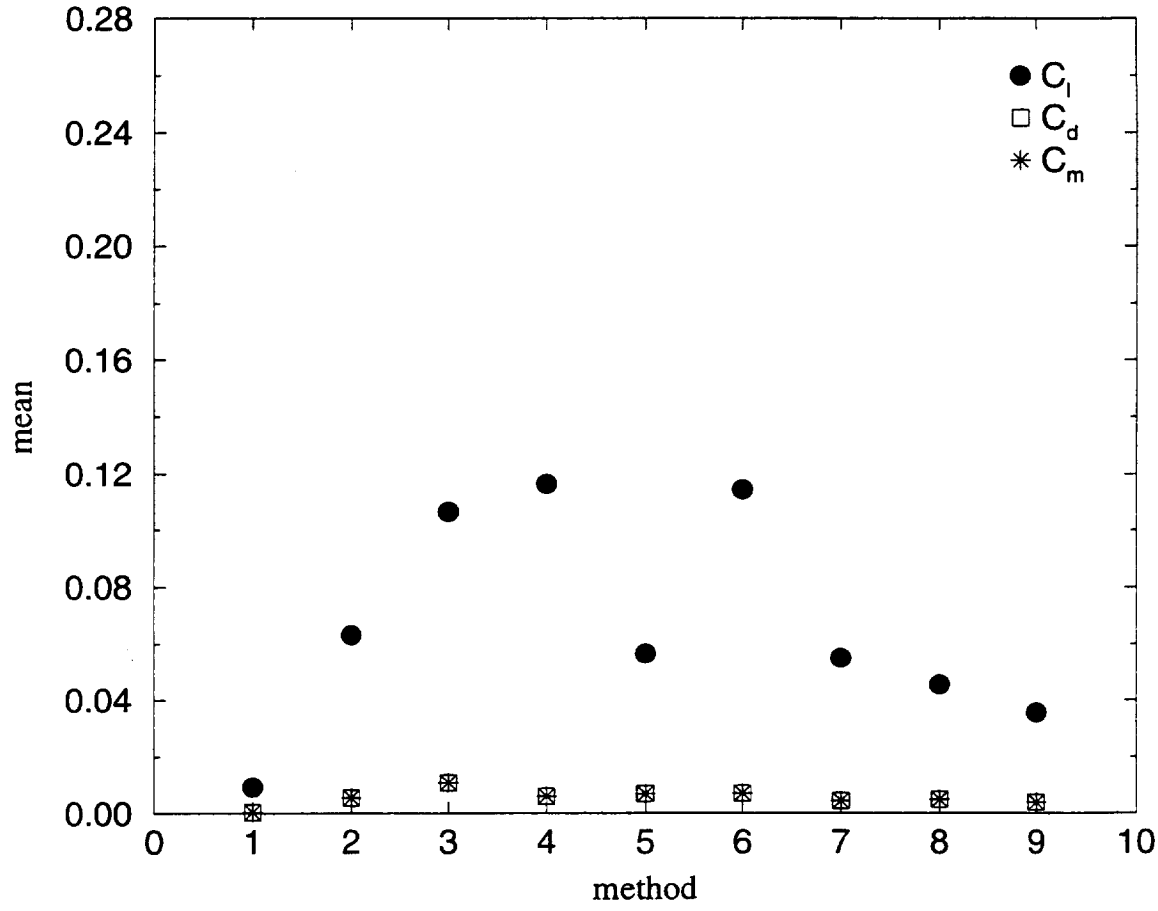


Figure 6.18 Mean rms errors for subsets of the training data for six-degree slat deflection.

contains 55% of the data. This method is generated by choosing the corners and the center of each deflection matrix as shown in Figure 6.8. Examining this method closely shows that there is not enough representation of the different combinations of high-lift configurations to train the neural networks to learn and to predict the aerodynamics of the Flap-Edge geometry. This led to the more carefully chosen methods of 5, 7, 8, and 9.

Method 5 (Figure 6.13) is the most accurate method of training if only approximately 50% of the data is available. Method 5 consists of 52% of the total configurations but more importantly, it contains a good variety of the different flap deflections, gaps, and overlaps and combinations of these to accurately represent the aerodynamics of the multi-element airfoil. Method 7 is also accurate and contains only 56% of the total configurations. If more computer resources are available, then Methods 8 and 9 can be used to train the neural networks. These methods have the lowest mean and standard deviations of the rms errors besides Method 1. Methods 8 and 9 contain 70% and 74% of the entire training set, respectively, and their mean rms errors are below the acceptable error. Again, these methods have a good representation of the high-lift riggings within the design space. Thus, with only 70% of the sparse training data set, the neural networks can be trained to accurately predict the aerodynamic performance of a multi-element airfoil.

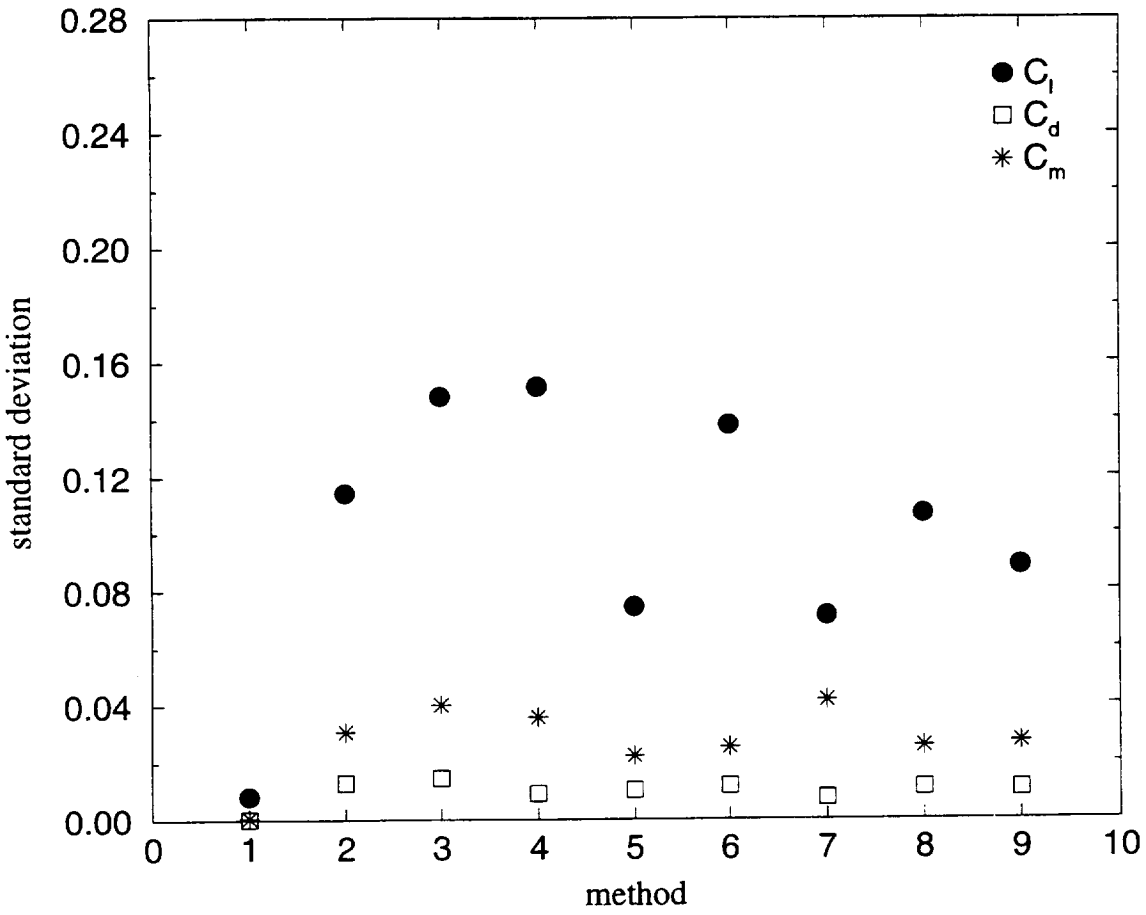


Figure 6.19 Standard deviation of the rms errors for subsets of the training data for six-degree slat deflection.

The same procedure that is used for the slat with six-degree-deflected slat is used for the twenty-six-degree-deflected slat. The methods of training or subsets of the data have the same patterns as shown in Figure 6.8, however, cases twenty-eight through fifty-four are used instead (refer to Figure 6.2) and are shown in Figure 6.20. After training the neural networks with the nine different methods and predicting the aerodynamic coefficients, the means and standard deviations of the rms errors for each individual method of training are calculated. The mean rms errors, show the same trends as in the previous slat deflection. Method 1 has the lowest C_l rms error, however, Methods 8 and 9 have about the same error as Method 1 for C_l. Methods 8 and 9 have the lowest C_d rms errors. Again, if only approximately 50% of the configurations are used to train the neural networks, then Methods 5 and 7 should be used. The standard deviation of the rms errors for the twenty-six-degree-deflected slat is shown in Figure 6.22. The same trends are seen as was previously described. By comparing the values of the rms errors for the two different slat deflections airfoils, it is shown that the six-degree-deflected slat has lower means and standard deviation of the rms errors. The high-lift physics of the multi-element airfoils with these two different slat deflections are discussed and compared in the next subsection.

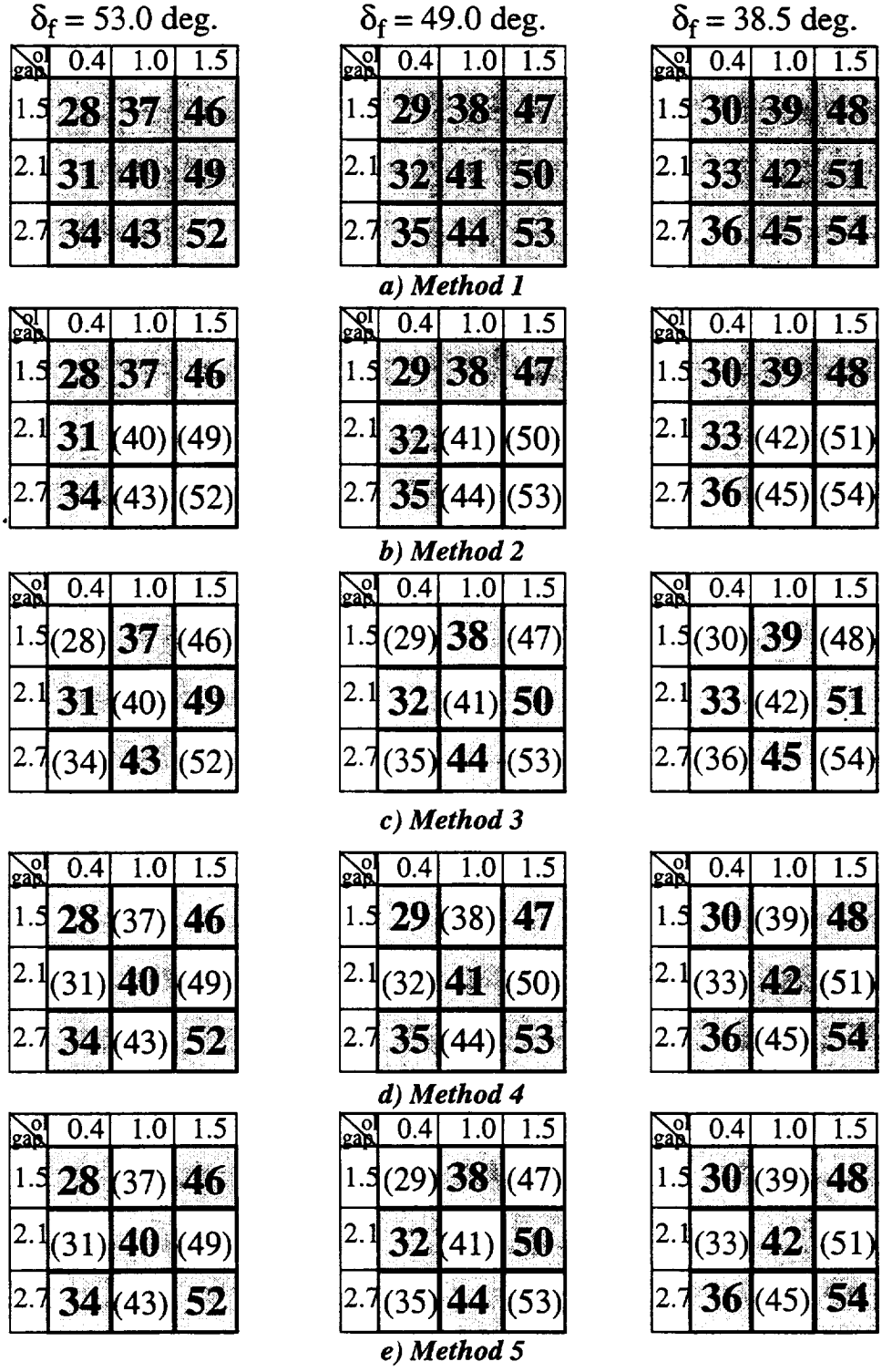


Figure 6.20 Training data subsets for δ_s = 26° (shaded boxes represent cases that are included in the training set whereas the cases in the white boxes and parentheses are omitted).

6.3.3 High-Lift Physics

In order to understand the physics of this complex flow, the pressure distributions are plotted and compared for standard cases with slat deflection of 6 and 26 degrees. The high-lift setting for the first case is δ_s = 6.0°, gap_s = *2.0%c*, ol_s = *-0.05%c*, δ_f = 38.5°, gap_f = *2.1%c*, ol_f = *1.0%c* at α = 10.0° and the second high-lift setting is δ_s = 26.0°, gap_s = *2.0%c*, ol_s = *-0.05%c*, δ_f = 53.0°, gap_f = *2.1%c*, ol_f = *1.0%c* at α = 10.0°. Figure 6.23a shows the slat, main, and flap elements for both configurations. As a reminder, the main element is the same

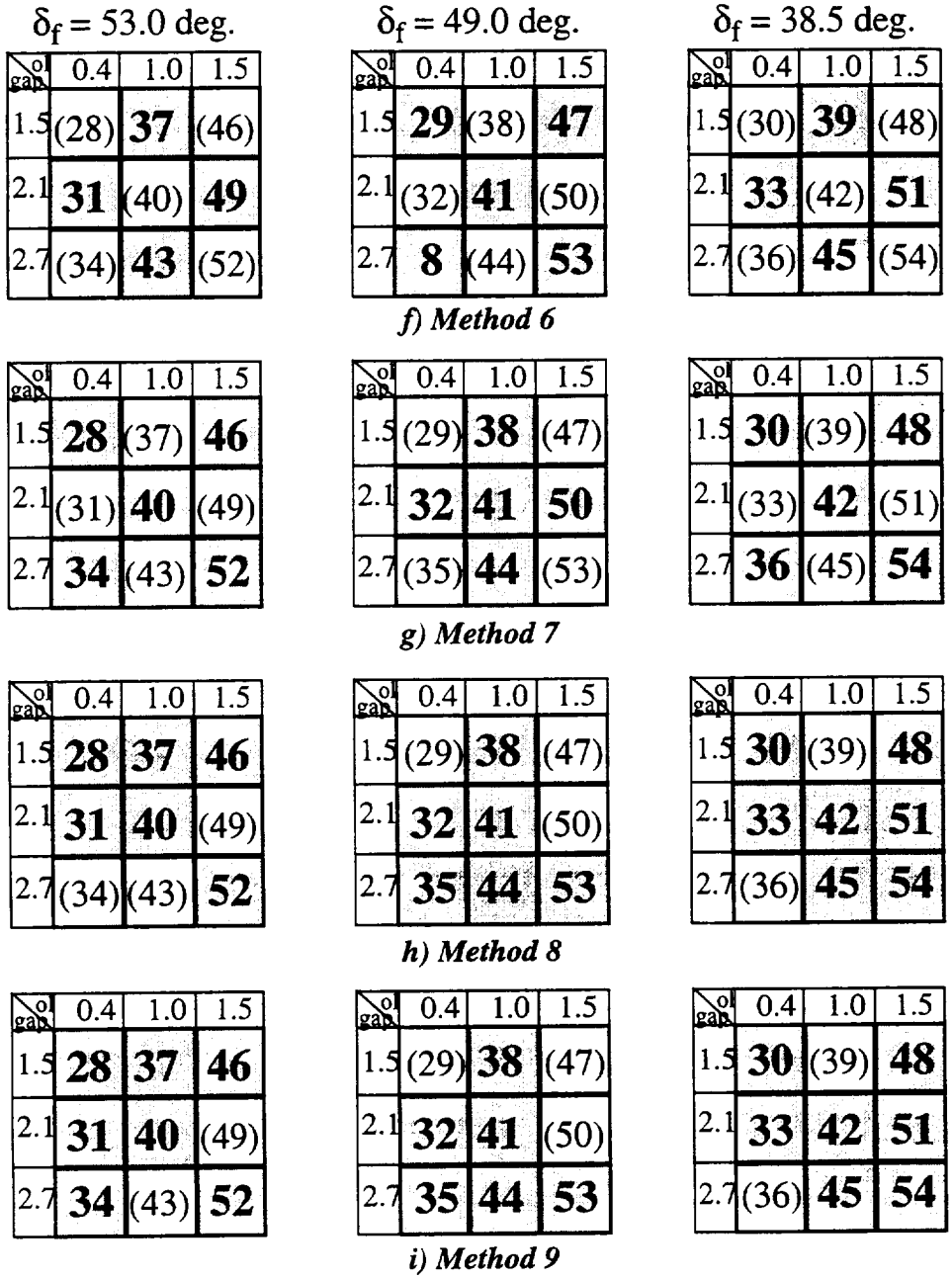


Figure 6.20 (continued) Training data subsets δ_s = 26° (shaded boxes represent cases that are included in the training set whereas the cases in the white boxes and parentheses are omitted).

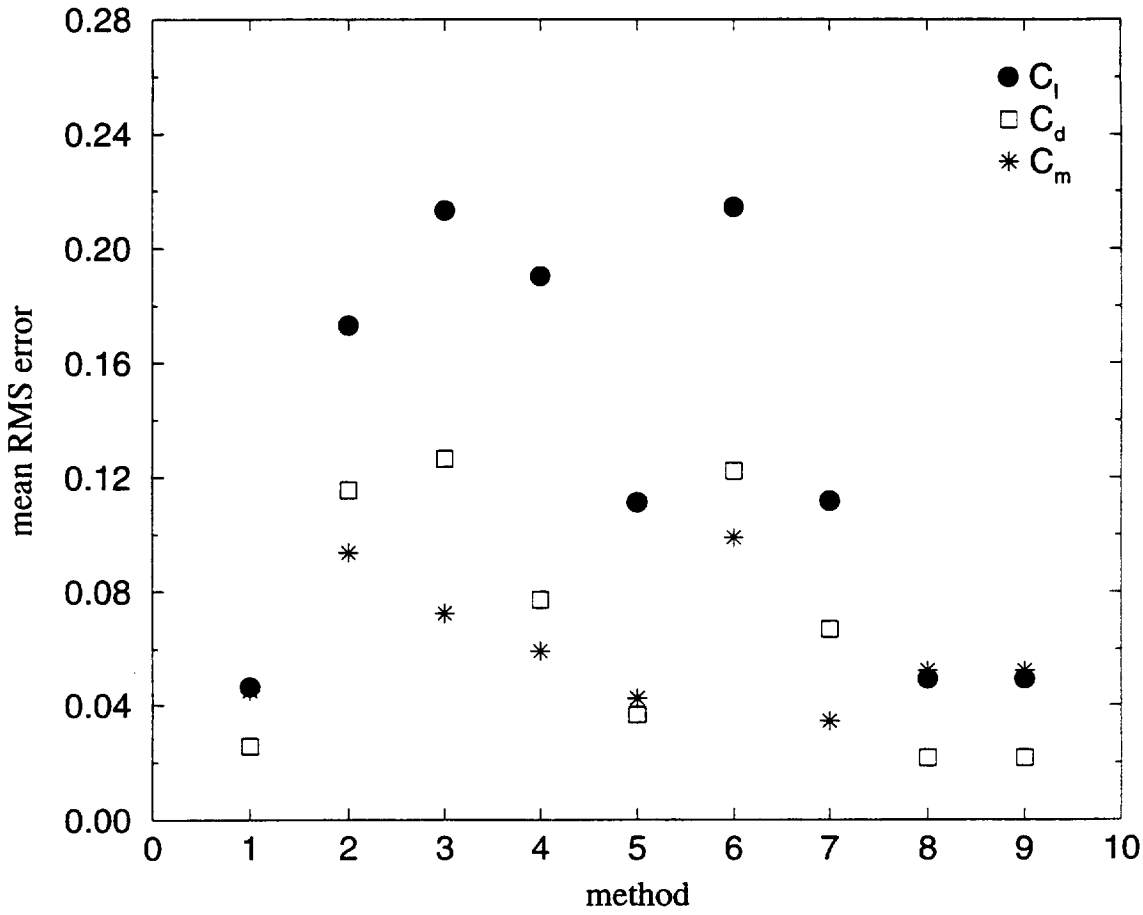


Figure 6.21 Mean rms errors for subsets of the training data for twenty-six-degree slat deflection.

for both configurations. For the six-degree-deflected slat case, the flap element as well as the slat element are deflected less than in the other configuration. The pressure distribution is different for both cases for all elements. First, the six-degree-deflected slat configuration has the flow attached for all three elements as seen in Figure 6.23b. The suction pressure on the slat is larger than in the higher slat deflection case. On the contrary, the suction pressure is lower on the main element than the δ_s = 26.0° configuration. There are interesting features on the flap element. The sharp spike at the trailing edge (also seen in the slat elements) occurs from the sharp point at the trailing edge of the flap geometry. The numerical grid comes to a sharp corner at the trailing edge, the flow must accelerate at this point causing the pressure to drop. The multiple spikes that are located at the leading-edge of the flap element are associated with the original definition of the geometry. The flap at this region is faceted due to the high curvature. The pressure spikes are representative of what the flow is actually doing. The flow is turning around at these facets and accelerating. Second, the δ_s = 26.0° airfoil shows that the flow is separated for the slat and flap elements. This is common for the configurations with the higher slat deflection. The flowfields for the configurations with δ_s = 26.0° are severely separated and thus the numerical data that is used to train the neural networks is not predicted as accurately because the aerodynamics vary so dramatically for the different configurations. The data for the six-degree-deflected slat is better behaved and thus less noisy. The twenty-six-degree-deflected slat

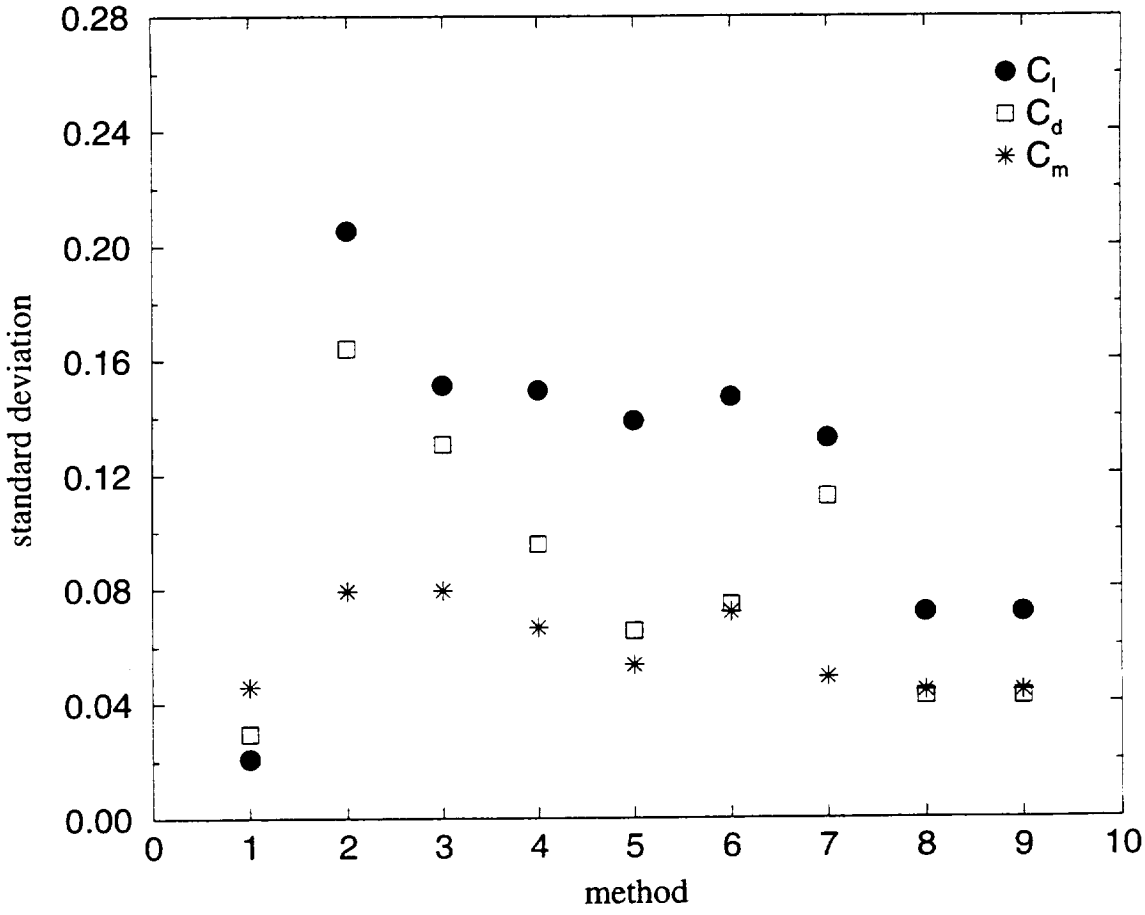
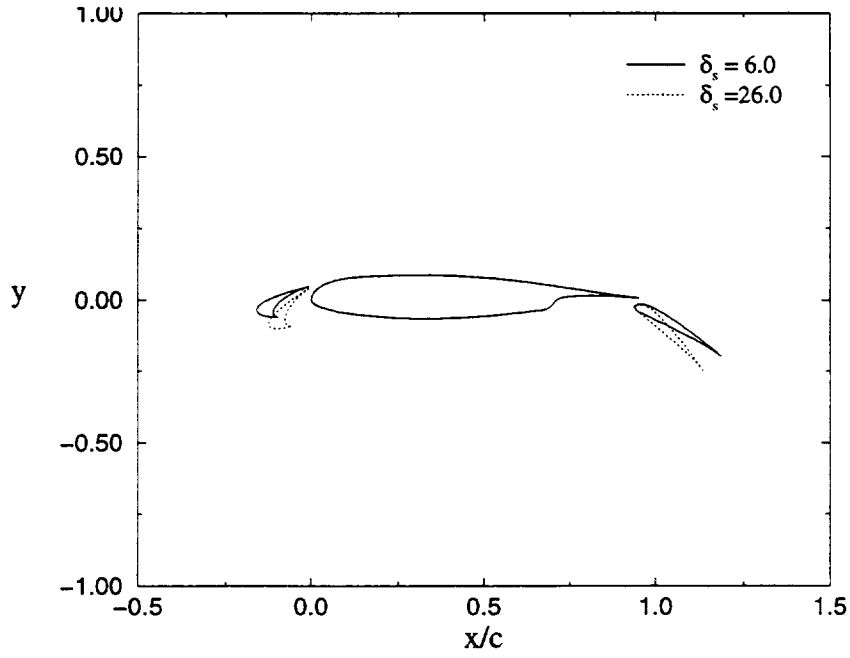


Figure 6.22 Standard deviation of the rms errors for subsets of the training data for twenty-six-degree slat deflection.

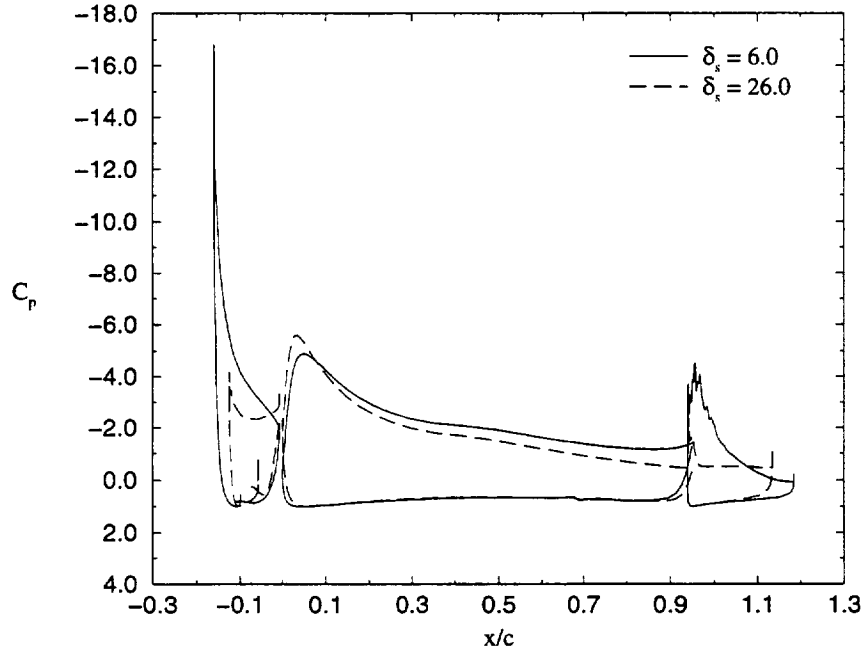
configurations have extremely high flap deflection angles which are well beyond the normal flight envelope. The flow is severely separated at the higher deflected flaps. Consequently, the neural networks do a better job of learning and predicting the flowfield of the more benign six-degree-deflected slat airfoil.

6.3.4 Example of Neural Network Prediction

The neural network's ability to predict the aerodynamics of a high-lift rigging that is not included in the training set is tested by training the neural network with Method 8 which contains 70% of the full training set and comparing the predicted values (denoted by an open square in Figure 6.24) with the INS2D calculated values (denoted by a filled diamond Figure 6.24). The neural networks were used to predict the aerodynamics of the airfoil with a flap high-lift setting of $\delta_f = 27.0°$, $gap_f = 2.4\%c$, $ol_f = 1.1\%c$ which is not used to train the neural networks as shown in Figure 6.24. The lift coefficient versus angle of attack is shown in Figure 6.24a and the neural network does accurately (to within 1.5% of C_l) predict C_l for all angles of attack tested. In this case, the pressure difference rule predicted $\alpha = 10.0°$ to be the location of maximum lift. Thus, the neural network was only tested from $\alpha = 0.0°$ to $\alpha = 10.0°$. The

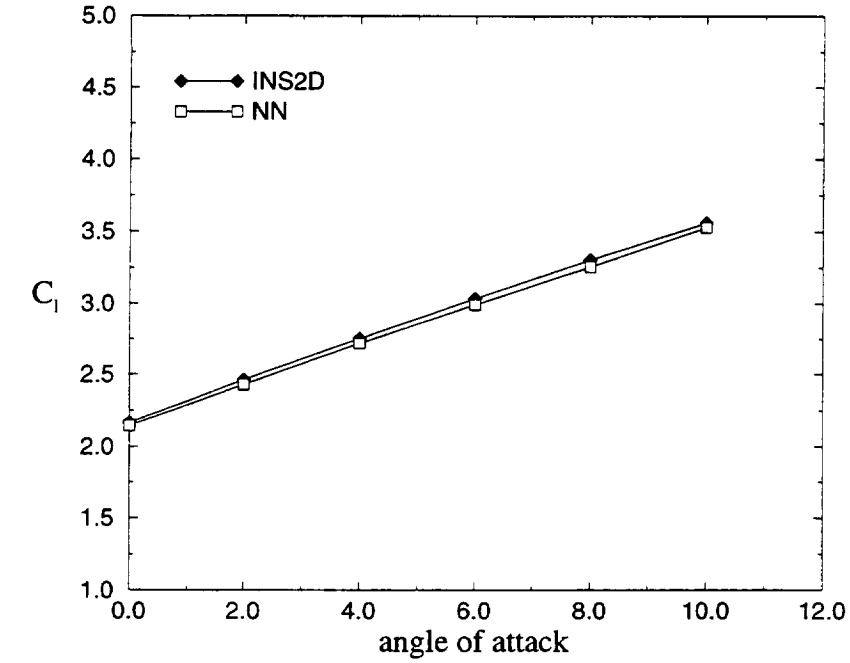


a) High-lift settings

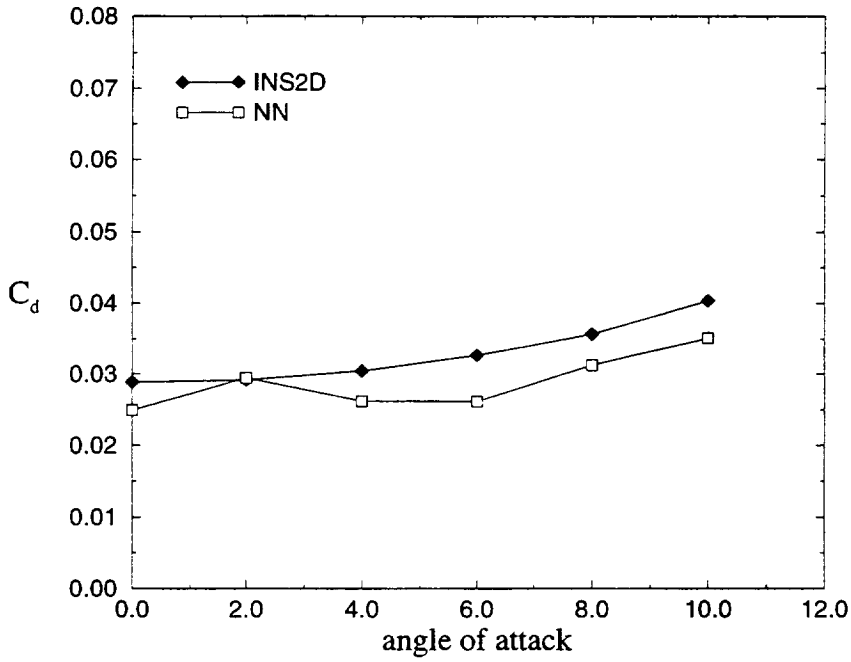


b) Pressure distribution

Figure 6.23 Comparison of slat deflection aerodynamics: $\delta_s = 6.0°$, $gap_s = 2.0\%c$, $ol_s = -0.05\%c$, $\delta_f = 38.5°$, $gap_f = 2.1\%c$, $ol_f = 1.0\%c$, $\alpha = 10.0°$ and $\delta_s = 26.0°$, $gap_s = 2.0\%c$, $ol_s = -0.05\%c$, $\delta_f = 53.0°$, $gap_f = 2.1\%c$, $ol_f = 1.0\%c$ $\alpha = 10.0°$

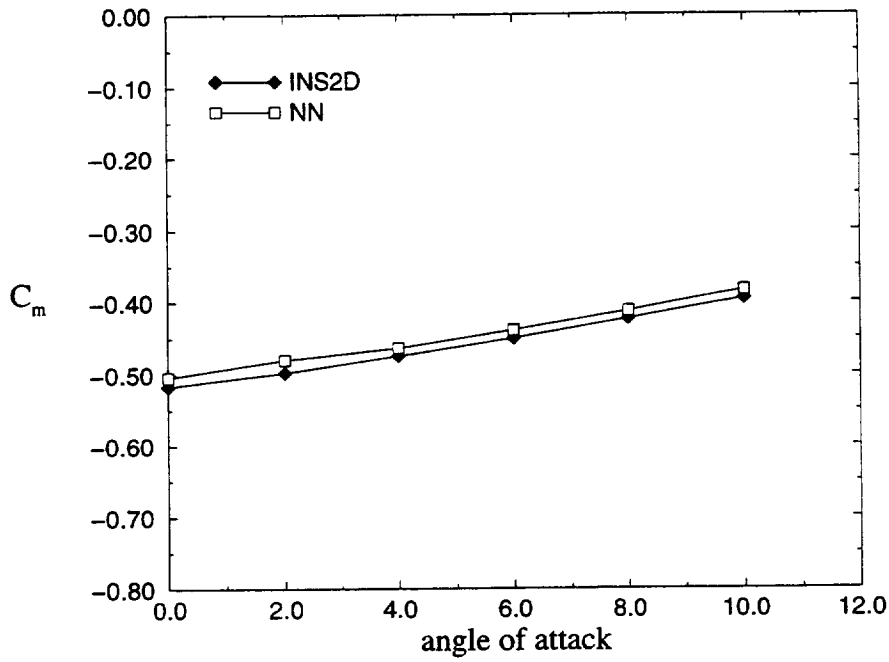


a) Lift versus angle of attack

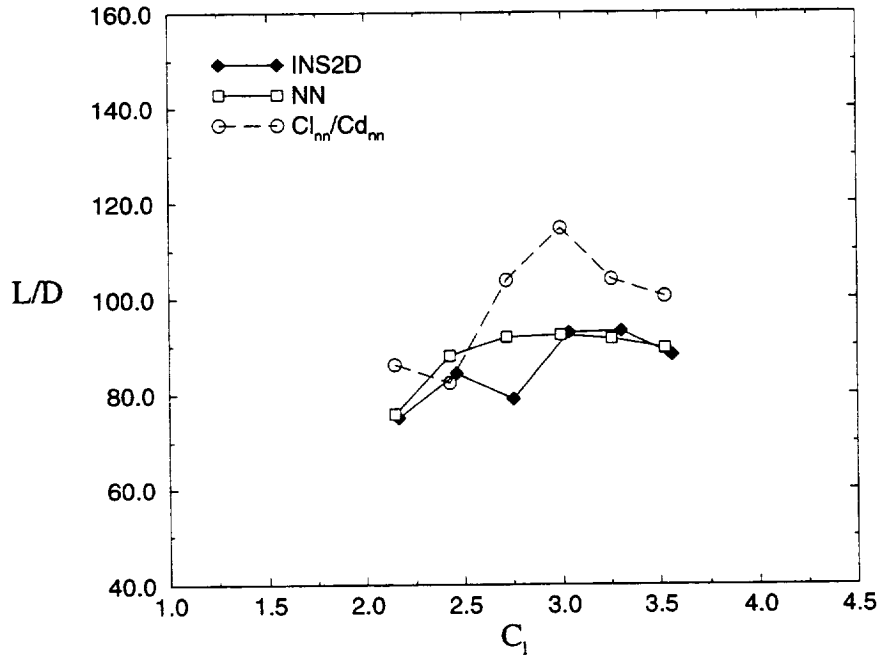


b) Drag versus angle of attack

Figure 6.24 Comparison of aerodynamic characteristics for $\delta_f = 27.0°$, $gap_f = 2.4\%c$, $ol_f = 1.1\%c$ which is not in the training set.



c) Pitching moment versus angle of attack



d) Lift-to-Drag ratio

Figure 6.24 (continued) Comparison of aerodynamic characteristics for $\delta_f = 27.0°$, $gap_f =$ *2.4%c*, $ol_f =$ *1.1%c* which is not in the training set.

neural network did not predict the drag coefficient exactly right as is seen in Figure 6.24b, however, the neural network did predict the same trend as the calculated INS2D values. The neural network under-predicted drag for all angles of attack except $\alpha = 2.0°$ where it predicted the same value as the numerical value. The pitching moment prediction has the same trends as the numerical data as illustrated in Figure 6.24c. The neural network slightly over-predicted the numerical data at all angles of attack. The lift-to-drag ratio prediction does not show the same trends as the INS2D calculated values since it does not predict the dip in *L/D* that is seen at $C_l = 2.75$. There is good agreement between the neural network prediction and the numerical data for $C_l = 2.16$ and for $3.03 < C_l < 3.56$. Some of the error might by caused by the fact that the neural networks are trained with 250 iterations, but the learning curves in Figure 6.3 and Figure 6.4 show that the neural network to predict *L/D* accurately should be trained with at least 400 iterations. $(L/D)_{nn}$ is also calculated from $C_{l_{nn}} / C_{d_{nn}}$ and is also shown in Figure 6.24d (denoted with an open circle) to test if the neural network to predict *L/D* is necessary. These predicted values are inaccurate. This occurs because the drag coefficient was predicted inaccurately and the errors are amplified in calculating $C_{l_{nn}} / C_{d_{nn}}$. Overall, however, the neural networks accurately predict the aerodynamics of a high-lift rigging that is not included in the training set that contains only 70% of the data.

6.4 Optimizing Using Neural Nets

In order for computational fluid dynamics to be able to impact the aircraft design, the design space needs to be easily searched for subsets of the design space specified by constraints, for maximums or minimums, and/or for a specific set of design variables. By integrating an optimizer to the enhanced design space capturing procedure that is developed with the neural networks, computational fluid dynamics data can now be readily accessible to the designers. The optimization process that is used is discussed in detail in Section 5.3 and will be referred to as optimization using neural networks.

The high-lift flap aerodynamics are optimized for the Flap-Edge airfoil by maximizing the lift coefficient. The design variables in this study are chosen to be the flap deflection, gap, overlap, and angle of attack. The optimization is performed with and without constraining any of the aerodynamic coefficients. The bounds on the design space are chosen to be the same as the design space that are used to train the neural networks with the exception that the angle of attack is bounded to $\alpha = 10.0°$ (for the optimization cases performed without constraints) since this is near the range where maximum lift is predicted to occur by the pressure difference rule for most of the configurations. The bounds for the design variables are shown in Table 6.1 for the six-degree-deflected slat data. To start the optimization, the original values of the design variables are arbitrarily chosen.

6.4.1 Optimization with Method 5 as Training Set

The optimization using the neural networks procedure is used to optimize the flap rigging for the six-degree-deflected slat. The different methods of training the neural networks that are discussed in Section 6.3 are used for the optimization. The optimal configurations, as well as the prediction accuracy, are different for each training method. The first method that is used to train the neural networks which are integrated with the optimizer is Method 5. As it has been

noted, Method 5 contains only 52% of the entire configurations. The results for five different optimization runs are shown in Table 6.2. Each of these runs has different initial or starting values (orig) of the design variables (DV). Gradient-based optimizers do not guarantee that the maximum which is found is the global maximum of the design space; it only guarantees an improvement. Thus, different starting values of the design variables are used to search the entire design space. The first optimization run, 5-A, has the initial design variables set to the lower bounds. Whereas, the second run, 5-B, has the initial values set to the upper bounds of the design space. In the third run, 5-C, the initial conditions are set to the average value of the lower and upper bounds. The last two runs have arbitrary initial values to test different regions of the design space. With this AI optimization process, the design space can be easily searched with several optimization runs because each run only requires several seconds of CPU time. A total of 34.7 CPU seconds are used to optimize these five runs. In this case, however, all 5 optimization runs led to the same optimal configuration (mod). The optimal flap setting chosen by the optimizer is $\delta_f = 38.5°$, $gap_f = 1.69\%c$, $ol_f = 0.97\%c$, and $\alpha = 10.0°$. The flap deflection and angle of attack are at the upper bounds, whereas, the other two design variables are free variables (the variable lies between the upper and lower bound).

The accuracy of the neural networks' prediction is tested for both the initial and modified configurations by generating the appropriate grid and computing the INS2D solution. Then the predicted and computed C_l are compared and the percent difference ($\Delta\%$) is shown in Table 6.2. Run 5-A has zero error in predicting the original C_l. Runs 5-B and 5-C have errors of -0.28% and 0.31%, respectively. However, Runs 5-D and 5-E have prediction errors greater than the acceptable error. The maximum lift for the optimized configuration predicted by the neural network is $C_l = 4.27$, however, the actual lift coefficient predicted by INS2D is $C_l = 4.11$. The neural network overpredicted C_l by 3.82%. For these reasons, the neural networks should probably be trained with a data set that contains more information on the aerodynamics of this airfoil. Also, the pressure difference of the modified configuration, $C_{p_{diff}} = -15.7$, is much higher than the value of $C_{p_{diff}} = -13.0$ which is used to predict maximum lift. This will have an negative impact on the accuracy of prediction of the neural networks. The aerodynamics of this configuration is post-stall (or post maximum lift) and the neural networks are trained with data only containing information on the aerodynamics up to maximum lift. Since the bound on angle of attack is chosen to be an average value of where maximum lift occurs, it may influence the region where the optimizer is forced to look for the maximum. This will be discussed in greater detail in this chapter. Also, a procedure that is developed and tested to correct this is dis-

Table 6.1 Design Space for $\delta_s = 6.0$ degrees

Design Variables	Lower Bound	Upper Bound
δ_f	25.0	38.5
gap_f	1.5	2.7
$overlap_f$	0.4	1.5
α	0	10

cussed.

To get a better understanding of the flow physics, the pressure distribution of the modified and original configurations for optimization Run 5-B are examined. Figure 6.25a shows the

Table 6.2 Optimization Results for δ_s = 6 degrees with Method 5 as the Training Set

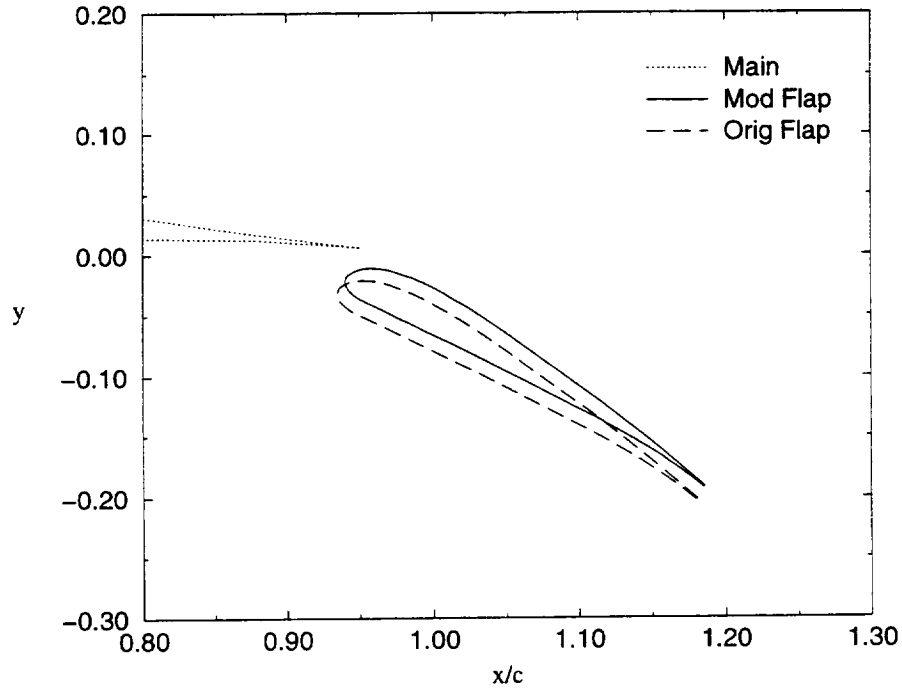
Run	DV	orig	mod	C_l *orig NN*	C_l *orig INS2D*	Δ% *orig*	C_l *mod NN*	C_l *mod INS2D*	Δ% mod	$\Delta C_{p\ diff}$ *mod INS2D*	CPU (sec)
5-A	δ_f	25.0	38.5	2.04	2.04	0.0	4.27	4.11	3.82	-15.7	6.03
	gap_f	1.5	1.69								
	ol_f	0.4	0.97								
	α	0.0	10.0								
5-B	δ_f	38.5	38.5	3.55	3.56	-0.28	4.27	4.11	3.82	-15.7	4.41
	gap_f	2.7	1.69								
	ol_f	1.5	0.97								
	α	10.0	10.0								
5-C	δ_f	32.0	38.5	3.21	3.20	0.31	4.27	4.11	3.82	-15.7	9.32
	gap_f	2.1	1.69								
	ol_f	0.95	0.97								
	α	5.0	10.0								
5-D	δ_f	30.0	38.5	3.06	2.96	3.30	4.27	4.11	3.82	-15.7	5.82
	gap_f	1.9	1.69								
	ol_f	0.75	0.97								
	α	4.0	10.0								
5-E	δ_f	27.0	38.5	2.53	2.47	2.40	4.27	4.11	3.82	-15.7	9.13
	gap_f	2.1	1.69								
	ol_f	0.5	0.97								
	α	2.0	10.0								

modified and original flap positions in relation to the main element trailing edge. The flap deflections are the same, but the gap and overlap are smaller for the modified flap setting. Figure 6.25b shows the pressure distribution of the slat, main, and flap elements in a solid line for the modified configuration and a dashed line for the original configuration. The basic shape of the C_p curves are similar for slat and main elements for both configurations. The flow is attached for both the slat and main elements. The suction pressure on the modified slat and main elements are clearly larger than the original configuration resulting in greater lift. However, there are greater differences between the original and modified flap elements. The spike at the trailing-edge and the multiple spikes at the leading-edge are representing the actual flow physics as was discussed earlier (refer to Figure Figure 6.23b). When designing the high-lift system, the goal is to achieve maximum lift without causing separation. The highest lift will not occur with flow separation on the elements. Figure 6.25 shows the original flap element to be separated, thus the airfoil is not capable of holding maximum loads. The modified flap, however, is attached and allows more lift to be carried by all elements.

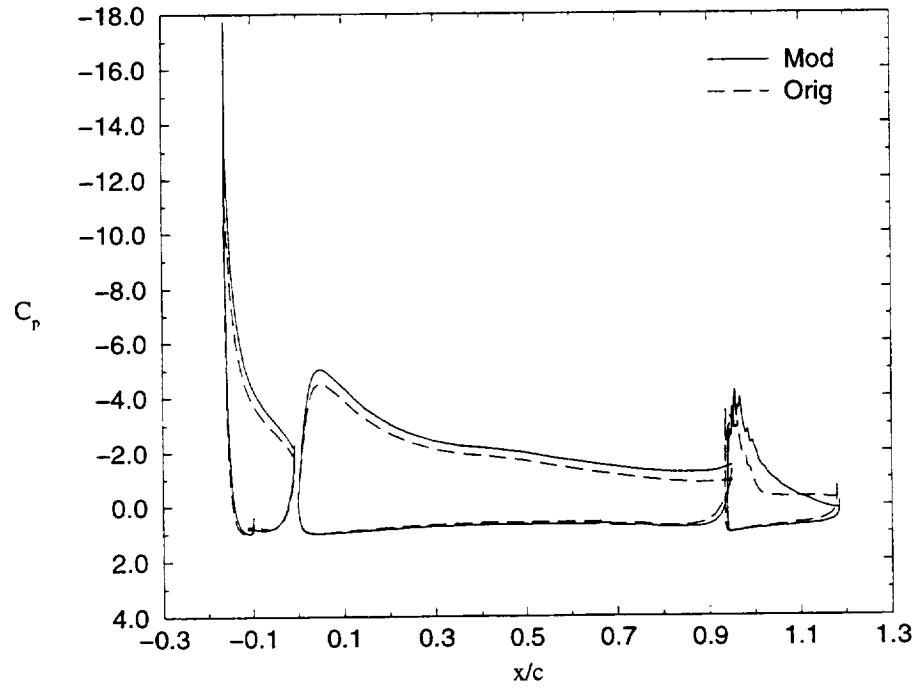
6.4.2 Optimization with Method 8 as Training Set

Next, Method 8 which contains 70% of the entire training set is used to train the neural networks that are integrated with the NPSOL optimizer to see if the accuracy can be improved. As is discussed above, this training method has low prediction errors and would save 30% of computer resources when compared with Method 1. The optimization results are shown in Table 6.3 for five different optimization runs. The five optimization runs are started with the same initial design variables as the previous optimization runs. As shown in Table 6.3, there are two different values of maximum lift that are found in these five runs. Runs 8-A and 8-E found the maximum lift coefficient to be $C_l = 3.77$ for the flap rigging of $\delta_f = 30.7°$, $gap_f = 2.7\%c$, and $ol_f = 0.74\%c$ at $\alpha = 10.0°$. This lift coefficient is only an improvement from the original and not the global maximum. The best improvement of the lift coefficient has a modified value of is $C_l = 4.18$ and the corresponding modified design variables are $\delta_f = 38.5°$, $gap_f = 1.74\%c$, and $ol_f = 0.4\%c$ at $\alpha = 10.0°$. In this case, the flap deflection and angle of attack are at the upper bound and overlap is at the lower bound. The modified gap is a free variable. The flap setting is shown in Figure 6.26a for the case with the highest improvement and is compared to the original flap setting. The flap deflection is the same as the original but the gap and overlap are smaller. The pressure coefficients on the surface of the airfoils are plotted in Figure 6.26b for the modified and original configurations for optimization Run 8-B. Similar features and trends are shown here as was discussed in the previous results. The pressure distribution again has larger suction pressure for the modified elements than the original. There are only slight changes on the pressure distribution of the bottom surface. More importantly, the modified flap flow is attached whereas the flow on the original flap is separated. This results in the modified airfoil being able to carry more loads and produce more lift.

The accuracy of the neural network to predict the C_l of the original configurations did improve for Method 8 as shown in Table 6.3. The error in the prediction for the original geometry is good and is within the acceptable error. As is seen, the error is as low as 0.10% and the highest error is only 1.53%. The neural-net-predicted-modified C_l has a slightly higher error but is also within the acceptable range (2% and lower). Here, the highest is 1.95% and the lowest is 0.76%. This is very good considering that only 70% of the training data is used to train the neural networks to predict the aerodynamics of the multi-element airfoil and that the CPU time is



a) Optimized Flap Rigging



b) Pressure Distribution

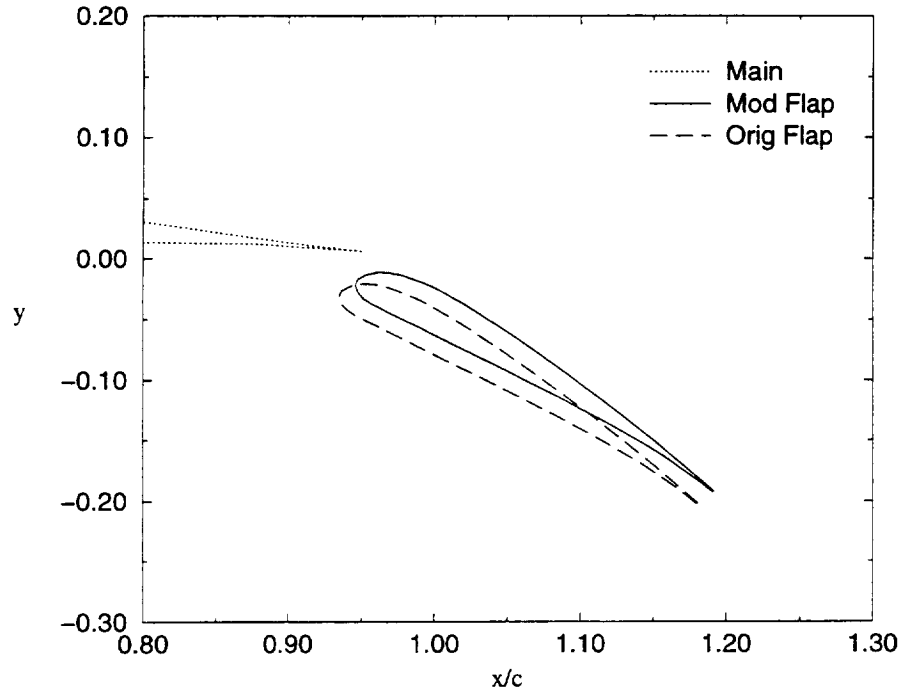
Figure 6.25 Optimization results for Run 5-B with modified configuration of $\delta_f = 38.5°$, $gap_f = 1.69\%c$, $ol_f = 0.97\%c$ and original configuration of $\delta_f = 38.5°$, $gap_f = 2.7\%c$, $ol_f = 1.5\%c$ at $\alpha = 10.0°$.

only a few seconds (also shown in Table 6.3). Once the neural networks are trained, this optimization procedure can be performed with about 5 seconds of CPU time on a SGI workstation with R4000 processor. To optimize all five cases, less than 25 seconds of CPU time is required.

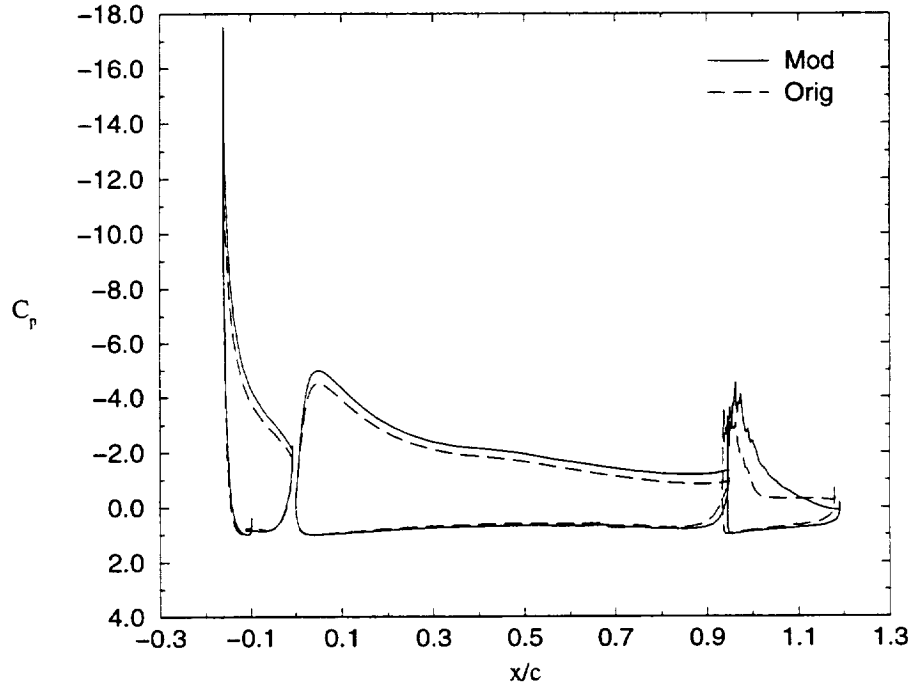
6.4.3 Optimization with Method 9 as Training Set

Table 6.3 Optimization Results for δ_s = 6 degrees with Method 8 as the Training Set

Run	DV	orig	mod	C_l *orig NN*	C_l *orig INS2D*	Δ% *orig*	C_l *mod NN*	C_l *mod INS2D*	Δ% mod	ΔC_p *diff mod INS2D*	CPU (sec)
8-A	δ_f	25.0	30.7	2.03	2.03	-0.29	3.77	3.74	0.76	-13.08	4.81
	gap_f	1.50	2.70								
	ol_f	0.40	0.74								
	α	0.0	10.0								
8-B	δ_f	38.5	38.5	3.52	3.56	-1.15	4.18	4.10	1.95	-13.42	3.91
	gap_f	2.70	1.74								
	ol_f	1.50	0.40								
	α	10.0	10.0								
8-C	δ_f	32.0	38.5	3.15	3.20	-1.53	4.18	4.10	1.95	-13.42	4.87
	gap_f	2.10	1.74								
	ol_f	0.95	0.40								
	α	5.0	10.0								
8-D	δ_f	30.0	38.5	2.96	2.96	0.10	4.18	4.10	1.95	-13.42	6.67
	gap_f	1.90	1.74								
	ol_f	0.75	0.40								
	α	4.0	10.0								
8-E	δ_f	27.0	30.7	2.5	2.47	1.17	3.77	3.74	0.80	-13.08	4.64
	gap_f	2.10	2.70								
	ol_f	0.50	0.74								
	α	2.0	10.0								



a) Optimized Flap Rigging



b) Pressure Distribution

Figure 6.26 Optimization results for Run 8-B with modified configuration of $\delta_f = 38.5°$, $gap_f = 1.74\%c$, $ol_f = 0.4\%c$ and original configuration of $\delta_f = 38.5°$, $gap_f = 2.7\%c$, $ol_f = 1.5\%c$ at $\alpha = 10.0°$.

Method 9 was also found to be a good method in training the neural networks. Thus, the optimization procedure is next performed with Method 9 to train the neural networks. The same initial design variable values are used as in the previous study as shown in Table 6.4. Again, two different local maximums are found that have improved lift coefficients. The smallest local

Table 6.4 Optimization Results for δ_s = 6 degrees with Method 9 as the Training Set

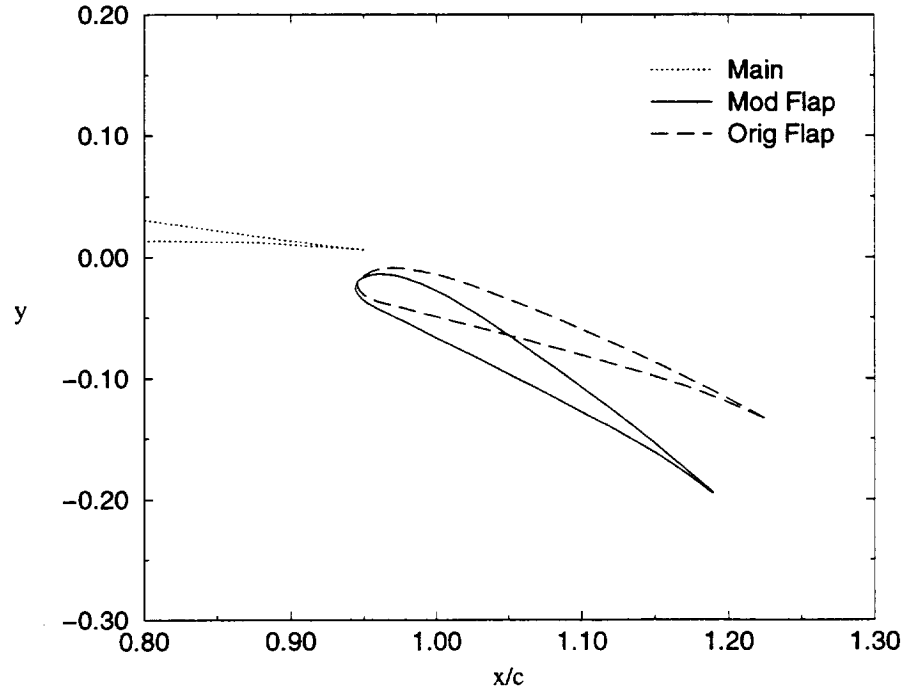
Run	DV	orig	mod	C_l *orig NN*	C_l *orig INS2D*	Δ% *orig*	C_l *mod NN*	C_l *mod INS2D*	Δ% mod	ΔC_p *diff mod INS2D*	CPU (sec)
9-A	δ_f	25.0	38.5	2.04	2.04	0.0	4.13	4.03	-2.56	-14.8	6.9
	gap_f	1.5	2.01								
	ol_f	0.4	0.558								
	α	0.0	10.0								
9-B	δ_f	38.5	38.5	3.54	3.56	-0.56	4.11	4.00	-2.72	-14.5	3.3
	gap_f	2.7	2.04								
	ol_f	1.5	1.5								
	α	10.0	10.0								
9-C	δ_f	32.0	38.5	3.19	3.20	-0.31	4.11	4.00	-2.72	-14.5	6.9
	gap_f	2.1	2.04								
	ol_f	0.95	1.5								
	α	5.0	10.0								
9-D	δ_f	30.0	38.5	3.02	2.96	2.03	4.11	4.00	-2.72	-14.5	5.5
	gap_f	1.9	2.04								
	ol_f	0.75	1.5								
	α	4.0	10.0								
9-E	δ_f	27.0	38.5	2.51	2.47	1.62	4.11	4.00	-2.72	-14.5	6.0
	gap_f	2.1	2.04								
	ol_f	0.5	1.5								
	α	2.0	10.0								

maximum is found using the initial values of the design variables of Runs 9-B through 9-E. The modified high-lift rigging is $\delta_f = 38.5°$, $gap_f = 2.04\%c$, $ol_f = 1.5\%c$, and $\alpha = 10.0°$ (Figure 6.27a) and has $C_l = 4.11$. The largest improvement in C_l for this particular study is just slightly higher at $C_l = 4.13$. The modified values of the design variables for this case are $\delta_f = 38.5°$, $gap_f = 2.01\%c$, $ol_f = 0.56\%c$, and $\alpha = 10.0°$ (Figure 6.27a). The flap deflection for both instances is optimal at the upper bound. The modified gaps are free variables and close to each other, whereas the overlaps are quite different. The smallest local maximum has the overlap at the upper bound whereas the larger local maximum has it as a free variable. Both configurations have the angle of attack to be optimal at the upper bound. The pressure distributions are plotted in Figure 6.27b. The original configuration was initially at $\alpha = 0.0°$ (plotted in a dotted line) but in order to compare the pressure distributions the original configuration result is also plotted at $\alpha = 10.0°$ (in a dashed line). The pressure distribution for the original airfoil at $\alpha = 0.0°$ has small suction pressures on the slat and main elements. The modified airfoil has larger suction pressure peaks than the original configuration which accounts for the additional lift that is created. The original configuration has separated flow on the flap element, whereas the modified flap has attached flow. Thus, the modified airfoil is capable of withstanding more loads and thus has higher lift.

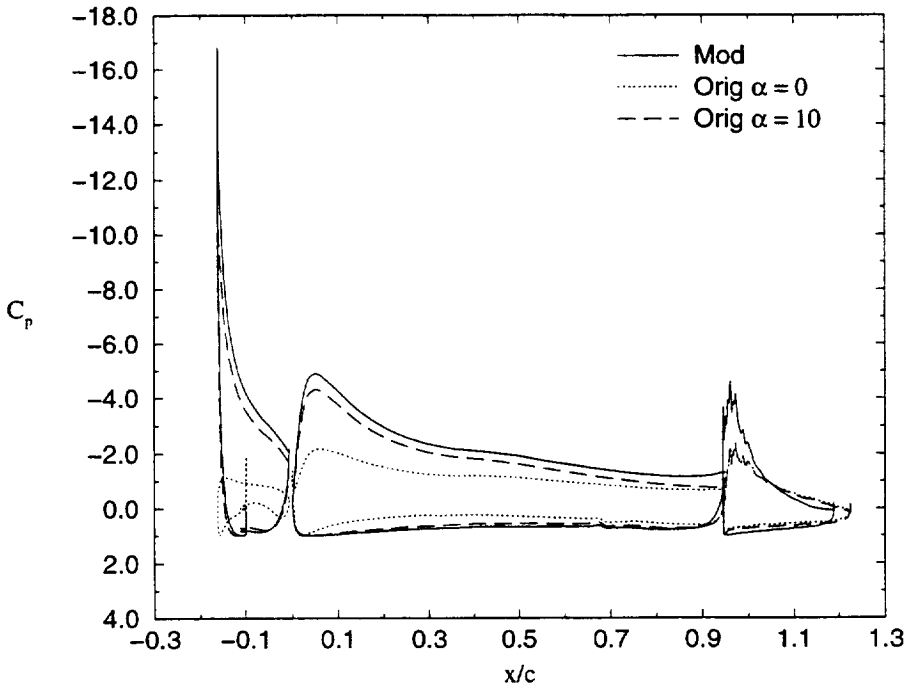
The errors in the accuracy of the prediction are higher for Method 9 than Method 8 eventhough Method 9 contains 74% of the entire data whereas Method 8 contains 70%. The initial configurations again have lower errors than the modified configurations. In Run 9-A there is zero error and only one case has an error greater than 2%. All of the modified configurations have prediction errors greater than 2%. The pressure difference rule is applied to these cases. Examining the outcome, it is shown that the pressure difference exceeds the allowable value of $C_{p_{diff}} = -13.0$. All the pressure differences are equal to or greater than $C_{p_{diff}} = -14.5$. This is also seen in Figure 6.27b where the pressure difference for the slat is quite high. Thus, the angle of attack upper bound is set too high and $\alpha = 10.0°$ for these particular configurations is beyond maximum lift as defined in this current study. Thus, the neural networks are not properly trained to predict the aerodynamics in this range. This did not occur in the previous optimization case that uses Method 8 to train the neural networks. The configurations that are found to be optimal using Method 8 have pressure differences just slightly greater than $C_{p_{diff}} = -13.0$.

6.4.4 Optimization of Twenty-Six Degree Deflected Slat

The optimization procedure using neural networks is also applied to the twenty-six-degree-deflected slat training data. The flap deflection and angle of attack bounds are different since the range of the flap deflection angles are higher as shown in Table 6.5. In some optimization runs, the upper bound on the angle of attack is increased to 20 degrees because the pressure difference at $\alpha = 10.0°$ is much lower than $C_{p_{diff}} = -13.0$ (this was determined during the optimization process). The results for this data set are shown in Table 6.6. Runs 1-26A and 1-26B use Method 1 to train the neural networks and the bounds on angle of attack are ten and twenty degrees, respectively. Likewise, Runs 9-26A and 9-26B are trained with Method 9 and again the bounds on the angle of attack are ten and twenty degrees, respectively. All optimization runs are started with the design variables set to the average of the bounds.



a) Optimized Flap Setting



b) Pressure Distribution

Figure 6.27 Optimization result for Run 9-A with modified configuration of δ_f = 38.5°, gap_f = *2.01%c*, ol_f = *0.56%c*, and α = 10.0° and original configuration of δ_f = 25.0°, gap_f = *1.5%c*, ol_f = *0.4%c*.

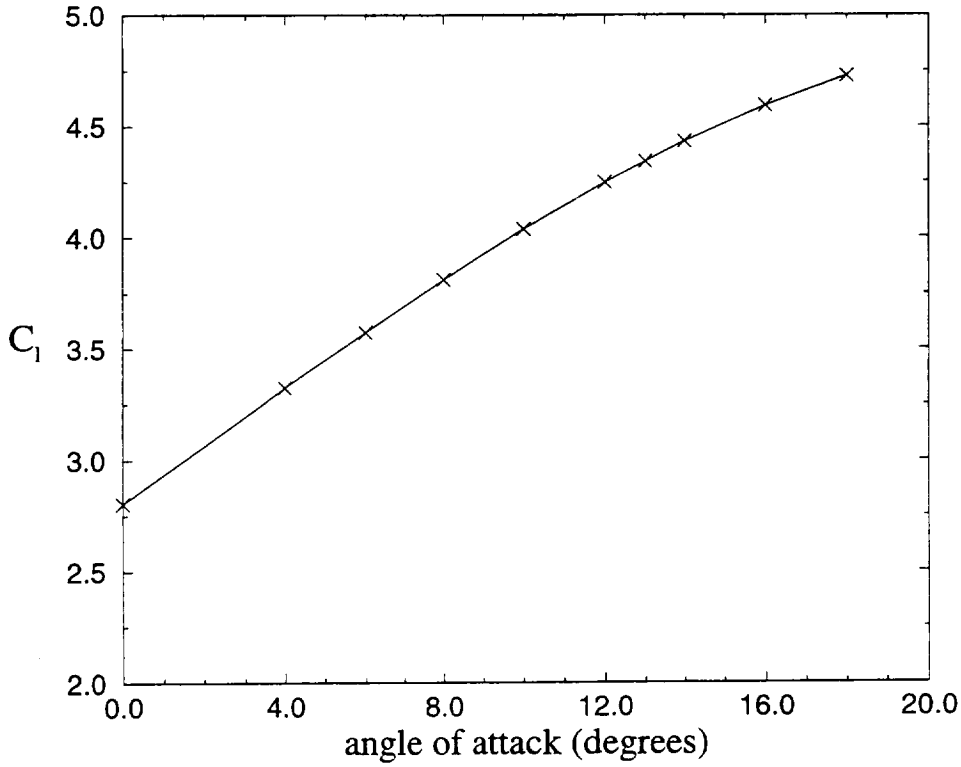


Figure 6.28 Lift coefficient versus angle of attack for optimization Run 1-26B; $\delta_f = 38.5°$, $gap_f = 2.1\%c$, and $ol_f = 0.4\%c$.

First, when the angle of attack upper bound is equal to ten degrees, the configurations that are chosen are different for each training method as shown in Table 6.6. The pressure difference for these cases are small around $C_{p_{diff}} = -3.5$. For this slat deflection, $\alpha = 10.0°$ is not near maximum lift but is still in the linear range as shown in Figure 6.28. Thus, the lift coefficients are 14% lower than the cases where the bound on the angle of attack is increased.

Second, the bound on the angle of attack is increased to $\alpha = 20.0°$ and both optimization

Table 6.5 Bounds of Design Variables for $\delta_s = 26.0$ degrees

Design variables	Lower Bound	Upper Bound
δ_f	38.5	53.0
gap_f	1.5	2.7
$overlap_f$	0.4	1.5
α	0	10 or 20

runs 1-26B and 9-26B found the same configuration to be optimal, $\delta_f = 38.5°$, $gap_f = 1.5\%c$, $ol_f = 1.4\%c$, and $\alpha = 18°$. Each run predicts a different value of $C_{l_{max}}$. The C_l calculated by INS2D is 4.73 but optimization Run 1-26B predicted $C_l = 4.70$ which has an error of -0.63%. On the other hand, Run 9-26B predicted the modified lift coefficient to be $C_l = 4.77$ which is overpredicted by 0.85%. The modified angle of attack is $\alpha = 18.0°$ which is a free variable. This is the first case where angle of attack is not optimal at the upper bound. The pressure difference for this configuration is $C_{p_{diff}} = -13.2$. This means that the optimizer did predict a configuration near maximum lift. The computed lift coefficient is plotted against the angle of attack in Figure 6.28. The lift curve appears to be bending over at the higher angles of attack.

Table 6.6 Optimization Results for $\delta_s = 26$ degrees

Run	DV	orig	mod	C_l *orig NN*	C_l *orig INS2D*	$\Delta\%$ *orig*	C_l *mod NN*	C_l *mod INS2D*	$\Delta\%$ mod	ΔC_p *diff mod INS2D*	CPU (sec)
1-26A	δ_f	32.0	38.5	2.59	2.50	3.6	4.04	4.04	0.05	-3.52	4.9
	gap_f	2.10	1.50								
	ol_f	0.95	0.40								
	α	5.0	10.0								
1-26B	δ_f	32.0	38.5	2.59	2.50	3.6	4.70	4.73	-0.63	-13.2	6.0
	gap_f	2.10	2.10								
	ol_f	0.95	0.40								
	α	5.0	18.0								
9-26A	δ_f	32.0	38.5	2.65	2.50	6.0	4.03	4.01	0.50	-3.47	6.3
	gap_f	2.10	1.62								
	ol_f	0.95	1.50								
	α	5.0	10.0								
9-26B	δ_f	32.0	38.5	2.65	2.50	6.0	4.77	4.73	0.85	-13.2	3.9
	gap_f	2.10	2.10								
	ol_f	0.95	1.40								
	α	5.0	18.0								

The optimizer chose the flap deflection to be optimal at $\delta_f = 38.5°$ for both the six-degree and twenty-six degree slat deflection airfoil. For the smaller slat deflection, $\delta_f = 38.5°$ was the upper bound and for the larger slat deflection it is the lower bound. This shows that the higher flap settings are not optimal since at some point increasing the flap deflection will degrade performance. Run 1-26A predicts the maximum lift to occur at a flap setting of $\delta_f = 38.5°$, $gap_f = 1.5\%c$, $ol_f = 0.4\%c$, and $\alpha = 10°$. The gap is the same as in Runs 1-26B and 9-26B and the overlap is lower and is optimal at the lower bound. On the other hand, Run 9-26A has the following optimal setting, $\delta_f = 38.5°$, $gap_f = 1.62\%c$, $ol_f = 1.5\%c$, and $\alpha = 10.0°$. The gap and overlap are slightly higher for this optimization run.

The prediction errors are high for the lift coefficients of the original configurations but are excellent for the lift coefficients of the modified configurations. The prediction errors of the modified configurations range from 0.0 - 0.85%. The neural network did accurately predict the modified lift coefficients. The CPU time required to optimize all four cases was only 21.17 seconds.

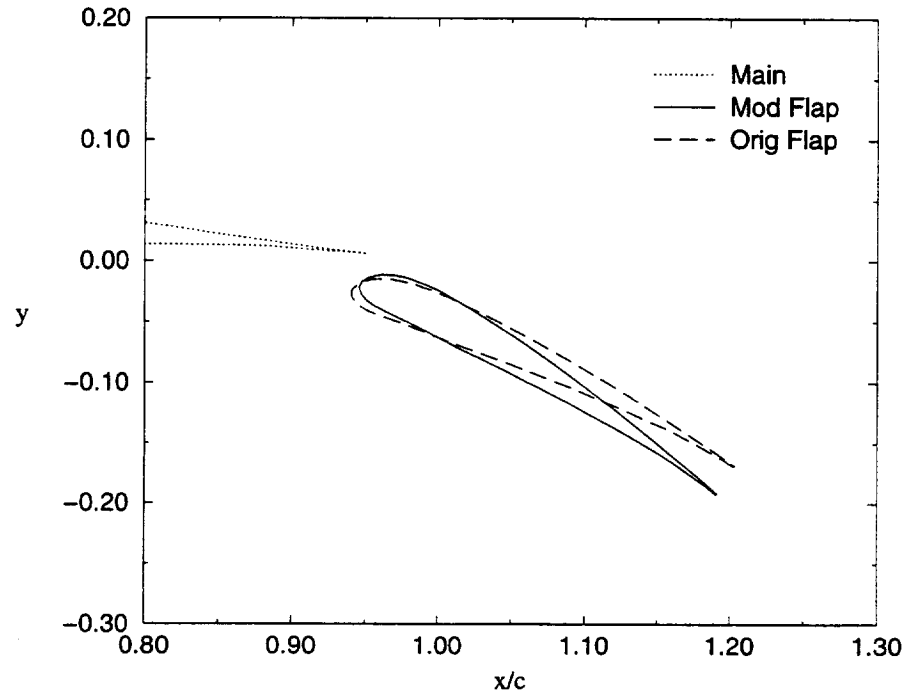
The original and modified flap settings are shown in Figure 6.29a for optimization Run 1-26B. The modified flap has a larger deflection angle, a smaller overlap, and has the same gap as the original setting. The pressure distribution for Run 1-26B is shown in Figure 6.29b. The pressure distribution shows that the modified flap has attached flow and the original flap has separated flow. Again, the modified configuration has larger suction pressure on all elements which also contributes to the greater lift. Comparing Figure 6.29 with Figure 6.27, shows the differences in the C_p curve for the different slats. In the higher-deflected slat, the slat is working well and inducing high C_p on the main element which results in higher lift.

6.4.5 Optimization With Large Design Space

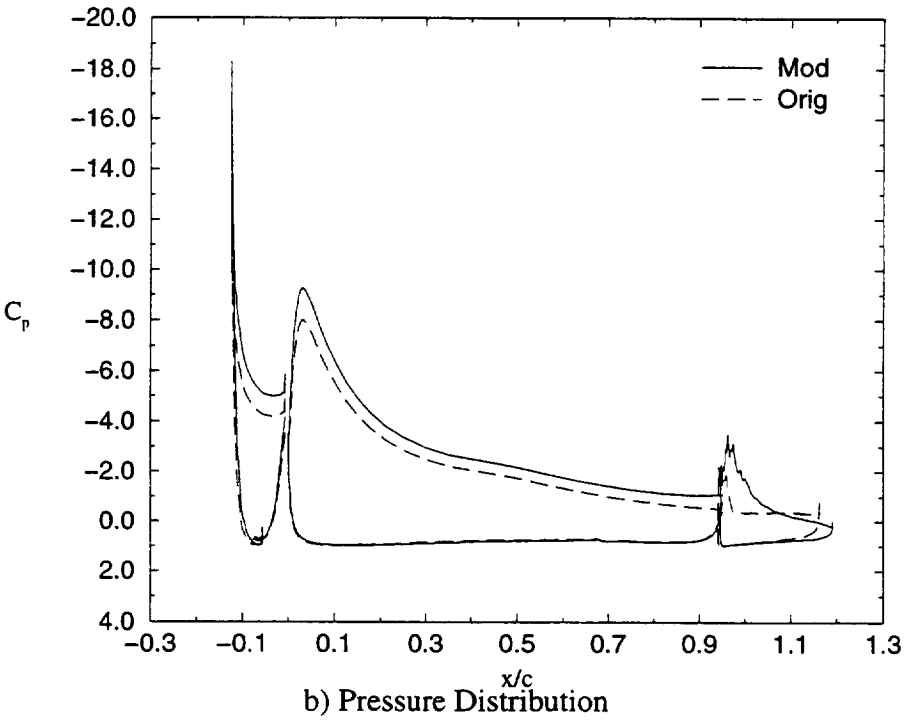
The next optimization study that is conducted is to train the neural networks with a larger design space for the six-degree deflected slat. The design space was increased to include deflection angles between $\delta_f = 25.0°$ and $\delta_f = 53.0°$. The gap and overlap bounds remained the same as shown in Table 6.7. The neural networks are trained with 45 different configurations including combinations of the following: $\delta_f = 25.0$, *29.0, 38.5, 49.0,* and *53.0* degrees; gap_f = *1.5, 2.1,* and *2.7%c;* ol_f = *0.4, 1.0,* and *1.5%c.* The pressure difference rule is applied to the data set and the neural networks are trained with 250 iterations as before. Since the higher flap

Table 6.7 Design Variable Bounds for $\delta_s = 6.0$ with 5 values of flap deflection

Design variables	Lower Bound	Upper Bound
δ_f	25.0	53.0
gap_f	1.5	2.7
$overlap_f$	0.4	1.5
α	0	10



a) Optimized Flap Setting



b) Pressure Distribution

Figure 6.29 Optimization results for Run 1-26B with modified configuration of $\delta_f = 38.5°$, $gap_f = 2.1\%c$, $ol_f = 0.4\%c$ and original configuration of $\delta_f = 32.0°$, $gap_f = 2.\%c1$, $ol_f = 0.95\%c$ at $\alpha = 18°$.

deflections are greater than the common flight envelope, the optimizer is tested in choosing the optimal position. The optimization results for this study are shown in Table 6.8. The same five initial conditions are used as in the previous studies. The optimization produced three different optimal flap configurations which were found to have maximum C_l. The smallest local maxi-

Table 6.8 Optimization Results for δ_s = 6 degrees with large design space and Method 1 as the Training Set

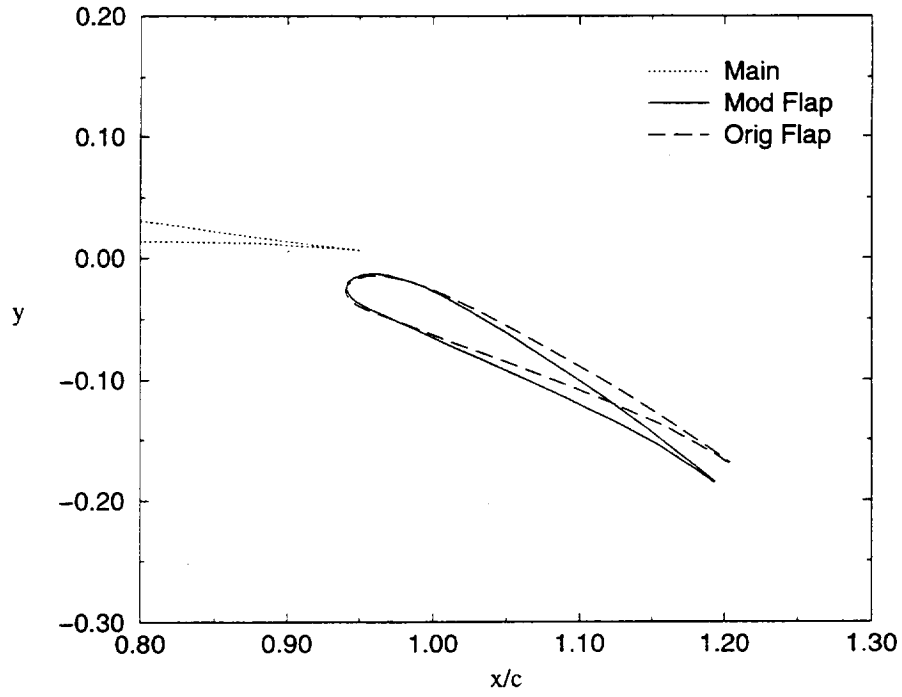
Run	DV	orig	mod	C_l *orig NN*	C_l *orig INS2D*	Δ% *orig*	C_l *mod NN*	C_l *mod INS2D*	Δ% mod	$\Delta C_{p\ diff}$ *mod INS2D*	CPU (sec)
1 -5A	δ_f	25.0	38.5	2.04	2.04	0.00	3.76	3.74	0.53	-13.1	8.38
	gap_f	1.50	2.52								
	ol_f	0.40	0.60								
	α	0.0	10.0								
1 -5B	δ_f	38.5	53.0	3.13	3.13	0.0	3.34	3.33	0.30	-11.7	4.82
	gap_f	2.70	1.5								
	ol_f	1.50	1.21								
	α	10.0	10.0								
1 -5C	δ_f	32.0	36.2	3.29	3.20	2.81	4.14	4.01	3.24	-14.9	9.5
	gap_f	2.10	1.92								
	ol_f	0.95	0.89								
	α	5.0	10.0								
1 -5D	δ_f	30.0	36.2	3.06	2.96	3.34	4.14	4.01	3.24	-14.9	6.8
	gap_f	1.90	1.92								
	ol_f	0.75	0.89								
	α	4.0	10.0								
1 -5E	δ_f	27.0	36.2	2.51	2.47	1.62	4.14	4.01	3.24	-14.9	7.58
	gap_f	2.10	1.92								
	ol_f	0.50	0.89								
	α	2.0	10.0								

mum has the following setting $\delta_f = 53.0°$, $gap_f = 1.5\%c$, $ol_f = 1.21\%c$, and $\alpha = 10.0°$ with $C_l = 3.33$. This is clearly not the best improvement and is surprising that the optimizer found this to be the maximum since $\delta_f = 53.0°$ has highly separated flow on the flap. The next largest local maximum chosen is $\delta_f = 38.5°$, $gap_f = 2.52\%c$, $ol_f = 0.60\%c$, and $\alpha = 10.0°$ giving $C_l = 3.74$. Three runs chose the optimal position to be $\delta_f = 36.2°$, $gap_f = 1.92\%c$, $ol_f = 0.89\%c$, and $\alpha = 10.0°$ (Figure 6.30a) with $C_l = 4.013$. This maximum has three design variables as free variables and as expected the angle of attack is at the upper bound. The pressure difference is $C_{p_{diff}} = -14.9$ which is slightly greater than the value used to predict maximum lift. For this reason, the neural network prediction error is high and is above 3%. The CPU time that is used to optimize these five runs is 37.04 seconds. The pressure distribution is shown in Figure 6.30b. The original configuration is shown for $\alpha = 5°$ and $\alpha = 10°$. The original configuration at the higher angle of attack has a pressure distribution that is similar to the modified configuration. The modified configuration does have slightly higher suction pressure peak than the original configuration, thus it has greater lift. In this case, the flow on both the original and modified flaps have separated flow.

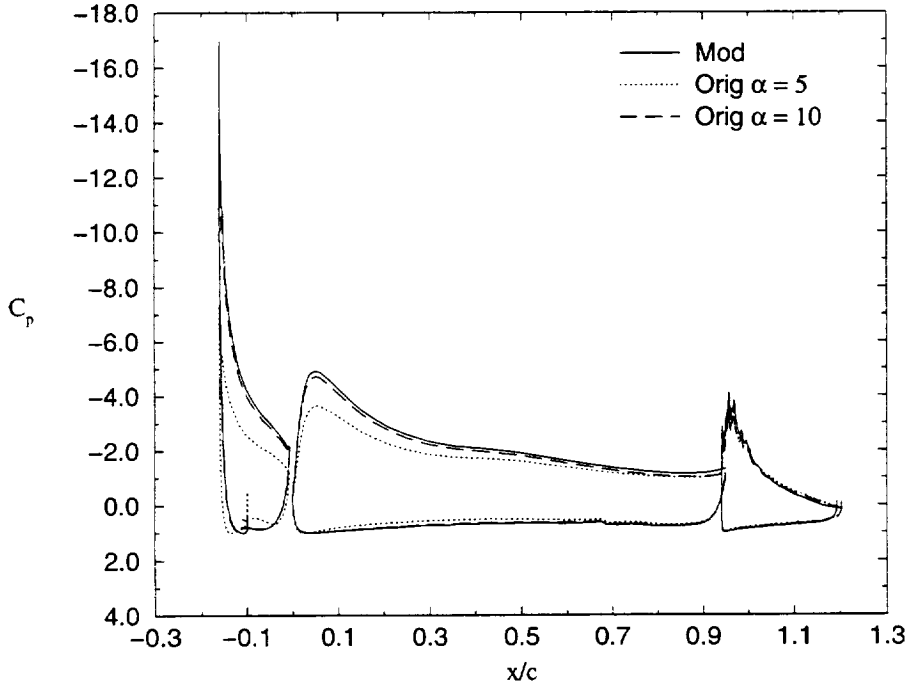
6.4.6 Constrained Optimization

In order to test whether the accuracy would get better if the modified configurations were restricted within the empirically predicted pre-stall range, the upper bound on the angle of attack design variable is removed. Instead a constraint is placed on the value of the pressure difference, $C_{p_{diff}} \geq -13.0$. An additional neural network is trained with flap deflection, gap, overlap, and angle of attack to predict the pressure difference. In this case, the entire training data is used to predict $C_{p_{diff}}$, whereas the neural networks that predict the aerodynamic coefficients are trained with data only including pre-stall data that is predicted by the pressure difference rule. The design variables of the optimization runs remain the same as does the objective function. The results of the case that found the best improvement by the optimizer is shown in Table 6.9 for Run 9-C-ΔC_p. The modified design variables are $\delta_f = 37.5°$, $gap_f = 2.08\%c$, $ol_f = 0.40\%c$, and $\alpha = 9.0°$. As expected, the modified angle of attack is lower than in the previous case that specified the upper bound to be $\alpha = 10.0°$. The neural network predicted the pressure difference value to be exactly what is calculated with the INS2D solution. The neural network predicted the modified lift coefficient to be over 2% of the actual INS2D value.

To further reduce the prediction error in the modified lift coefficient, the INS2D data from this optimal case is added to the training data. The neural networks are then re-trained with this additional information in hope that it will improve the accuracy. Again, the neural network that predicts the lift coefficient is trained with the data set that includes the data points that are at or below the maximum lift. The neural network that predicts the pressure difference is trained with the entire training set. The optimization runs are again constrained and the best improvement is shown in Table 6.9 denoted by Run 9-C-opt. The values of the modified design variables are different for the flap deflection, gap, and angle of attack and are the same for the overlap as in the previous case. The modified lift coefficient predicted by the neural network happens to be the same as in the previous optimization run, however, the INS2D value of the modified coefficient is different and the error is reduced to only 0.51%. Thus, by constraining the design space that the optimizer is allowed to search and by adding one data point near maximum lift to the training data, the prediction error is reduced and all constraints are met. The predicted and actual pressure difference are close and differ by only 0.4. It should be noted that



a) Optimized Flap Rigging



b) Pressure Distribution

Figure 6.30 Optimization results for Run 1-5C with modified configuration of $\delta_f = 36.2°$, $gap_f = 1.92\%c$, $ol_f = 0.89\%c$, and $\alpha = 10°$ and original configuration of $\delta_f = 32.0°$, $gap_f = 2.1\%c$, $ol_f = 0.95\%c$.

the CPU time required to run a constraint optimization run is increased, however, it is still less than 28 seconds as shown in Table 6.9.

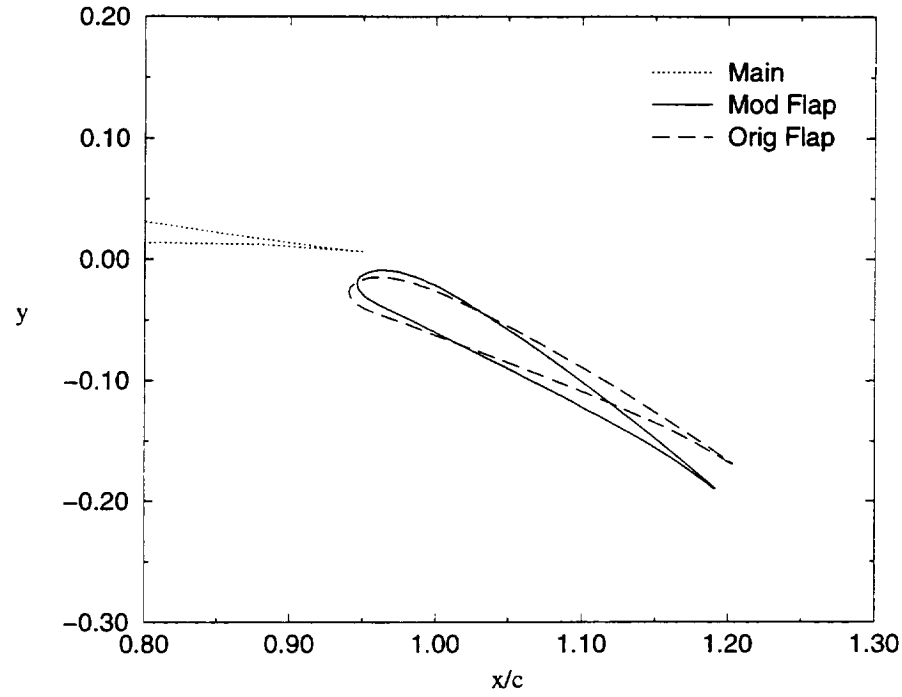
To get a better understanding of the flow physics, the pressure distribution of the modified and original configurations for optimization Run 9-C-opt are examined. Figure 6.31a shows the modified and original flap positions in relation to the main element trailing edge. Figure 6.31b shows the pressure distribution of the slat, main, and flap elements in a solid line for the modified configuration. The original configuration was initially at $\alpha = 5.0°$ (plotted in a dotted line) but in order to compare the pressure distributions, the original configuration is also plotted at $\alpha = 8.3°$ (in a dashed line). The basic shape of the C_p curves are similar for all elements for both configurations. The flow is attached for all elements. The suction pressure on the modified elements are clearly larger than the original configuration resulting in greater lift. Again, there are interesting features on the original and modified flap elements that are discussed in a previous case.

6.4.7 Summary of Optimization Runs

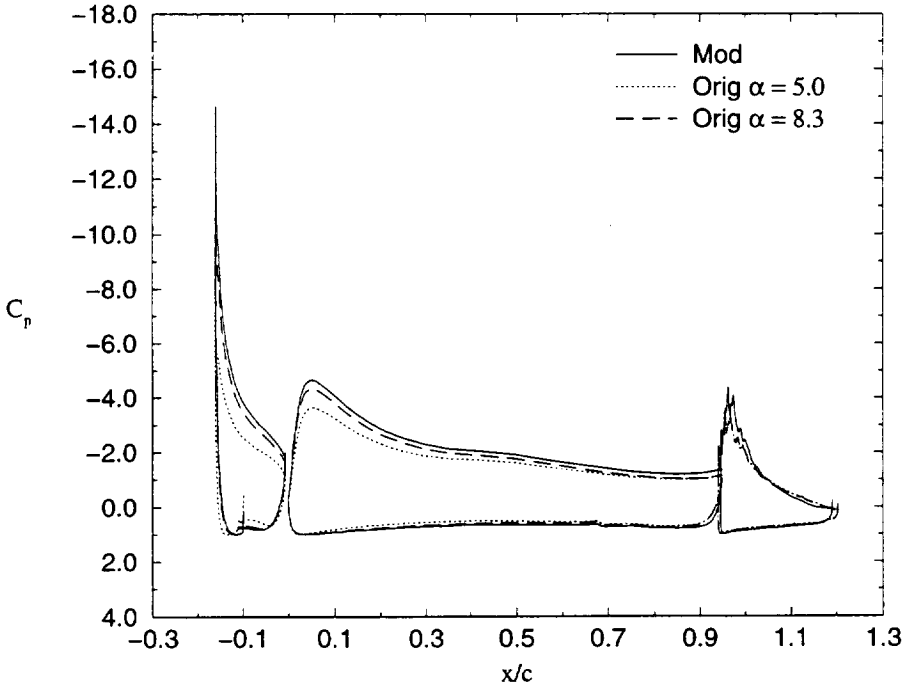
In summary, the neural network does accurately predict the lift coefficient for training

Table 6.9 Constrained Optimization Results for Method 9 as the Training Set

Run	DV	orig	mod	C_l *orig NN*	C_l *orig INS2D*	Δ% *orig*	C_l *mod NN*	C_l *mod INS2D*	Δ% mod	ΔC_p *diff mod NN*	ΔC_p *diff mod INS2D*	CPU (sec)
9-C-ΔC_p	δ_f	32.0	37.5	3.19	3.20	-0.31	3.94	3.86	2.07	-13.0	-13.0	27.3
	gap_f	2.10	2.08									
	ol_f	0.95	0.40									
	α	5.0	9.0									
9-C-opt	δ_f	32.0	38.5	3.18	3.20	-0.63	3.94	3.92	0.51	-13.0	-13.4	26.1
	gap_f	2.10	1.5									
	ol_f	0.95	0.4									
	α	5.0	8.30									



a) Optimized Flap Setting



b) Pressure Distribution

Figure 6.31 Optimization results for Run 9-C-opt (flap settings denoted in Table 6.9).

Methods 8 and 9, however, Method 5 has high prediction errors. The optimizer and neural networks are successfully integrated to predict maximum lift within the specified design space. Only Method 5 predicted the same maximum for the five different starting locations of the $\delta_s = 6.0°$ design space. All other design methods predicted different maximum lift coefficients. This procedure does not guarantee a global maximum, but instead an improvement from the original objective function. Thus, it is important to search the design space in several locations to search for the best improvement as is illustrated. Using the empirical constraint together with an iterative optimization process which re-inserted the optimized configuration into the training data set and repeated the optimization reduced the prediction error. Method 5 predicted the largest maximum lift coefficient of $C_{l_{INS2D}} = 4.11$ for $\delta_s = 6.0°$, $\delta_f = 38.5°$, $gap_f = 1.69\%c$, $ol_f = 0.97\%c$, and $\alpha = 10.0°$. The maximum lift coefficient for the twenty-six degree slat is $C_{l_{INS2D}} = 4.73$ for $\delta_f = 38.5°$, $gap_f = 2.1\%c$, $ol_f = 0.4\%c$, and $\alpha = 18.0°$.

6.5 Benefits of New Process

The aerodynamic design space of a multi-element airfoil is very complex and may have many local maximums and minimums. When a gradient-based optimizer is used to search the design space, many starting points need to be examined in order to find the best improvement. The advantage of using neural networks in the optimization process versus the traditional optimization process (Figure 6.32) is the turn around time and the CPU time that is saved for many optimization runs. In the traditional optimization process, every time that the design variables are perturbed, the gradient needs to be calculated to determine the search direction. In order to calculate the gradient, a grid needs to be generated and the aerodynamic coefficients must be calculated by solving the flowfield with INS2D. Eventhough, the traditional optimization method will have a shorter turn around time and CPU time used for one or two optimization runs, there is no guarantee that one or two optimization runs will find the best improvement. On the contrary, the neural networks will have an overall turn around time and CPU time for many optimization runs and there is no major increase in overall or CPU time for additional runs. Once the neural networks are trained, only 5-10 seconds are required for each additional optimization run. The CPU time that is used in this optimization study for the different training methods used is shown in Figure 6.33. Also plotted in this figure are the calculated CPU time that would of been used if the traditional optimization process is used. The CPU time for the traditional method is estimated by using the same number of gradient calls that is used in the neural network optimization procedure. Then for each iteration it is estimated that the CPU time will consist of 4.3 seconds to generate a grid and 600 seconds for each flow solution on a CRAY C90. It is estimated that 600 seconds will be required to converge the solution because more time is required to converge a solution near maximum lift. If more than three optimization runs are executed, then the neural network optimization procedure should be used. The neural network optimization procedure curves are nearly flat. Thus, the major contributor to the CPU time in the neural network optimization is training the neural networks to learn and to predict the aerodynamics of the airfoil. Many more optimization runs can be executed with this procedure without requiring large additional amount of CPU time. On the other hand, the traditional optimization procedure will continue to increase at a fairly linear rate as shown.

Another advantage of using the neural network optimization procedure is reduction of cost. There are many factors contributing to the total cost of a research job including the cost of the

engineering support or personnel, computer resources, and wall clock turn around time. One of the largest contributors to turn around time is waiting for a computer job to be completed especially if the job executes within a batch queue. The CFD cases in this current study are executed on a CRAY C90 or J90 computer and then the neural networks and optimizer are executed on a workstation. At the Numerical Aerospace Simulation Facility (NAS) at NASA Ames Research Center, there are three CRAY supercomputers that are available which are referred to as Eagle, von Neumann, and Newton. The average turn around time for an eight-hour batch job for a one month period for these three machines is shown in Figure 6.34. The average turn around time for these three computers, 23.45 hours, is used for all calculations in this study.

To calculate the cost that is related to the two types of optimization procedures considered, it is assumed that an experienced engineer is executing both optimization processes. This engineer is familiar with the different components to each process such as grid generation, flow simulation, neural networks, and optimization. The set-up time is assumed to be equal for both processes. The engineer is a full time equivalent of $200,000 per year and there are 2080 working hours in a year. Thus, there is a charge of $96.15 per hour for an engineer. Another expense which must be considered is computer resources. For this comparison, assume the cost of a

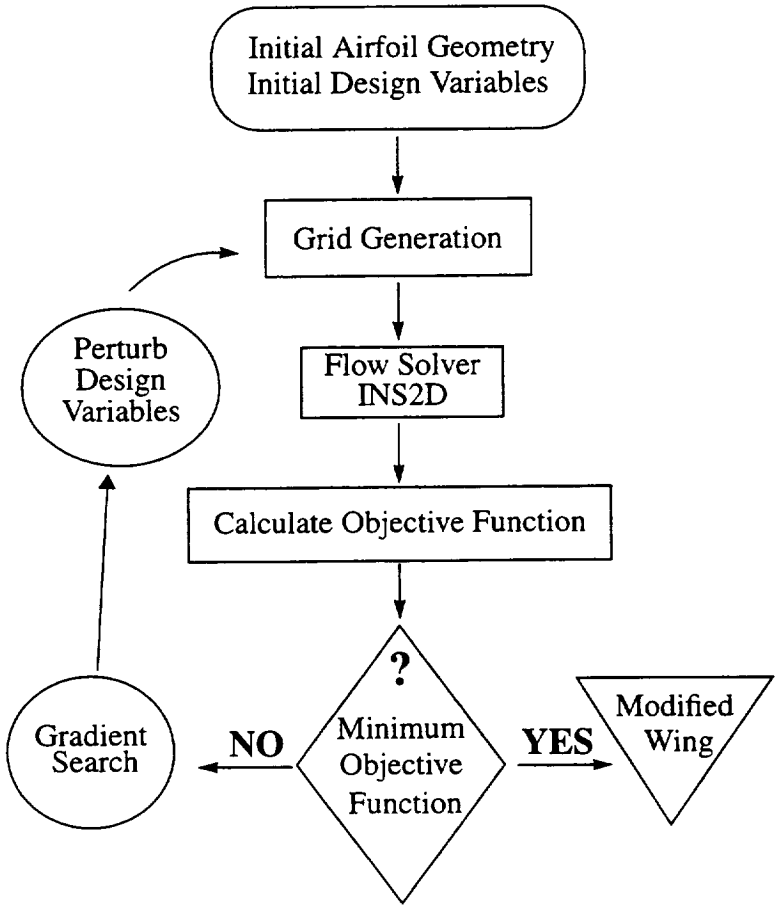


Figure 6.32 Traditional optimization process [75].

computing hour is $39.00 and an average turn around time for an eight-hour-queue-job is 23.45 hours.

First, the cost of the neural network optimization procedure is calculated. As a reminder, Methods 1, 5, 8, and 9 contain 27, 14, 19, and 20 configurations, respectively. A grid is generated for each configuration included in the training method and solutions are calculated for 10 different angles of attack for each configuration. The grid generation requires 4.3 CPU seconds per grid and 269 CPU seconds per flow solution (the convergence time is low since these solutions are below maximum lift). The CPU time required to train each method and used to optimize all five optimization runs (see Table 6.10) must also be added to the total wall clock time and the charged CPU time. The two additional constrained optimization runs for Method 9 are shown separate from the baseline optimization runs to illustrate the additional cost that is required. The total cost of the neural network optimization procedure is shown Table 6.10. The major element in the cost is the time and computer resources required to set-up the training matrix data. Consequently, it is very important to determine the level of prediction accuracy that is required and to choose the proper method to train the neural networks. By choosing Method 8 which has excellent prediction accuracy instead of Method 1, 30% of the total cost is reduced.

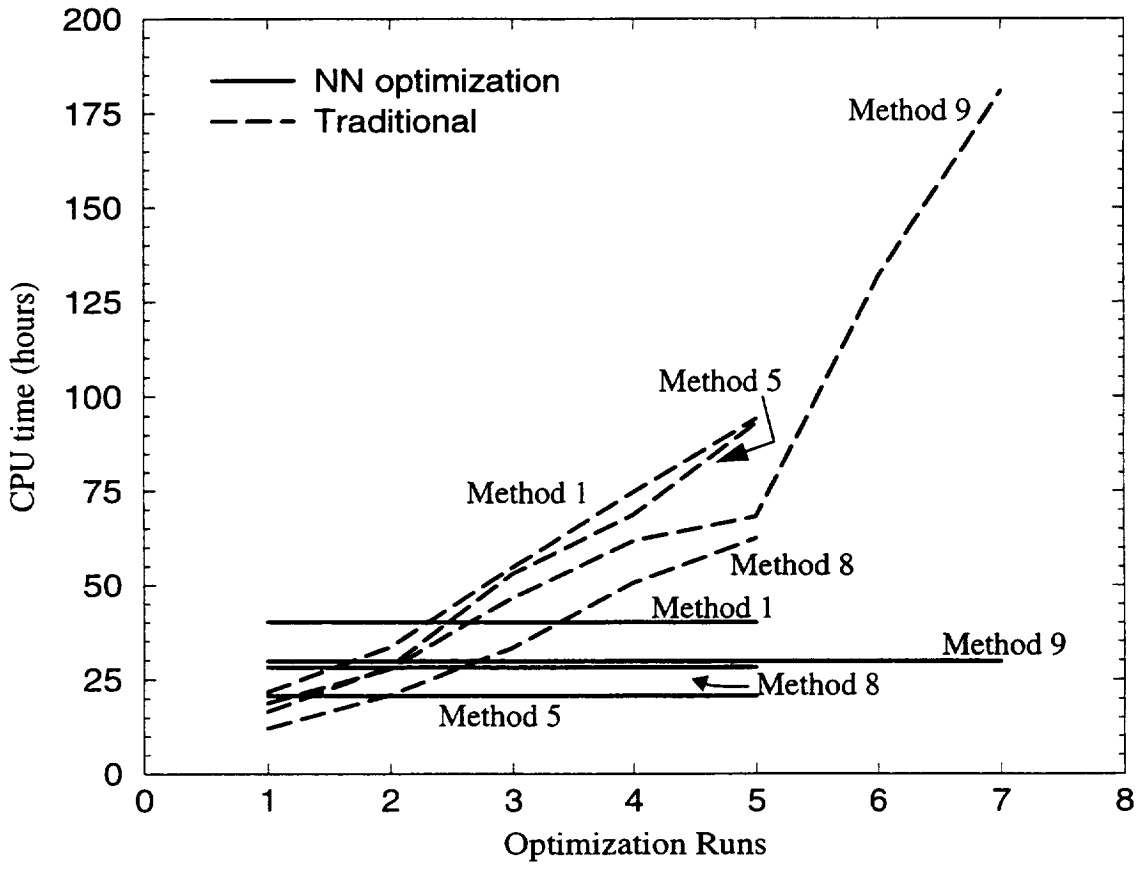


Figure 6.33 Comparison of CPU time required for traditional and neural network optimization procedures.

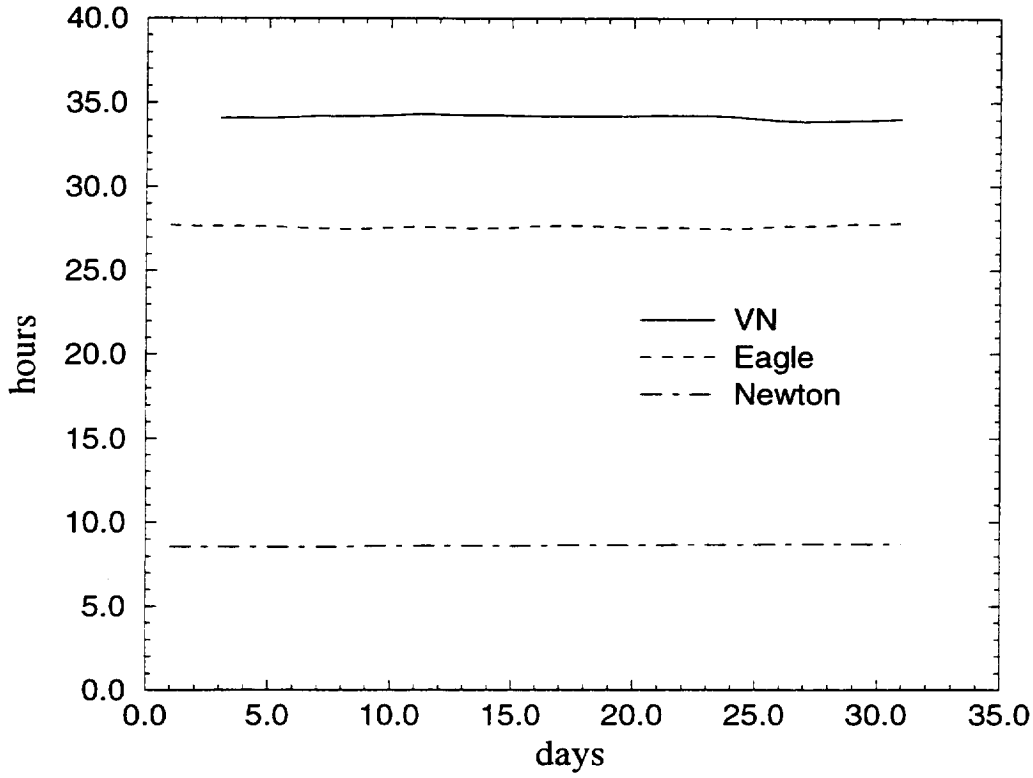


Figure 6.34 Average turn around time for an eight hour batch queue on CRAY computers for 31 days at NASA Ames Research Center.

Table 6.10 Neural Network Optimization Procedure Cost

Method	Configurations	Training CPU time (seconds)	Optimization CPU time (seconds)	Total Cost (dollars)
1	27	281	263	6466.50
5	14	207	93.5	3353.50
8	19	229	197.3	4551.66
9	20	232	176.6	4790.16
9-Constrain (2 runs)	-	-	409.29	10.93

Table 6.11 Traditional Optimization Procedure Cost

Method	CPU Hours	Number of 8 hour jobs	Wall Clock (hours)	Total Cost (dollars)
1	94.31	11.79	276.45	30,261.21
5	93.13	7.83	272.96	29,876.98
8	62.65	11.64	183.61	20,097.70
9	110.31	13.79	328.30	35,391.83
9-Constrain (2 runs)	70.57	8.82	206.82	22,638.84

Second, the cost of the traditional optimization procedure is calculated with the same assumptions. The wall clock time and the CPU hours charged are calculated based on the number of iterations (or gradient calls) that are made by the optimizer for each optimization run. For each method that is used in the neural network optimization procedure, the traditional optimization cost is calculated for the same five optimization starting runs. The total turn around (wall clock) time that the engineer waits for the job to be finished is multiplied by $96.15 and is added to the total CPU hours that are charged. The traditional optimization procedure is performed on the CRAY computer in the batch queue. This is one of the reasons that the cost is higher than the neural network optimization procedure as shown in Table 6.11. In the traditional optimization procedure, the cost to execute the two additional constrained optimization runs is very high. In contrast, the additional cost in the neural network optimization procedure is insignificant when compared to the cost to train the neural networks.

The total costs are compared in Figure 6.35 for the two optimization procedures. For five optimization runs for each training method and the two additional constrained optimization runs, the neural network optimization procedure does cost less. Again, if only one or two optimization runs are performed, then the traditional optimization procedure would cost less, however, for multiple runs, the neural network optimization procedure uses less resources. Also, constrained optimization is very costly because of the high number of gradient calls that are required to find the minimum objective function that satisfies all the constraints. Method 5 had the lowest cost for the neural network optimization procedure whereas Method 8 had the least cost for the traditional optimization procedure. The biggest advantage now is that many more optimization runs can be performed with the neural network optimization procedure while only adding seconds to the CPU time and turn around time as demonstrated by the addition of the constrained optimization. Thus, the cost would slightly increase but this is insignificant when compared to the traditional method. On the contrary, if additional optimization runs are performed with the traditional optimization procedure then both the CPU time and turn around time required would increase which would then drive the cost up in a linear fashion.

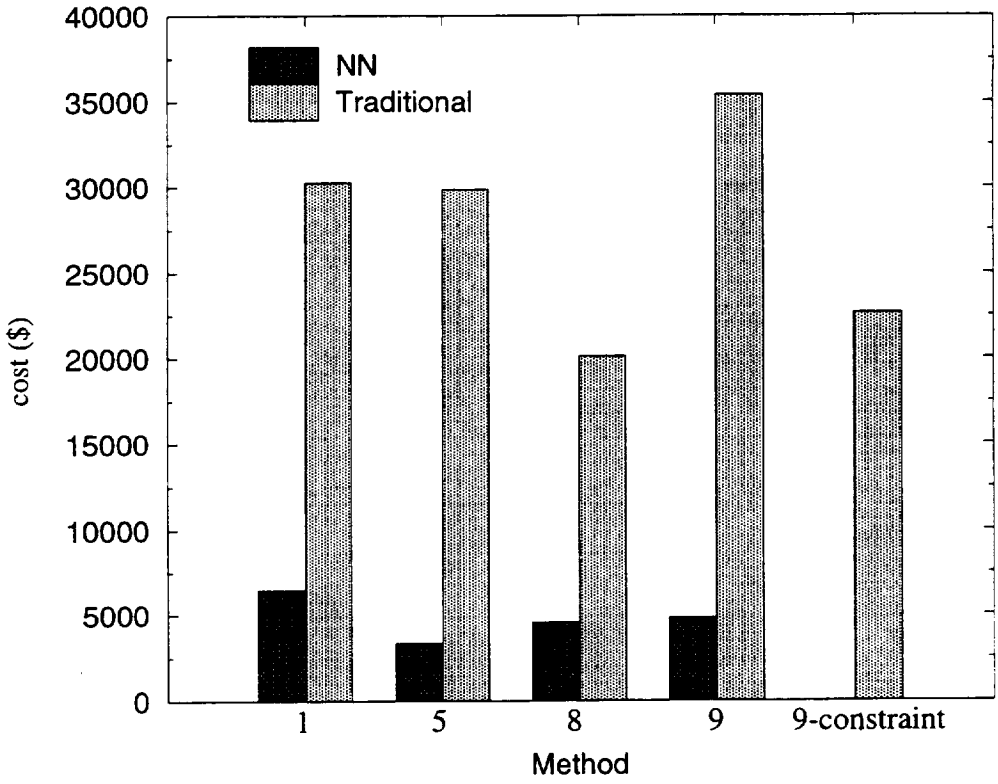


Figure 6.35 Comparison of total cost for the neural network and traditional optimization procedure.

Clearly, the neural network optimization procedure should be used for design because several designs with different constraints or design space can be considered without driving the cost and turn around time up. Also, once a design is chosen, the design space can be altered and the optimization procedure can now be performed again.

Chapter 7

Conclusions

A numerical investigation of the ability of artificial neural networks to predict the high-lift aerodynamics of a multi-element airfoil has been performed. An AI enhanced design process was developed which integrates neural network and optimizer technologies together with a computational database. This process is modular, allowing insertion of emerging neural network, optimization, and CFD techniques within its framework. This design process was tested for a typical high-lift design problem to optimize flap rigging for maximum lift. The ability of the neural networks to accurately predict the aerodynamic coefficients, lift, drag, and moment coefficients for any high-lift flap deflection, gap, and overlap, was demonstrated for both computational and experimental training data sets. Different methods of training the neural networks have been investigated to reduce the amount of data that is required to teach the neural networks to predict the aerodynamics precisely.

7.1 Summary

Multiple input, single output networks were trained using the NASA Ames variation of the Levenberg-Marquardt algorithm. The neural networks were first trained with wind tunnel experimental data of the three-element airfoil to test the validity of the neural networks. The networks did accurately predict the lift coefficients of the individual main and flap elements. However, there was noticeable error in predicting the slat lift coefficient. The prediction error in $C_{l_{slat}}$ is most likely caused by the sparse training data since there were only five different configurations used to train the neural networks. This resulted in high error in predicting the total lift coefficient since the total lift of the airfoil is the sum of the lift from the individual elements. This results from the fact that the errors are also summed and amplify in the prediction of the total lift coefficient.

Computational data is next used to train the neural networks to test if computational data can be used to train the neural networks. The neural networks were used to create a computational data base which may be used to impact design. Solutions were obtained by solving the two-dimensional Reynolds-averaged incompressible Navier-Stokes equations using the INS2D-UP code. The flowfield was assumed to be fully turbulent and the Spalart-Allmaras turbulence model was used. Two data sets were generated for two different slat deflections each consisting of configurations with different flap deflection, gap, and overlap. The data set con-

sisted of twenty-seven configurations. Subsets of this data set were generated to reduce the amount of data that is required to train the neural networks to accurately predict the aerodynamics.

The computational data set had to be pre-processed to reduce the prediction error at or beyond maximum lift. In high-lift aerodynamics, both experimentally and computationally, it is difficult to predict the maximum lift, and at which angle of attack it occurs. In order to predict maximum lift and the angle of attack where it occurs, a maximum lift criteria was needed. The pressure difference rule, which states that there exists a certain pressure difference between the peak suction pressure and the pressure at the trailing edge of the element at the maximum lift condition, was applied to all three elements. For this configuration, it was found that only the pressure difference on the slat element was needed to predict maximum lift. By applying the pressure difference rule, the prediction errors of the neural networks were reduced.

The amount of data that is required to train the neural networks was reduced to allow computational fluid dynamics to impact the design phase. Different subsets of the training methods were created by removing entire configurations from the six-degree-deflected slat training set. The mean and standard deviations of the root-mean-square prediction errors were calculated to compare the different methods of training. Even though the entire computational data set was sparse, it was reduced to only 70% of the entire data. It was found that the trained neural networks predicted the aerodynamic coefficients within an acceptable accuracy defined to be the experimental error. The aerodynamic data had to be represented in a nonlinear fashion so that the neural networks could learn and predict accurately. By carefully choosing the training subset, the computational data set was even further reduced to contain only 52% of the configurations. These trained neural networks also predicted the aerodynamic coefficients within the acceptable error. Thus, the computational data required to accurately represent the flowfield of a multi-element airfoil was reduced to allow computational fluid dynamics to be a usable tool for design.

This same procedure was followed in the twenty-six-degree-deflected slat computational data. This data set had higher deflected flaps which were actually out of the normal flight envelope. The same trends were found except that the prediction error was much higher in this training set than the previous one. This was caused by the fact that the flowfield was severely separated with the higher deflected flaps. Thus, the training data representing the flowfield was noisy which leads to prediction errors.

The computational design space needs to be easily searched for areas of interest such as maximums or optimal points. An optimization study to search the design space was conducted by using neural networks that were trained with computational data. Artificial neural networks have been successfully integrated with a gradient based optimizer to minimize the amount of data required to completely define the design space of a three-element airfoil. The accuracy of the neural networks' prediction was tested for both the initial and modified configurations by generating the grid and computing the INS2D-UP solution.

The high-lift flap aerodynamics were optimized for a three-element airfoil by maximizing the lift coefficient. The design variables were flap deflection, gap, and overlap. The bounds of the design space had to be the same as the bounds that were used to train the neural networks

since the neural networks are good interpolators and bad extrapolators. Multiple optimization runs were conducted in order to find the best improvement.

The different training subsets were used in the optimization with neural networks process. The prediction errors were below the acceptable value when only 70% of the computational data set was used to train the neural networks. The highest maximum lift was found with the following high-lift flap setting for $\delta_s = 6.0°$: $\delta_f = 38.5°$, $gap_f = 1.69\%c$, $ol_f = 0.97\%c$, and $\alpha = 10.0°$ which produced a $C_{l_{INS2D}} = 4.11$. The optimal flap setting for the twenty-six degree slat is $\delta_f = 38.5°$, $gap_f = 2.1\%c$, $ol_f = 0.4\%c$, and $\alpha = 18.0°$ with $C_{l_{INS2D}} = 4.73$. The pressure distribution of the original and modified configurations were compared. The modified configurations had larger suction pressures which contributed to the additional lift that was generated. Several, modified flaps had attached flow whereas the original flaps had separated flow. This resulted in the modified airfoil producing more lift.

Initial studies showed that although optimization could be conducted using a sparse training dataset, unconstrained optimization of the high-lift system produced unacceptably high errors. Due to the complexity of the high-lift flow physics near the maximum lift condition, an empirically based constraint, which identifies configurations at the maximum lift condition within the computational database, was required in order to achieve accurate neural net predictions for this design problem. Using the empirical constraint together with an iterative optimization procedure which re-inserted the optimized configuration into the training database and repeated the optimization produced an optimal configuration with only 0.5% error.

A cost analysis was conducted by comparing the optimization with neural networks procedure to the traditional optimization procedure. It was found that the optimization with neural networks procedure resulted in a reduction of turnaround time, CPU time, and cost if more than two optimization runs were conducted. After the neural networks were trained and integrated with the optimizer, many optimization runs were executed with only using an average of 6.5 CPU seconds per run and 30 turnaround seconds per run.

Overall, the neural networks were trained successfully to predict the high-lift aerodynamics of a multi-element airfoil. The neural networks were also able to predict the aerodynamics successfully when only 52-70% of the entire computational data set was used to train. The neural networks were integrated with an optimizer thus allowing a quick way to search the design space for points of interest. Optimization with neural networks reduced the turnaround time, CPU time, and cost of multiple optimization runs. Therefore, neural networks are an excellent tool to allow computational fluid dynamics to impact the design space.

7.2 Recommendations

Based on the results of this investigation, several recommendations can be made. Different types of neural networks should be studied to see if the prediction error can be further reduced. Adaptive neural networks which learn as they predict should be examined. These networks may lead to faster training time, smaller training data sets, and may generalize better for new test samples. Also, they may produce better fidelity with the same amount of training data used as in the current neural networks. If the training time is reduced and the training set can be further

reduced than it was in this study, then this will further reduce the cost of the optimization procedure.

Second, when optimizing a multi-element airfoil or wing for maximum lift, the constrained optimization process that is described in Section 6.4.6 should be performed. The pressure difference rule should be applied to define the maximum lift and the angle of attack where it occurs. The design space is then constrained to include only the angles of attack at maximum lift and lower. By constraining the design space that the optimizer is allowed to search and by adding one data point near maximum lift to the training data, the prediction error is reduced and all constraints are met.

Next, a hybrid training data set consisting of experimental and computational data needs to be pursued. This would require correction factors and correlating factors to be generated to correctly join the data sets together. The design space would be enlarged and would be better defined. Thus, the neural networks can be trained with the more detailed training data set. Further, computational fluid dynamics for predicting high-lift flowfields needs to be improved, including turbulence modeling. This will reduce some of the requirements of correction and correlating factors.

Another recommendation would be to integrate an artificial intelligence tool such as genetic algorithms to help direct the optimizer to find the global maximum in fewer optimization runs. Lastly, the optimization with neural networks procedure should be executed for a three-dimensional body. The optimization with neural networks procedure should even further reduce the cost in a three-dimensional optimization problem than the traditional procedure since three-dimensional computations require more resources. The ability of the neural networks that were trained with computational data to predict three-dimensional aerodynamic data needs to be investigated.

Bibliography

[1] Ying, S. X.: High Lift Challenges and Directions for CFD. AIAA/NPU AFM Conference Proceedings, June 1996.

[2] LaMarsh, W. J.; Walsh, J. L.; and Rogers, J. L.: Aerodynamic Performance Optimization of a Rotor Blade Using a Neural Network as the Analysis. AIAA Paper 92-4837, Sept. 1992.

[3] Faller, W. E.; and Schreck, S. J.: Real-Time Prediction of Unsteady Aerodynamics: Application for Aircraft Control and Maneuverability Enhancement. IEEE Transactions on Neural Networks, vol. 6, no. 6, Nov. 1995, pp. 1461 - 1468.

[4] McMillen, R. L.; Steck, J. E.; and Rokhsaz, K.: Application of an Artificial Neural Network as a Flight Test Data Estimator. J. Aircraft, vol. 32, no. 5, Sept. - Oct. 1995, pp. 1088 - 1094.

[5] Steck, J. E.; and Rokhsaz, K.: Some Applications of Artificial Neural Networks in Modeling of Nonlinear Aerodynamics and Flight Dynamics. AIAA Paper 97-0338, Jan. 1997.

[6] Rai, M. M.; and Madavan, N. K.: Application of Artificial Neural Networks to the Design of Turbomachinery Airfoils. AIAA Paper 98-1003, Jan. 1998.

[7] Jorgensen, C. C.; and Ross, J. C.: System and Method for Modeling the Flow Performance Features of an Object. U. S. Patent No. 5,649,064, July 1997.

[8] Ross, J. C.; Jorgenson, C. C.; and Norgaard, M.: Reducing Wind Tunnel Data Requirements Using Neural Networks. NASA TM 112193, May 1997.

[9] Storms, B.: Private communication, July 1997.

[10] Anderson, D. A.; Tannehill, J. C.; Pletcher, R. H.: Computational Fluid Dynamics and Heat Transfer. Hemisphere Publishing Co., 1984.

[11] Rumsey, C. L.; Gatski, T. B.; Ying, S. X.; and Bertelrud, A.: Prediction of High-Lift Flows Using Turbulent Closure Models. AIAA Paper 97-2260, June 1997.

[12] Spalart, P. R.; and Allmaras, S. R.: A One-Equation Turbulence Model for Aerodynamic Flows. AIAA Paper 92-0439, Jan. 1992.

[13] Rogers, S.: Progress in High-Lift Aerodynamic Calculations. AIAA Paper 93-0194, Jan. 1993.

[14] Rogers, S. E.; Menter, F. R.; Durbin, P. A.; and Mansour, N. N.: A Comparison of Turbulence Models in Computing Multi-Element Airfoil Flows. AIAA Paper 94-0291, Jan. 1994.

[15] Ashby, D. L.: Experimental and Computational Investigation of Lift-Enhancing Tabs on a Multi-Element Airfoil. NASA TM 110432, Dec. 1996.

[16] Jones, K. M.; Biedron, R. T.; and Whitlock, M.: Application of a Navier-Stokes Solver to the Analysis of Multielement Airfoils and Wings Using Multizonal Grid Techniques. AIAA Paper 95-1855, June 1995.

[17] Ko, S.; and McCroskey, W. J.: Computations of Unsteady Separating Flows Over an Oscillating Airfoil. AIAA Paper 95-0312, Jan. 1995.

[18] Rogers, S.; and Kwak, D.: An Upwind-Differencing Scheme for the Incompressible Navier-Stokes Equations. NASA TM 101051, Nov. 1988.

[19] Rogers, S.; and Kwak, D.: Upwind Differencing Scheme for the Time-Accurate Incompressible Navier-Stokes Equations. AIAA J., vol. 28, no. 2, Feb. 1990, pp. 253-262.

[20] Chorin, A.: A Numerical Method for Solving Incompressible Viscous Flow Problems. J.

of Computational Physics, vol. 2, 1967, pp. 12-16.

[21] Rogers, S. E.: Numerical Solution of the Incompressible Navier-Stokes Equations. NASA TM 102199, Nov. 1990.

[22] Agosta, R. M.: Numerical Analysis of Tangential Slot Blowing on a Generic Chined Forebody. NASA TM 108845, Sept. 1994.

[23] Roe, P. L.: Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. J. of Computational Physics, vol. 43, 1981 pp. 357-372.

[24] Rogers, S. E.: A Comparison of Implicit Schemes for the Incompressible Navier-Stokes Equations With Artificial Compressibility. AIAA Paper 95-0567, Jan. 1995.

[25] Rogers, S.: Manual for the OVERMAGG Script System. NASA Ames Research Center, July 1997.

[26] Chan, W. M.; Chui, I. T.; and Buning, P. G.: User's Manual for the HYPGEN Hyperbolic Grid Generator and the HGUI Graphical User Interface. NASA TM 108791, Oct. 1993.

[27] Suhs, N. E.; and Tramel R. W.: PEGSUS 4.0 User's Manual. AEDC-TR-91-8, Nov. 1991.

[28] Public Affairs Office, NASA Ames Research Center: Exploring Aeronautics: A Curriculum in Aeronautics for the 4th through 6th Grades. NASA Ames Research Center, 1997.

[29] Kung, S. Y.: Digital Neural Networks. PTR Prentice-Hall, Inc., 1993.

[30] Mehrotra, K.; Mohan, C. K.; and Ranka, S.: Elements of Artificial Neural Networks. MIT Press, 1997.

[31] Neural Computing: A Technology Handbook for Professional II/Plus and NeuralWorks Explorer. NeuralWare Inc., 1996.

[32] Cowan, J. D.; and Sharp, D. H.: Neural Nets. Technical Report, Mathematics Department, University of Chicago, 1987.

[33] Grant, I.; and Pan, X.: An Investigation of the Performance of Multi Layer, Neural Networks Applied to Analysis of PIV Images. Experiments in Fluids, vol. 19, 1995, pp. 159 - 166.

[34] Zeidenberg, M.: Neural Networks in Artificial Intelligence. Ellis Horwood Limited, 1990.

[35] Widrow, R.; and Lehr, M.: 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. Proceedings of the IEEE, vol. 78, 1990, pp. 1415 - 1442.

[36] Rumelhart, D. E.; McClelland, J. L.; and the PDP Group: Parallel Distributed Processing, Vol. I and II. MIT Press, 1986.

[37] Karayiannis, N. B.; and Venetsanopoulos, A. N.: Artificial Neural Networks Learning Algorithms, Performance Evaluation, and Applications. Kluwer Academic Publishers, 1993.

[38] Rashevsky, N.: Mathematical Biophysics. University of Chicago Press, 1938.

[39] McCulloch, W. S.; and Pitts, W.: A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, vol. 5, 1943, pp. 115 - 133.

[40] Hebb, D.: The Organization of Behavior. John Wiley, 1949.

[41] Gabor, D.: Communication Theory and Cybernetics. IRE Transactions on Circuit Theory, CT-1, 1954, pp. 19 - 31.

[42] Rosenblatt, F.: The Perceptron, A Probabilistic Model for Information Storage and Organization in the Brain. Psychology Review, vol. 62, 1958, pp. 386 - 408.

[43] Widrow, B.; and Hoff, M.: Adaptive Switching Circuits. Western Electronic Show Convention, Convention Record, Institute of Radio Engineers (now *IEEE*), vol. 4, 1960, pp. 96 - 104.

[44] Rosenblatt, F.: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mech-

anisms. Spartan, 1961.

[45] Minsky, M. L.; and Papert, S. A.: Perceptrons. MIT Press, 1969.

[46] Dreyfus, S.: The Numerical Solution of Variational Problems. J. of Mathematical Analysis and Applications, vol. 5, no. 1, 1962, pp. 30 - 45.

[47] Bryson, A. E.; and Ho, Y. C.: Applied Optimal Control. Blaisdell, New York, 1969.

[48] Anderson, J. A.; A Simple Neural Network Generating an Associative Memory. Mathematical Bioscience, vol. 14, 1972, pp. 197 - 220.

[49] Anderson, J. A.; Silverstein, J. W.; Ritz, S. A.; and Jones, R. S.: Distinctive Features, Categorical Perceptron, and Probability Learning: Some Applications of a Neural Model. Psychological Review, vol. 84, 1977, pp. 413 - 451.

[50] Werbos, P.: Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph. D. Dissertation, Harvard University, Cambridge, May 1974.

[51] Grossberg, S.: Classical and Instrumental Learning by Neural Networks. Progress in Theoretical Biology, vol. 3, 1977, pp. 51 - 141.

[52] McClelland J.; and Rumelhart, D.: Explorations in Parallel Distributed Processing. MIT Press, Cambridge, 1988.

[53] Kohonen, T.: Self-Organization and Associative Memory. Springer-Verlag, New York, 1988.

[54] Ackley, D. H.; Hinton, G. E.; and Sejnowski, T. J.: A Learning Algorithm for Boltzmann Machines. Cognitive Sciences, vol. 9, 1985, pp. 147 - 169.

[55] Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P.: Optimization by Simulated Annealing. Science, vol. 220, 1983, pp. 671 - 680.

[56] Gallant, S. I.: Optimal Linear Discriminants. Proceedings of the Eighth International

Conference on Pattern Recognition. Paris, 1986, pp. 849 - 852.

[57] Mah, R.: Intelligent Controller for Neurosurgery. Research and Technology 1996, NASA TM 112195, Sept. 1997, pp.126-127.

[58] Jorgensen, C. C.: Direct Adaptive Aircraft Control Using Dynamic Cell Structure Neural Networks. NASA TM 112198, May 1997.

[59] Levenberg, K.: A Method for the Solution of Certain Non-linear Problems in Least Squares. Quarterly Applied Mathematics, vol. 2, 1944, pp. 164 - 168.

[60] Marquardt, D. W.: An Algorithm for Least Squares Estimation of Nonlinear Parameters. Journal of Society of Industrial Applied Mathematics, vol. 11, no. 2, June 1963, pp. 431 - 441.

[61] Walsh, G. R.: Methods of Optimization. John Wiley & Sons, London, 1975.

[62] Kollias, S.; and Anastassiou, D.: An Adaptive Least Squares Algorithm for the Efficient Training of Artificial Neural Networks IEEE Transactions on Circuits and Systems, vol. 36, no. 8, Aug. 1989, pp. 1092 - 1101.

[63] Norgaard, M.; Jorgensen, C. C.; and Ross, J. C.: Neural-Network Prediction of New Aircraft Design Coefficients. NASA TM 112197, May 1997.

[64] Fletcher, R.: Practical Methods of Optimization. John Wiley & Sons, New York, 1987.

[65] Dulikravich, G.: Aerodynamic Shape Design and Optimization: Status and Trends. J. Aircraft, vol. 29, no. 6, Nov. 1992, pp. 1020-1026.

[66] Balling, R. J.; and Sobieszczanski-Sobieski, J.: Optimization of Coupled Systems: A Critical Overview of Approaches. NASA CR 195019, Dec. 1994.

[67] van den Dam, R. F.; van Egmond, J. A.; and Slooff, J. W.: Optimization of Target Pressure Distributions. Special Course on Inverse Methods for Airfoil Design for Aeronautical and Turbomachinery Applications. AGARD Report No. 780, Reference 3, Nov. 1990.

[68] Cheung, S.; Aaronson, P.; and Edwards, T.: CFD Optimization of a Theoretical Minimum-Drag Body. J. Aircraft, vol. 32, no. 1, January 1995, pp. 193-198.

[69] Cosentino, G. B.; and Holst, T. L.: Numerical Optimization of Advanced Transonic Wing Configurations. J. Aircraft, vol. 23, no. 3, May 1986, pp. 192-199.

[70] Ta'asan, S.; Kuruvila, G.; and Salas, M. D.: Aerodynamic Design and Optimization in One Shot. AIAA Paper 92-0025, Jan. 1992.

[71] Reuther, J.; and Jameson, A.: Control Theory Based Airfoil Design for Potential Flow and a Finite Volume Discretization. AIAA Paper 94-0499, Jan. 1994.

[72] Reuther, J.; and Jameson, A.: Aerodynamic Shape Optimization of Wing and Wing-Body Configurations Using Control Theory. AIAA Paper 95-0123, Jan. 1995.

[73] Obayashi, S.; and Takanashi, S.: Genetic Optimization of Target Pressure Distributions for Inverse Design Methods. AIAA Paper 95-1649, June 1995.

[74] Gill, P. E.; Murray, W.; Saunders, M. A.; and Wright, M. H.: User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming. Dept. of Operations Research, Stanford University, TR SOL 94, Stanford, CA, 1994.

[75] Greenman, R. M.; Cheung, S.; and Tu, E. L.: Coupled Navier-Stokes and Optimizer Analysis of a Transonic Wing. J. Aircraft, vol. 35, no. 3, May 1998, pp. 362-369.

[76] Gill, P. E.; Murray, W.; Saunders, M. A.; and Wright, M. H.: User's Guide for SOL/QPSOL Version 3.2. Report SOL 84-5, Department of Operations Research, Stanford, CA, 1984.

[77] Dennis, J. E., Jr.; and More, J. J.: Quasi-Newton Methods, Motivation and Theory. SIAM Review, vol. 19, pp. 46-89.

[78] Valarezo, W. O.; and Chin, V. D.: Method of Prediction of Wing Maximum Lift. J. Aircraft, vol. 31, no. 1, Feb. 1994, pp. 103-109.

[79] Fiddes, S. P.; Kirby, D. A.; Woodward, D. S.; and Peckham, D., H.: Investigation into the Effects of Scale and Compressibility on Lift and Drag in the RAE 5m Pressurized Low-Speed Wind Tunnel. Aeronautical J., vol. 89, Paper 1302, Mar. 1985, pp.93-108.

[80] Valarezo, W. O.; Dominik, C. J.; McGhee, R. J.; Goodman, W. L.; and Paschal, K. B.: Multi-Element Airfoil Optimization for Maximum Lift at High Reynolds Numbers. AIAA Paper 91-3332, Sept. 1991.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1998	Technical Memorandum

4. TITLE AND SUBTITLE

Two-Dimensional High-Lift Aerodynamic Optimization Using Neural Networks

5. FUNDING NUMBERS

5191042

6. AUTHOR(S)

Roxana M. Greenman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Ames Research Center
Moffett Field, CA 94035-1000

8. PERFORMING ORGANIZATION REPORT NUMBER

A-9811759

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA/TM—1998-112233

11. SUPPLEMENTARY NOTES

Point of Contact: Roxana M. Greenman, Ames Research Center, MS 258-1, Moffett Field, CA 94035-1000; (650) 604-3997

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified — Unlimited
Subject Category 02

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

The high-lift performance of a multi-element airfoil was optimized by using neural-net predictions that were trained using a computational data set. The numerical data was generated using a two-dimensional, incompressible, Navier-Stokes algorithm with the Spalart-Allmaras turbulence model. Because it is difficult to predict maximum lift for high-lift systems, an empirically-based maximum lift criteria was used in this study to determine both the maximum lift and the angle at which it occurs. The "pressure difference rule," which states that the maximum lift condition corresponds to a certain pressure difference between the peak suction pressure and the pressure at the trailing edge of the element, was applied and verified with experimental observations for this configuration. Multiple input, single output networks were trained using the NASA Ames variation of the Levenberg-Marquardt algorithm for each of the aerodynamic coefficients (lift, drag and moment). The artificial neural networks were integrated with a gradient-based optimizer. Using independent numerical simulations and experimental data for this high-lift configuration, it was shown that this design process successfully optimized flap deflection, gap, overlap, and angle of attack to maximize lift. Once the neural nets were trained and integrated with the optimizer, minimal additional computer resources were required to perform optimization runs with different initial conditions and parameters. Applying the neural networks within the high-lift rigging optimization process reduced the amount of computational time and resources by 44% compared with traditional gradient-based optimization procedures for multiple optimization runs.

14. SUBJECT TERMS

Neural networks, High-Lift, Optimization

15. NUMBER OF PAGES

146

16. PRICE CODE

A07

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		