CONTROL OF THE NASA LANGLEY 16-FOOT TRANSONIC TUNNEL WITH THE SELF-ORGANIZING FEATURE MAP

By

MARK A. MOTTER

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1998

To the memory of my parents

ACKNOWLEDGMENTS

I would like to thank Dr. Jose C. Principe, my advisor and supervisory committee chairman, for his encouragement, guidance, patience, and insight during the course of this research and the writing of this dissertation. I am also grateful for the time and patience of my committee members, Dr. Gijs Bosman, Dr. Thomas E. Bullock, Dr. John G. Harris, and Dr. Loc Vu-Quoc.

I would also like to acknowledge the support of this research by NASA Langley Research Center, and encouragement from both my former branch head, Kenneth L. Jacobs, and the current branch head, Carl E. Horne.

Finally, I would like to thank my wife, Lisa, for her unwavering support, and Benjamin, who makes every day new.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

CONTROL OF THE NASA LANGLEY 16-FOOT TRANSONIC TUNNEL
WITH THE SELF-ORGANIZING FEATURE MAP

By

MARK A. MOTTER

May 1998

Chairman: Dr. Jose C. Principe
Major Department: Electrical and Computer Engineering

A predictive, multiple model control strategy is developed based on an ensemble of local linear models of the nonlinear system dynamics for a transonic wind tunnel. The local linear models are estimated directly from the weights of a self organizing feature map (SOFM). Local linear modeling of nonlinear autonomous systems with the SOFM is extended to a control framework where the modeled system is nonautonomous, driven by an exogenous input. This extension to a control framework is based on the consideration of a finite number of subregions in the control space.

Multiple self organizing feature maps collectively model the global response of the wind tunnel to a finite set of representative prototype controls. These prototype controls partition the control space and incorporate experiential knowledge gained from

decades of operation. Each SOFM models the combination of the tunnel with one of the representative controls, over the entire range of operation. The SOFM based linear models are used to predict the tunnel response to a larger family of control sequences which are clustered on the representative prototypes. The control sequence which corresponds to the prediction that best satisfies the requirements on the system output is applied as the external driving signal.

Each SOFM provides a codebook representation of the tunnel dynamics corresponding to a prototype control. Different dynamic regimes are organized into topological neighborhoods where the adjacent entries in the codebook represent the minimization of a similarity metric which is the essence of the self organizing feature of the map. Thus, the SOFM is additionally employed to identify the local dynamical regime, and consequently implements a switching scheme than selects the best available model for the applied control.

Experimental results of controlling the wind tunnel, with the proposed method, during operational runs where strict research requirements on the control of the Mach number were met, are presented. Comparison to similar runs under the same conditions with the tunnel controlled by either the existing controller or an expert operator indicate the superiority of the method.

CHAPTER 1
INTRODUCTION

Motivation

The initial motivation for this research was to extend neural network based methods that had proven successful in modeling autonomous nonlinear dynamical systems [Principe and Kuo, 1994; Principe, Hsu, and Kuo, 1994; Principe, Kuo, and Celebi, 1994] to the modeling of nonautonomous dynamical systems. The temporal state evolution of an autonomous system is functionally dependent only on the system state, but nonautonomous systems allow for an explicit dependence on an independent variable, usually taken to be time [Jackson, 1989] or some function of time, in addition to the system state. For this study, this independent variable is taken to be an external, or *exogenous* driving signal, referred to as the control input. For an autonomous system, it is reasonable to assume that the future behavior, or output, of the system can be predicted over some finite interval from a finite number of observations of past outputs [Takens, 1980]. In contrast, predictions of the behavior of a nonautonomous system require consideration of not only the past outputs in response to past inputs, but the future input to the system as well.

It was also desired to develop a global representation of the underlying nonautonomous dynamic system, that is, a model, or a collection of models that fit all of the state space. This is in contrast to a local representation which is valid only in a

restricted region of the state space. The desired global representation may be achieved by a single model if the underlying system is simple, but most complex, nonlinear dynamical systems can only be represented in a localized region of the state space by a single model. This naturally leads to the use of multiple local models to represent the global characteristics of a system with some method employed to smoothly patch together the local models [Principe and Wang, 1995] in a system identification context, or to switch between models [Narendra, Balakrishnan, and Ciliz, 1995] in a control context.

Another prime motivation in the research was to develop models that would be amenable for control of the underlying system, as opposed to models developed solely for system identification. It was desired to have a system model that would provide a computationally cost-effective means of determining the input signal to be applied to the system in order to achieve a desired state. In this context, an approximate model that is linear in the control input is more desirous than an exact model which has a nonlinear dependence on the control [Narendra and Mukhopadhyay, 1997].

Finally, the combined modeling and control scheme was to be implemented in software and experimental tests conducted using the actual dynamical system under study.

Evolution of the Research

The dynamical system considered for this study is the 16-Foot Transonic Tunnel at the NASA Langley Research Center in Hampton, Virginia. The NASA Langley 16-Foot Transonic Tunnel, simply referred to as the *tunnel* in the sequel, is driven by a simple control input which provides the function of setting the desired output, which is

the Mach number, while compensating for any external disturbances. The task of modeling and controlling the Mach number with an artificial neural network system was undertaken with the vision to capture the underlying dynamics of an nonautonomous system from observations of time-dependent, input-output data. After suitably extracting the underlying dynamical model from the tunnel input-output data, predictions of the response to future control inputs are based on this model. A control input sequence which minimizes the error between the desired response and the predicted response, over a reasonable time horizon, is then selected from a set of candidate input sequences. This input sequence is finally applied as the control input to the wind tunnel.

The first major task was to find a suitable neural architecture for modeling the wind tunnel dynamics based solely on input-output data. Our initial studies investigated the use of several dynamic neural networks to identify the dynamics of the wind tunnel response to control inputs, at one particular operating point [Principe and Motter, 1994]. The most promising architecture from this study was investigated further, using a single global dynamic neural network for system identification over a wide range of operating points [Motter and Principe, 1994]. This model was reasonably successful in predicting the steady-state wind tunnel response at various operating points when driven by similar control inputs. A refinement of this model came when the wind tunnel responses were first clustered using a competitive neural network [Motter and Principe, 1995]. A competitive neural network was used to cluster the tunnel responses at several operating points to similar control inputs, thereby extracting pertinent features of the response. The clustering of the wind tunnel dynamic responses provided a basis for developing a set of predictors that collectively captured the dynamics of the wind tunnel response for a single

class of similar control inputs. At this point, it became clear that a significant improvement in the prediction accuracy could be realized from an ensemble of local models, each derived from a clustering of the tunnel dynamic responses.

The control input space was partitioned manually, based on experience and the bang-zero-bang (+1, 0, and -1) permissible values of the control signal. If the control input sequence is considered to be a p-component vector with each element having a value of +1, -1, or zero, then there are 3^p possible control sequences to be considered. The idea was to partition the control input space by manually constructing representative prototype vectors for the control sequence. The goal of this partitioning was to provide a set of control inputs capable of driving the tunnel from one operating point to another, regulating about a given operating point, rejecting disturbances, while eliminating control sequences known to be experimentally of no practical interest. Limiting the number of candidate controls to be evaluated by the predictive controller was a major consideration in partitioning the control input space. Initially this partitioning was done with five control input prototypes, but later, in the implementation of the experiment, the partitioning was extended to nine control input prototype vectors, to provide the desired control accuracy.

For each these control input classes, the tunnel Mach number responses were clustered using Kohonen's self-organizing feature map (SOFM) [Kohonen; 1990, 1995]. The SOFM is a competitive neural architecture that imposes a topographic ordering of the output neural field corresponding to features of the input patterns, which are in this case, the Mach number responses. For prediction purposes, the SOFM's advantage is that the topographic ordering imposes a similarity measure over the input neural field. This

similarity can be exploited in the construction of local linear models from the input neural field corresponding to the winning output. The construction of local linear models facilitated the evaluation of the wind tunnel response to a larger set of candidate controls than could have been realized with multiple dynamic models.

Background

The 16-Foot Transonic Tunnel at the NASA Langley Research Center, Hampton, Virginia, is a closed circuit, single-return, continuous-flow, atmospheric tunnel with a Mach number capability from 0.20 to 1.30. When the tunnel began operation in November 1941, it had a circular test section that was 16 feet in diameter and maximum Mach number of 0.71 [Peddrew, 1981]. Numerous upgrades to both the test section and drive system have expanded the test envelope of this facility. Currently, Mach numbers up to 1.05 are achieved using the tunnel main drive fans only. Mach numbers from 1.05 to 1.3 require the combination of test section plenum suction with the tunnel fans. The tunnel fans, 34 feet in diameter, are driven from 60 to 372 rpm by a 50 MW electric drive system. An air removal system using a 30 MW compressor and 10-Foot diameter butterfly valve provides test section plenum suction. At Mach numbers above 1.275, the 10-Foot valve is fully open and increases in Mach number are obtained from increased power to the tunnel main drive fans. Figure 1 is an aerial view of the tunnel. Figure 2 is a view of the tunnel test section with a model inserted. Figure 3 shows the arrangement of the major components of the tunnel. Figure 4 shows a view from the inside of the tunnel near the second set of turning vanes.

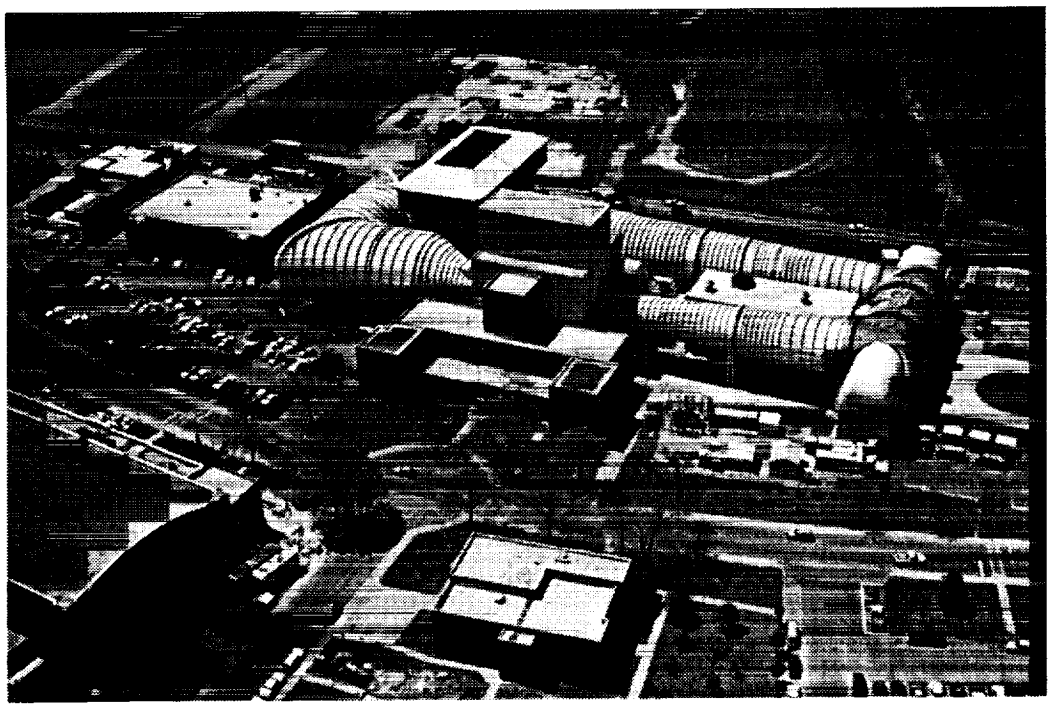


Figure 1. Aerial View of the 16-Foot Transonic Tunnel

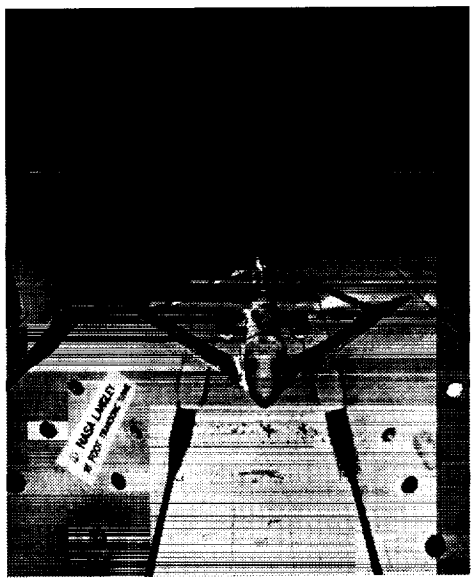


Figure 2. Test Section with model in place

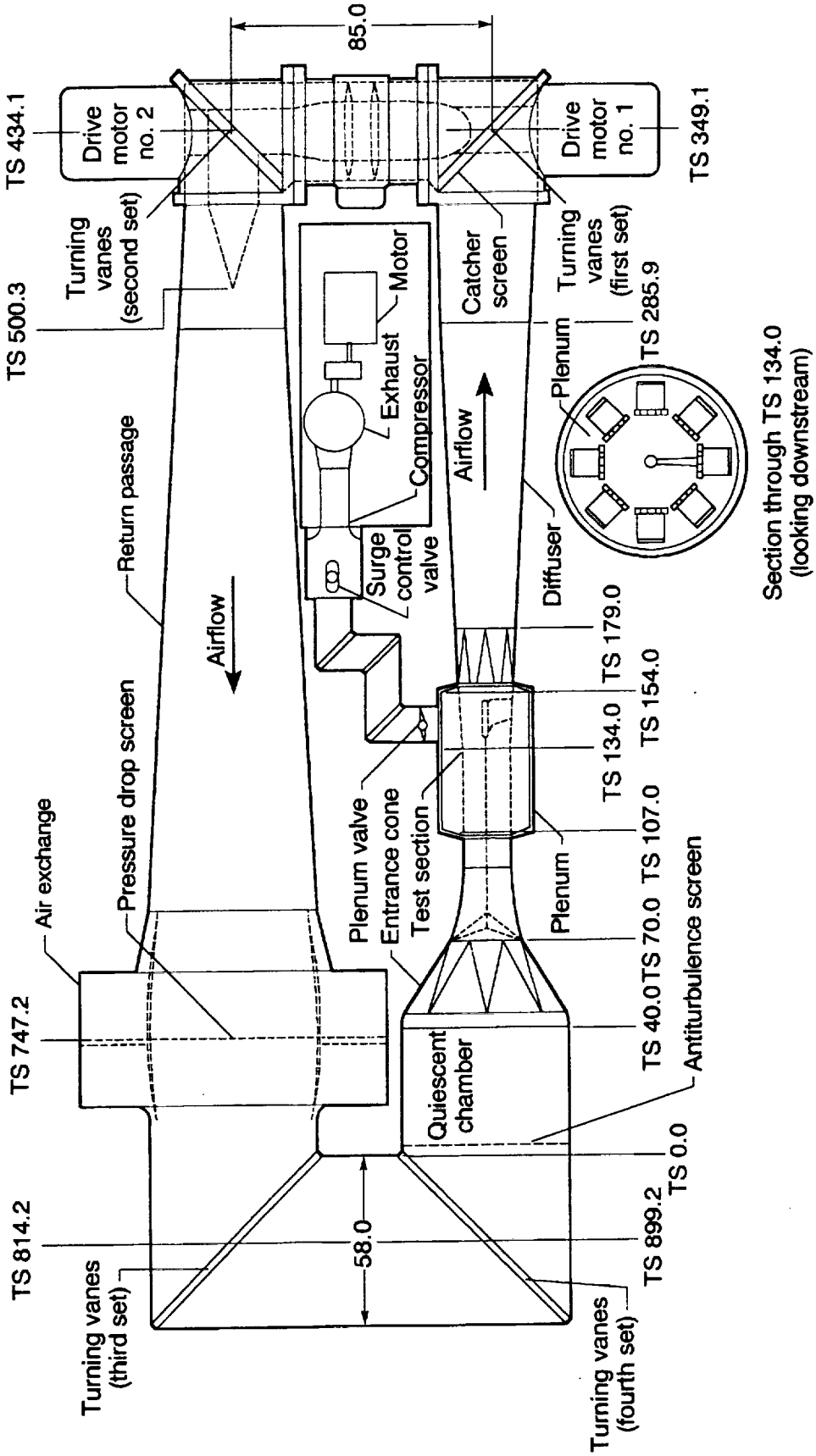


Figure 3. Arrangement of Langley 16-Foot Transonic Tunnel. Dimensions are in feet.

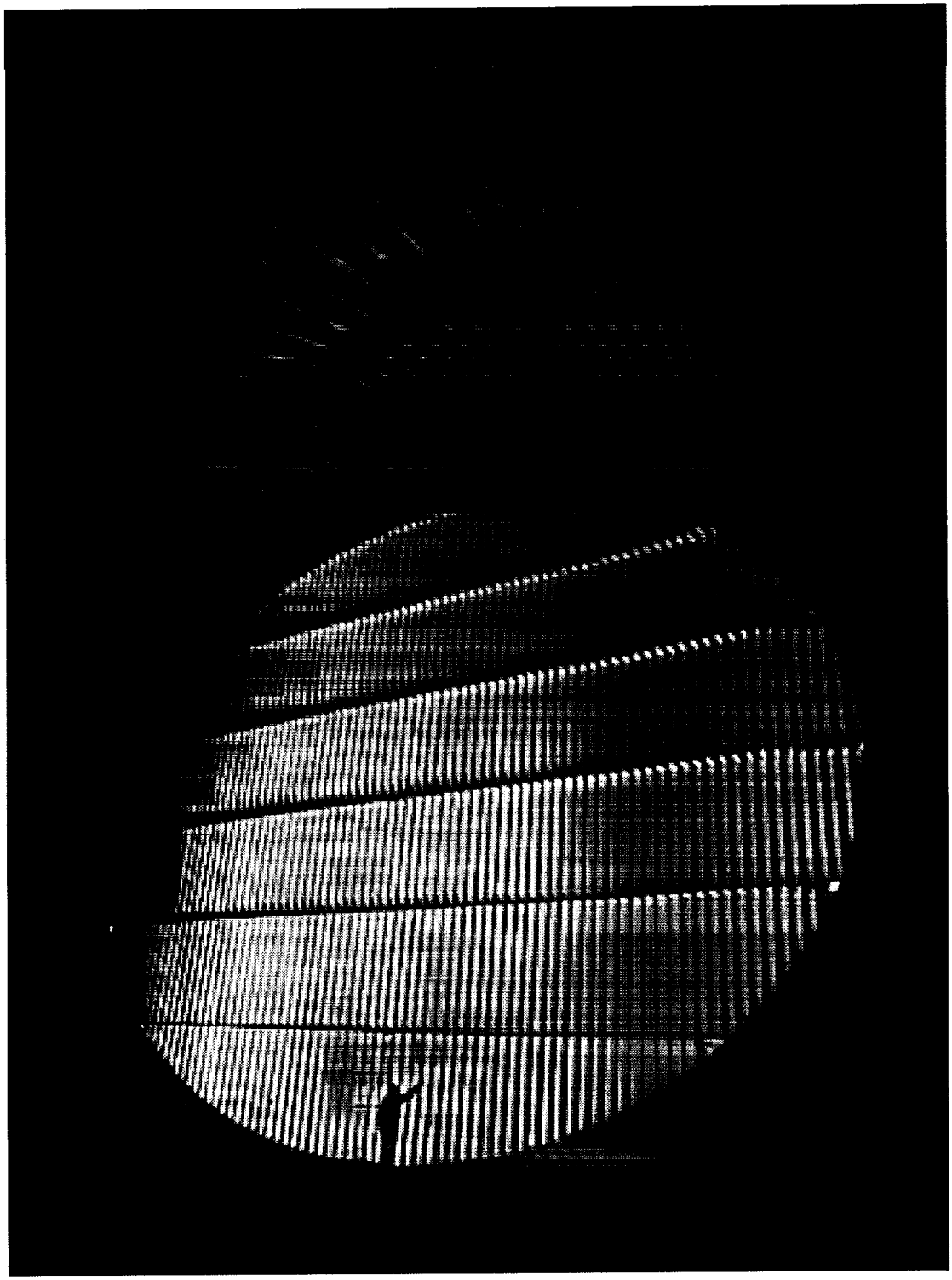


Figure 4. Inside View of the 16-Foot Tunnel downstream of the second set of turning vanes

The test section Mach number, generally referred to as *the* Mach number, is computed from a calibrated ratio of two measured quantities, the airstream stagnation pressure, $P_{stagnation}$, and the plenum static pressure, P_{static}. These two measured quantities are used to calculate the plenum Mach number. A tabulated wind-tunnel calibration provides the correlation between the test section airstream Mach number and the plenum Mach number. The relationship between the two measured pressures and the plenum Mach number, M, is [John, 1984; Mercer et al., 1984]:

$$\frac{P_{stagnation}}{P_{static}} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} \quad ; \quad M = \sqrt{5\left[\left(\frac{P_{stagnation}}{P_{static}}\right)^{\frac{-2}{7}} - 1\right]}; \qquad \gamma_{air} = 1.4. \qquad (1)$$

A large volume of test data relating the tunnel fan drive system control input (+1, 0, -1), and the Mach number, is available for nominal operating conditions over most of the operating range. Data from a typical subsonic run is shown in Figure 5.

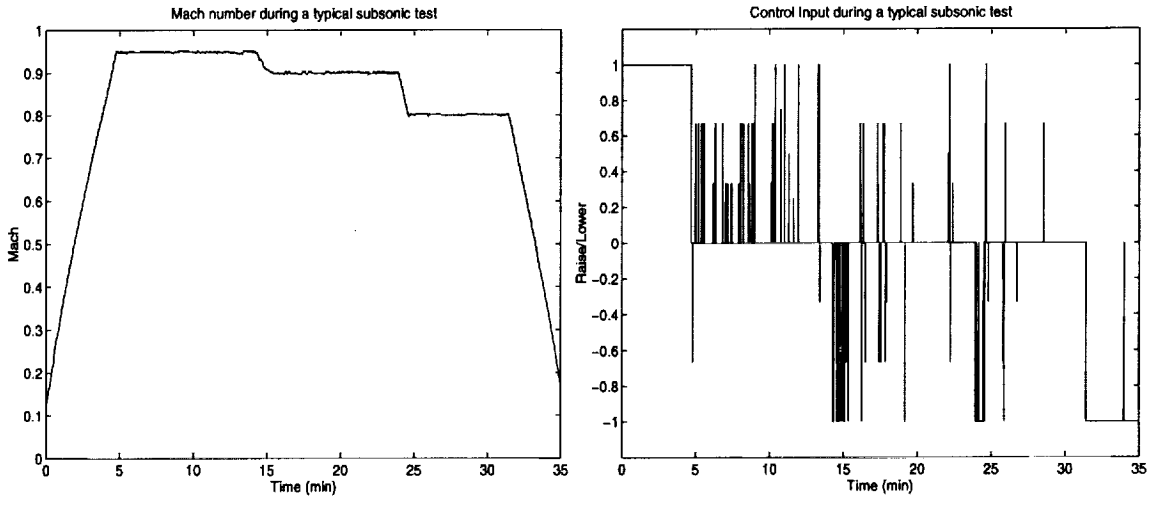


Figure 5. Mach Number and Tunnel Drive Control Inputs during a typical subsonic run

Control Challenges

The problem of controlling the Mach number at the 16-Foot Transonic Tunnel presents the following challenges to any control scheme, including a human operator-in-the-loop:

1. Both the linear and nonlinear characteristics of the tunnel dynamics vary significantly over the operational range of the tunnel. The rate-limited slewing of the tunnel Mach number varies by 50% over the subsonic range, as shown in Table 1. Linearized models identified at individual subsonic operating points contain a set of complex poles with damping ratios ranging from 0.4 - 0.7, and natural frequencies between 1/3 to 1/8 Hz. On the positive side, the open-loop plant is stable, so the control problem is concerned mainly with regulation about the desired set point
2. The control input to the tunnel fan drive system is bang-zero-bang (+1 raise, 0 to maintain speed, -1 lower)
3. The effectiveness of the control input varies by a factor of five over the nominal dynamic range
4. The effectiveness of the control input varies due to degradation of the drive system components, replacement of components, and routine maintenance
5. There is transport lag (pure delay) that varies from 0.3 to 3 seconds over the operational range
6. The Mach number varies with the temperature of the air for a fixed fan RPM

7. The dynamics can change dramatically and abruptly at any given operating point from a particular combination of model attitude and Mach number. This abnormal condition is referred to as a "blocked" tunnel condition
8. The effectiveness of the control input can abruptly change by an order of magnitude for blocked conditions
9. The test section Mach number computed from pressure measurements is noisy and nonstationary
10. The Mach number is to be controlled to within +/- 0.003 of set point
11. Research data is taken with the tunnel in an equilibrium condition, i.e. all Mach number transients have decayed to a minimum, with zero control input to the drive system
12. Power consumption is significant: 20 MW @ Mach 0.7, 80 MW @ Mach 1.3, so the potential for reduction in operating costs is high.

Figure 6 shows a typical operating scenario, with the tunnel under control of an expert human operator. The tunnel is being ramped up from a cold startup condition to a subsonic Mach number of 0.95. A steady raise command from the operator drives the Fan RPM up for approximately five minutes until the desired Mach number is attained. Table 1 shows the variation of the rate-limited increase in Mach number. Once the Mach number is within the 0.003 tolerance, the attitude of the aircraft model under test is stepped through the desired range. For this particular test, the angle of attack was varied directly with the pitch actuator. The tunnel operator is required to make frequent corrective inputs to regulate the Mach number to within the 0.003 tolerance, primarily

due to the rising temperature of the air in the tunnel. The tunnel Mach number is required to be within 0.003 of the set point at each of the angle of attack test values.

This operating scenario highlights the effect of unsteady temperature of the air on the stability of the tunnel Mach number. The tunnel is initially at 85 degrees Fahrenheit, and the temperature at the end of this test sequence is just over 150 degrees and still rising. The rate of temperature rise while ramping to $M = 0.95$ exceeds 10 degrees per minute. The rate of temperature rise decreases rapidly after the initial ramp, but still exceeds one degree per minute at the end of this interval.

t (seconds)	M (Mach)	ΔM	$(\Delta M / \Delta t) * 10^{-3}$
0	0.1119	–	–
30	0.2226	0.1107	3.69
60	0.3333	0.1107	3.69
90	0.4251	0.0918	3.06
120	0.5134	0.0883	2.94
150	0.5971	0.0837	2.79
180	0.6789	0.0818	2.73
210	0.7564	0.0775	2.58
240	0.8285	0.0721	2.40
270	0.9076	0.0791	2.63
282	0.9421	0.0345	2.87

Table 1. Variation of Mach number rate-limited increase while ramping up

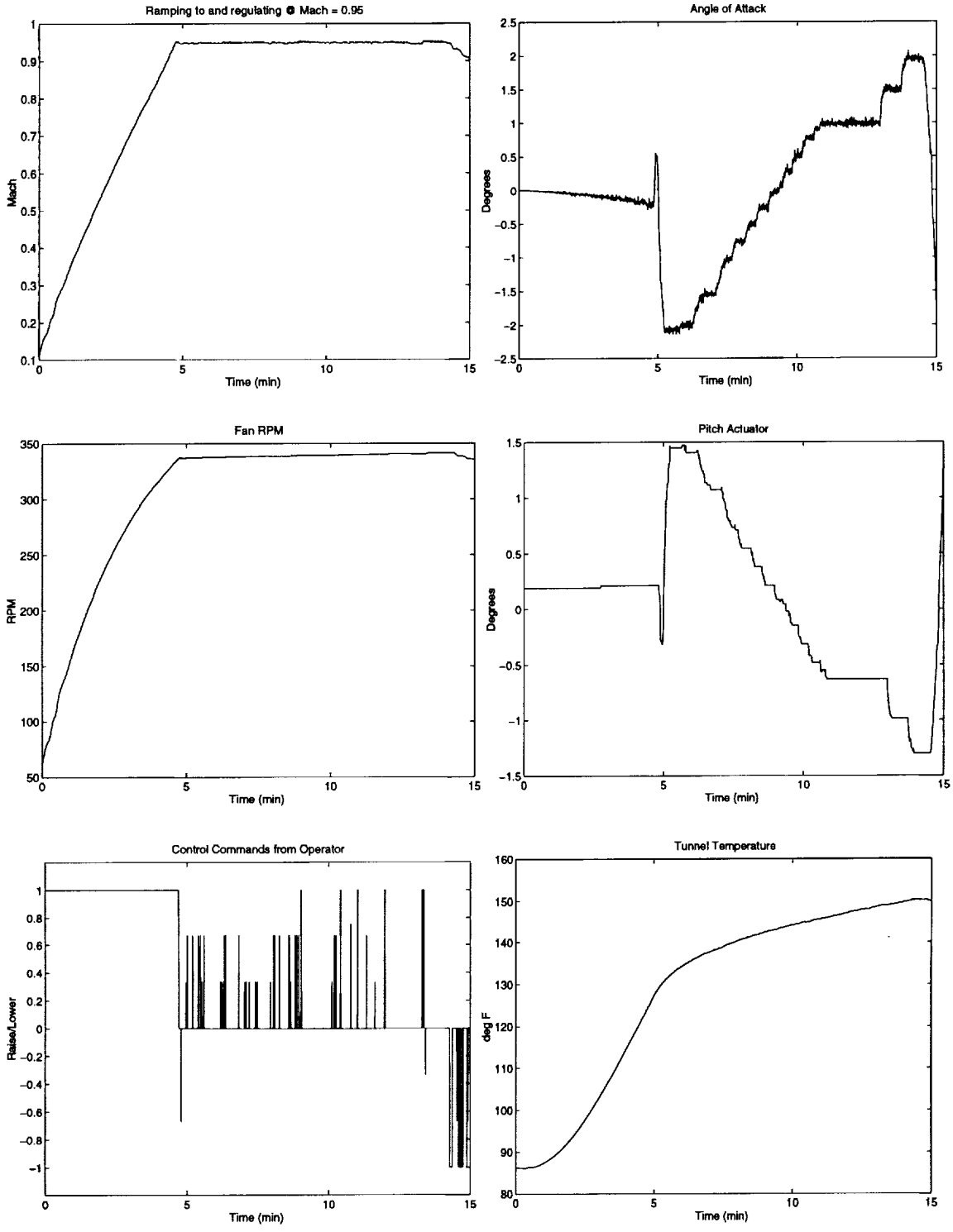


Figure 6. Tunnel conditions during a typical run with steady ramping to the desired test Mach number, M=0.95. The Mach number is to be held to within 0.003 of the desired value while varying the angle of attack.

The relationship of Mach number to temperature is embedded in the definition of Mach number [John, 1984]:

$$M = \frac{V}{a} = \frac{V}{\sqrt{\gamma RT}} \tag{2}$$

where V is velocity of the air, a is the speed of sound, γ and R are constants for air. For a constant air velocity, the variation of Mach number with temperature is:

$$\frac{\partial M}{\partial T} = -\frac{1}{2}\left(\frac{M}{T}\right) \tag{3}$$

with T in degrees Kelvin or Rankine.

At the conditions for this test

$$\left.\frac{\partial M}{\partial T}\right|_{\substack{M=0.95 \\ T=145F}} = -\frac{1}{2}\left(\frac{M}{T}\right) = \frac{-0.000786}{^\circ F}$$

which corresponds to a 0.003 decrease in Mach number for a 3.8 degree F increase in temperature.

Figure 7 illustrates in greater detail the last 3.5 minutes of the test. The test point taken at an angle of attack of one degree takes more than two minutes to acquire. Four corrective inputs applied over a period of more than a minute are required to regulate the Mach number to just barely within the tolerance required for this test point. The next increase in the angle of attack drives the Mach number out of tolerance, which is compensated for by the operator with a longer duration corrective input. During this interval, the effect of the moving the model is relatively small compared to the effect of the rising temperature, but the two can act in combination as illustrated in this example.

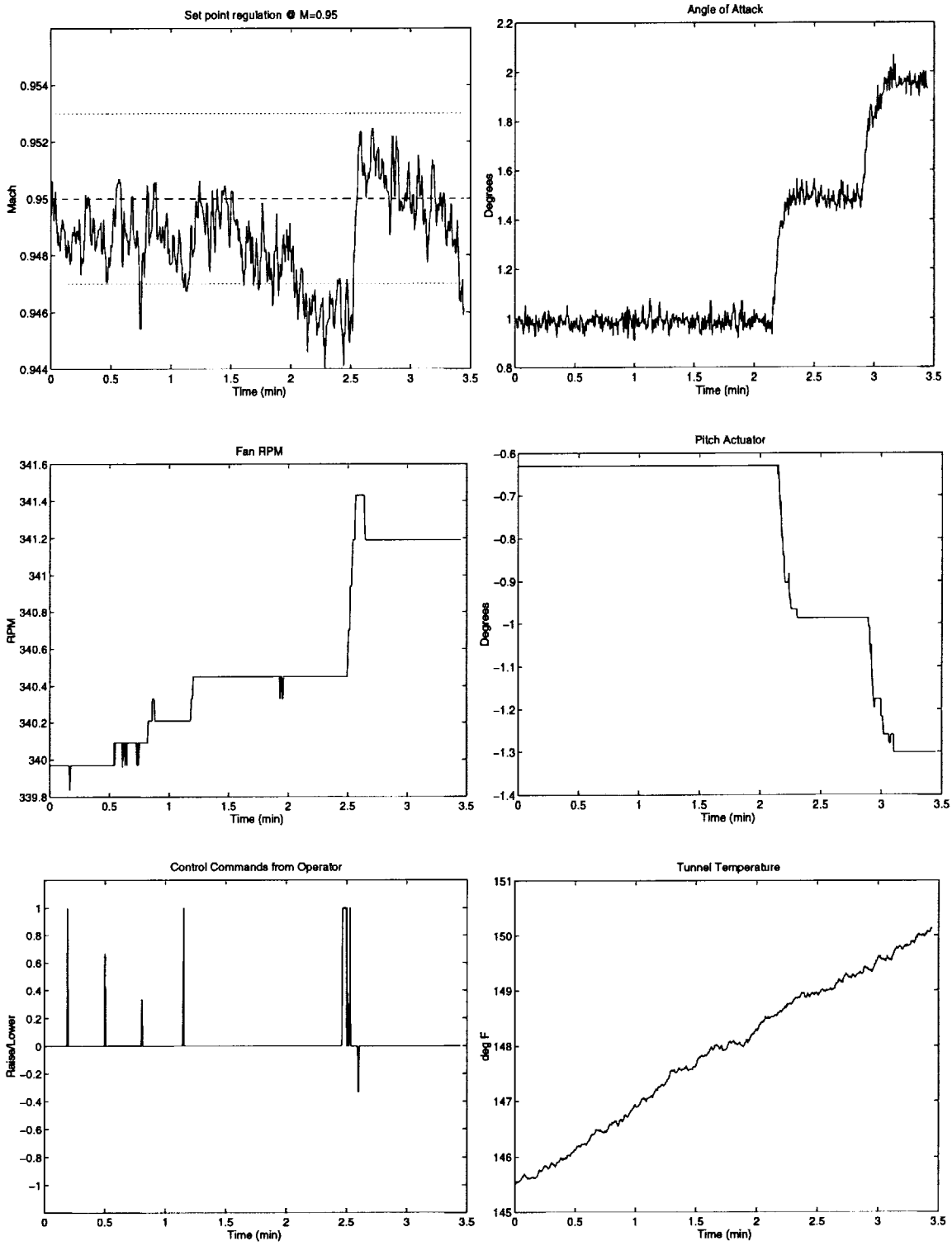


Figure 7. Out of tolerance Mach number extends the test duration during the last 3.5 minutes of the test @ M=0.95.

Figure 8, from a different test, illustrates the effect of large changes in angle of attack disturbing the Mach number under relatively steady temperature conditions. The large change in angle of attack from 5 to 15 degrees in the middle of the test produces a Mach number disturbance of approximately 0.02, or seven times the required tolerance. Here the variation in temperature accounts for only six percent of the total disturbance. The expert operator's response is quite effective in compensating for this disturbance, whereas a non-adaptive automatic controller tuned to the nominal, unblocked dynamics would be unacceptably slow in compensating for this type of disturbance. The effectiveness of the control input decreases abruptly as the model is moved from an angle of attack of five degrees to an angle of attack of fifteen, twenty and twenty-five degrees, respectively. Table 2 lists the changes of control input effectiveness from the nominal condition at five degrees as the model angle of attack is increased. For each large step change in the angle of attack (AOA), the corresponding disturbance is ΔM. The control effort applied to compensate for the disturbance Σu, which is the sample-by-sample sum of the control inputs required to return the Mach number to within tolerance. The effectiveness of the control input is evaluated for each of these cases as $\Delta M / \Sigma u$, simply the ratio of the change in Mach number over the cumulative control effort required to regulate the Mach number. This value is seen to vary by more than an order of magnitude over the test conditions listed in the table. This is a prime example of the kind of variation that motivates the need for multiple models to represent rapidly varying conditions of the plant to be controlled.

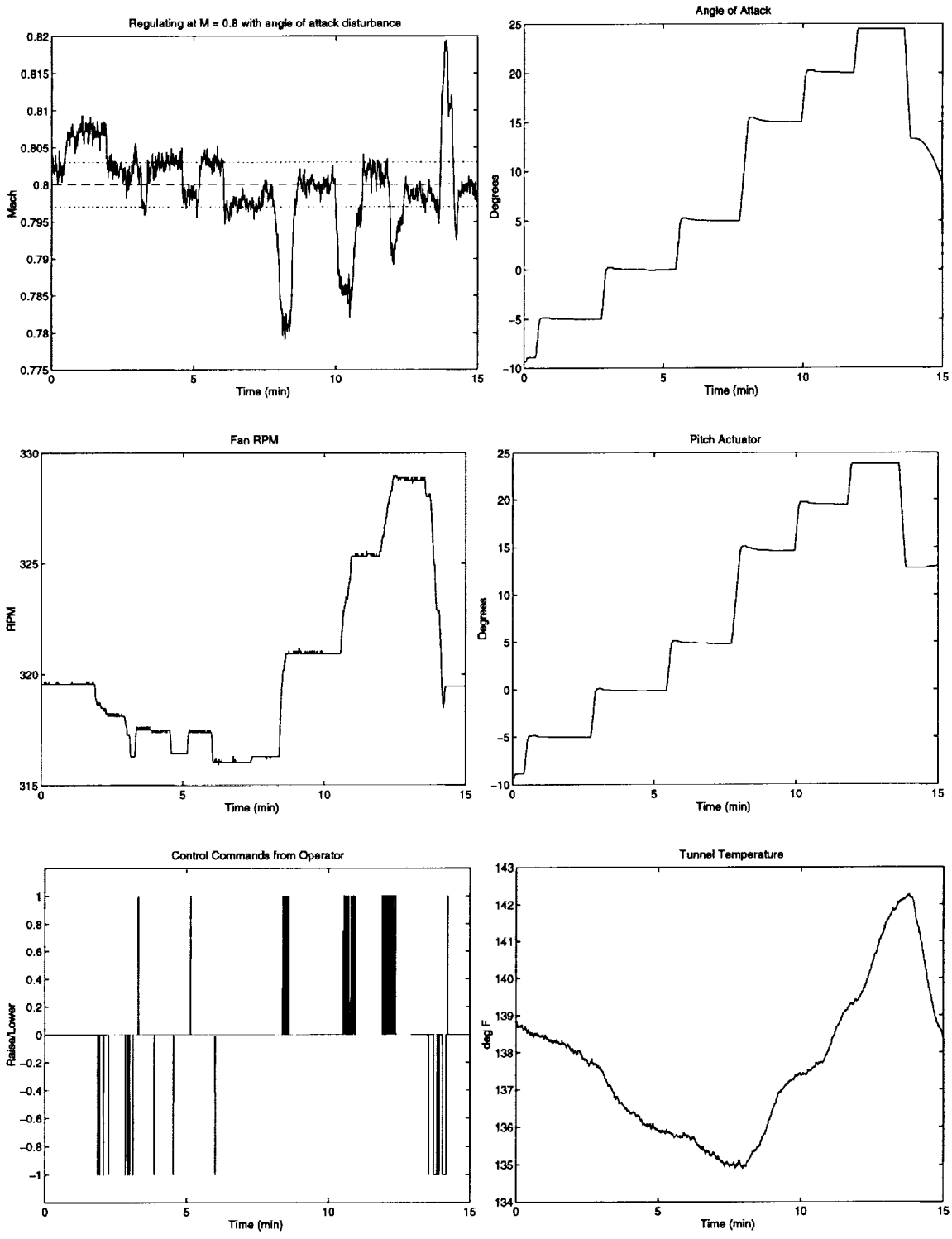


Figure 8. Regulating the Mach number at $M = 0.8$ with large disturbances from the model Angle of Attack

AOA	ΔM	Σu	$\Delta M / \Sigma u$ (10^{-4})
0-5	0.005	1	50
5-15	0.020	27	7.41
15-20	0.017	39	4.36
20-25	0.010	29	3.45

Table 2. Changes in control input effectiveness for blocked conditions

Experimental Framework

The experimental framework that evolved was essentially a predictive control scheme that used multiple models of the plant with *switching*. The controller switches between multiple, SOFM-based models which, collectively, describe the global input-output behavior of the tunnel. The tunnel response to a set of candidate controls is predicted p steps ahead, using the currently selected model. The overall system, which will be referred to in the sequel as the PMMSC, for **P**redictive **M**ultiple **M**odel **S**witching **C**ontroller, is shown in Figure 9. It is composed of the following major functions:

1. The recent control input, $u(k-1), u(k-2), \ldots, u(k-m)$, is clustered on a set of prototype control inputs which will choose one of the Kohonen self-organizing feature maps (SOFM)
2. The selected SOFM identifies the local dynamics of the tunnel based on the past $n+1$ Mach number measurements, $M(k), M(k-1), \ldots, M(k-n)$, and chooses a winning processing element (PE)

3. A linear predictor associated with each PE predicts the Mach number response p steps into the future for each of the candidate controls
4. The predicted effectiveness of the candidate control inputs is evaluated over the last $(p-l)$ steps of the p steps-ahead predictions
5. The control input that provides the best response with respect to the Mach number set point is chosen as the next control, *u(k)*.

The local model associated with the winning PE captures the dynamical regime of the wind tunnel. The controller still must decide what is the most appropriate control input to meet the set point specification. The controller sends candidate input sequences for p-step ahead prediction to the predictor of the winning node. The controller evaluates the relative effectiveness of the candidate control inputs in reducing the error between the predicted Mach number sequence, $\mathbf{M_P}$, and desired Mach number, $\mathbf{M_{SP}}$. This is accomplished by a suitable metric, the Euclidean norm over the error, $\|\mathbf{M_P} - \mathbf{M_{SP}}\|$ where $\mathbf{M_P} = M(k+l+1), M(k+l+2),..., M(k+p)$ and $\mathbf{M_{SP}}$ is a $(p-l)$ length constant vector of M_{SP}. Finally, the control input that provides the smallest error is sent to the wind tunnel fan control.

Overview of the Dissertation

The dissertation is composed of six chapters. Chapter 2 will survey the literature. Chapter 3 will focus on the modeling of the tunnel dynamics. Chapter 4 explains the development of the predictive controller from the local linear models. Chapter 5 describes the experimental setup and results from controlling and modeling the tunnel

responses during operational research runs. Chapter 6 will summarize the results and indicate directions for future research.

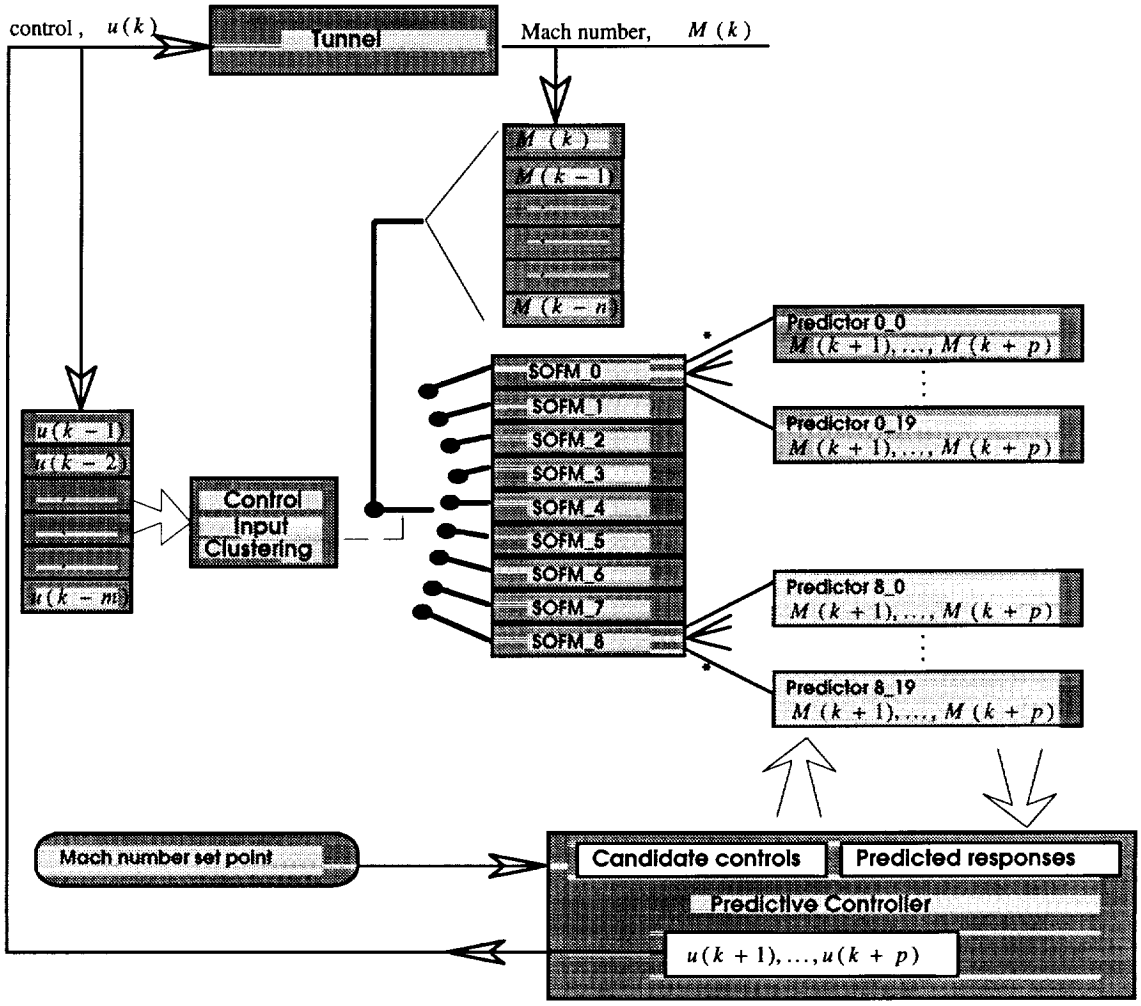


Figure 9. Experimental Framework with PMMSC

CHAPTER 2
REVIEW OF LITERATURE

Introduction

Two major ideas from the existing literature embodied in our system are the Self Organizing Feature Map (SOFM), credited to Kohonen [Kohonen, 1995], and control using multiple models and switching, credited to Narendra [Narendra, Li, and Cabrera, 1994]. In Narendra's multiple model control scheme, an external switching scheme is used to select the model to be used at any given instant of time. In the experiment described in this dissertation, the SOFM is used as the modeling infrastructure, with selection of the model done by the activity of the output neural field or *winner*. A description of both of these topics, as well as a brief review of adaptive control, SOFM applications to control, and more general review of applications of neural networks to control follow.

Self-Organizing Feature Map

The self-organizing feature map (SOFM) was adopted as the neural architecture for the experiment. The SOFM was chosen based on its ability to transform an incoming signal of arbitrary dimension into a lower dimensional, discrete, topologically ordered map, one dimensional in this case. The spatial location of the neurons, arranged in a one dimensional lattice, or linear array, corresponds to intrinsic features of the input signal.

The SOFM belongs to the class of artificial neural networks that use competitive or unsupervised learning. In contrast to supervised learning, the SOFM input-output behavior is *not* learned from a set of training examples which specify the desired output $y \in R^m$, for a given input $x \in R^n$, where the parameters of the network are adjusted by the backpropogation algorithm [Rumelhart, Hinton, and Williams, 1986; Werbos, 1990]. In feedback networks [Hopfield, 1982], the other major category of artificial neural networks, the input defines an initial state of activity of a feedback system which settles to a final asymptotic state that represents the response to the given input. In the SOFM, however, neurons *compete* to respond to the input signal, with the result that only one output neuron is fired or activated. The output neuron activated in response to a particular input is called the *winner,* while all the other neurons are inhibited, representing a winner-take-all (WTA) structure. During the training phase, the SOFM becomes topologically ordered by adapting the weights not only of the winner, but those of the neighboring neurons as well. This is inspired by lateral inhibitory feedback in biological neurons [Willshaw and von der Malsburg, 1976], but implemented in the SOFM by a computational shortcut, referred to as the neighborhood function. Not only do the individual neurons in the SOFM become specifically tuned to input patterns by means of this emulation of lateral feedback among neighboring units, but the locations of responses become ordered along the coordinates of the map, corresponding to intrinsic features of the input.

Let the input be a vector $\mathbf{x} \in R^n$:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T. \tag{4}$$

With each neuron j there corresponds a vector of synaptic weights $\mathbf{w} \in R^n$:

$$\mathbf{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jn}]^T. \tag{5}$$

The winner is identified by the index $i(\mathbf{x})$ that corresponds to the neuron whose synaptic weights are the best match to the input **x:**

$$i(\mathbf{x}) = \arg\min_j \left\| \mathbf{x} - \mathbf{w}_j \right\|, \quad j = 1,2,\ldots,N \tag{6}$$

where $\|\bullet\|$ denotes the Euclidean norm. Thus, the response of the network can be considered to be the index of the winning neuron, representing its location, or, equivalently, the synaptic weight vector that is closest to the input vector in a Euclidean sense [Haykin, 1994]. In this experiment, the latter interpretation of the network response is more appropriate.

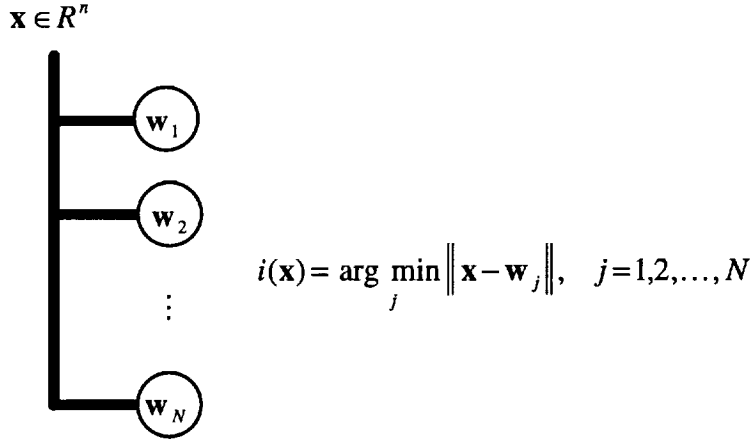


Figure 10. SOFM with a one-dimensional array of neurons

For the formation of an ordered map, it is crucial that the weights of the winner are *not* updated independently from the weights of the other neurons, as is the case of other competitive learning or vector quantization schemes. In the SOFM, the adaptation or updating of the weight vectors is done over a topologically related subset, resulting in weight vectors that are ordered along the output dimension of the network. At each learning step, the network is presented a sample **x**, drawn from the input distribution. The winner is determined as specified in (6), and a neighborhood set $N_{i(\mathbf{x})}$ identifies the neurons around the winner that will be updated as well. The width or radius of $N_{i(\mathbf{x})}$ is usually varied over the training phase [Kohonen, 1990]. To achieve good global ordering, $N_{i(\mathbf{x})}$ is made very wide at the beginning of the training, on the order of the one-half the map, and then shrinks monotonically as the training progresses. The rationale for this [Kohonen, 1990] is that the wide initial $N_{i(\mathbf{x})}$, corresponding to a coarse spatial resolution in the learning process, first induces a rough global ordering over the weight vectors. Then, as the $N_{i(\mathbf{x})}$ narrows, the spatial resolution of the map improves without destroying the acquired global order. Thus the use of the neighborhood function emulates the formation of a localized response in biological neurons by initially applying a strong positive lateral feedback corresponding to an ordering phase, followed by negative lateral feedback which corresponds to a convergence phase.

The updating of the weight vectors in discrete time proceeds as :

$$\mathbf{w}_j(k+1) = \begin{cases} \mathbf{w}_j(k) + \alpha(k)\left[\mathbf{x}(k) - \mathbf{w}_j(k)\right] & \text{if } j \in N_{i(\mathbf{x})} \\ \mathbf{w}_j(k) & \text{if } j \notin N_{i(\mathbf{x})} \end{cases} \tag{7}$$

with $\alpha(k)$ a scalar learning rate parameter, $0<\alpha(k)<1$, similar to the gain used in stochastic approximation processes [Robbins and Monroe, 1951], and should decrease over the training interval.

Practical Aspects for the Application of the SOFM Algorithm

1. The initial weight vectors $\mathbf{w}_j(0)$ are set to random values.
2. Samples $\mathbf{x}$ are drawn from the input distribution and presented to the network.
3. The best matching neuron is determined by (6).
4. The weight vectors of all the neurons are updated by (7).
5. Steps 2 through 4 are repeated until no noticeable changes are observed.

The "rules of thumb" are that for approximately the first 1000 steps, $\alpha(n)$ should be close to unity, then decrease monotonically. The actual rule for the decrease is not critical. The ordering of the map takes place during this period. The neighborhood function $N_{i(\mathbf{x})}$ should be fairly wide initially, perhaps on the order of half the map, and decrease linearly to one unit during this ordering phase. After the first thousand steps, a much longer convergence or fine-adjustment phase of the training proceeds with the learning rate $\alpha(n)$ slowly decreased to a value near 0.01. During this phase the neighborhood function may contain the nearest neighbors of the winner, with the final stages of the convergence phase updating only the winner. A rule of thumb for the number of steps to achieve convergence is at least 500 times the number of network units.

The following figures illustrate an example of training an SOFM used in the experiment. The inputs to the SOFM are a 50 sample window of the Mach number

response. There were 155 exemplars in the training set, shown in Figure 11. The SOFM consisted of 20 neurons, arranged in a linear array, similar to Figure 10, shown earlier. The SOFM weights were adjusted during 10,000 presentations of the training set, with the learning rate, $\alpha(n)$, and neighborhood function $N_{i(x)}$ varied as shown in Figure 12. The SOFM is shown at 100, 500, and 1000 training cycles, with the converged SOFM, after 10,000 training cycles, in Figure 13. The converged SOFM provides a smooth organization of the weights in the neural field, in contrast with the input patterns for training. The distribution of the training inputs among the converged SOFM clusters is shown in Figure 14.

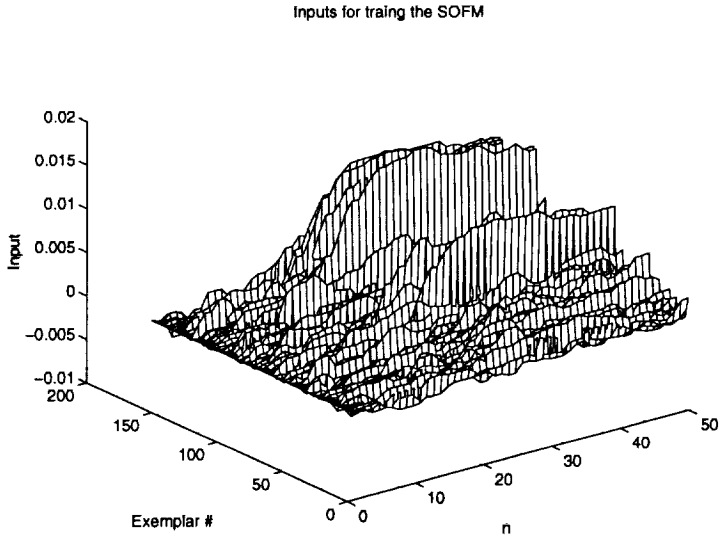


Figure 11. Input exemplars for training the example SOFM

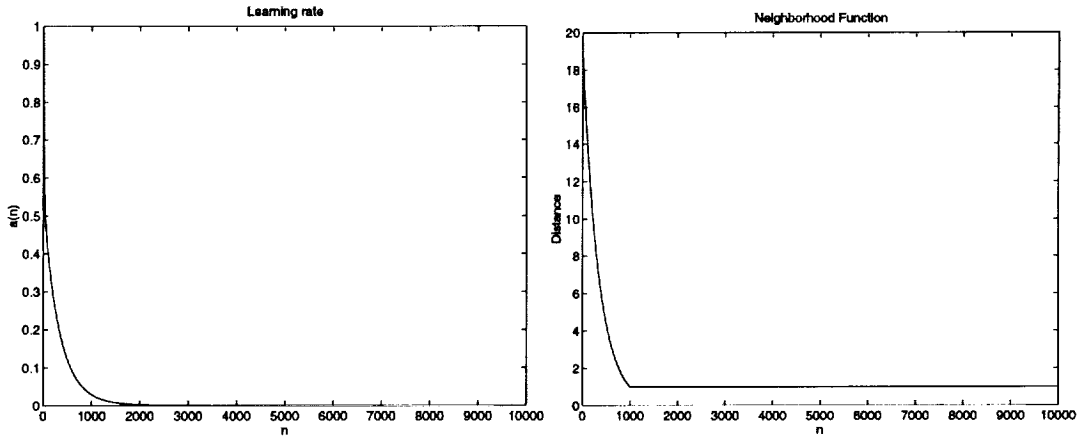


Figure 12. Learning rate and Neighborhood function during training

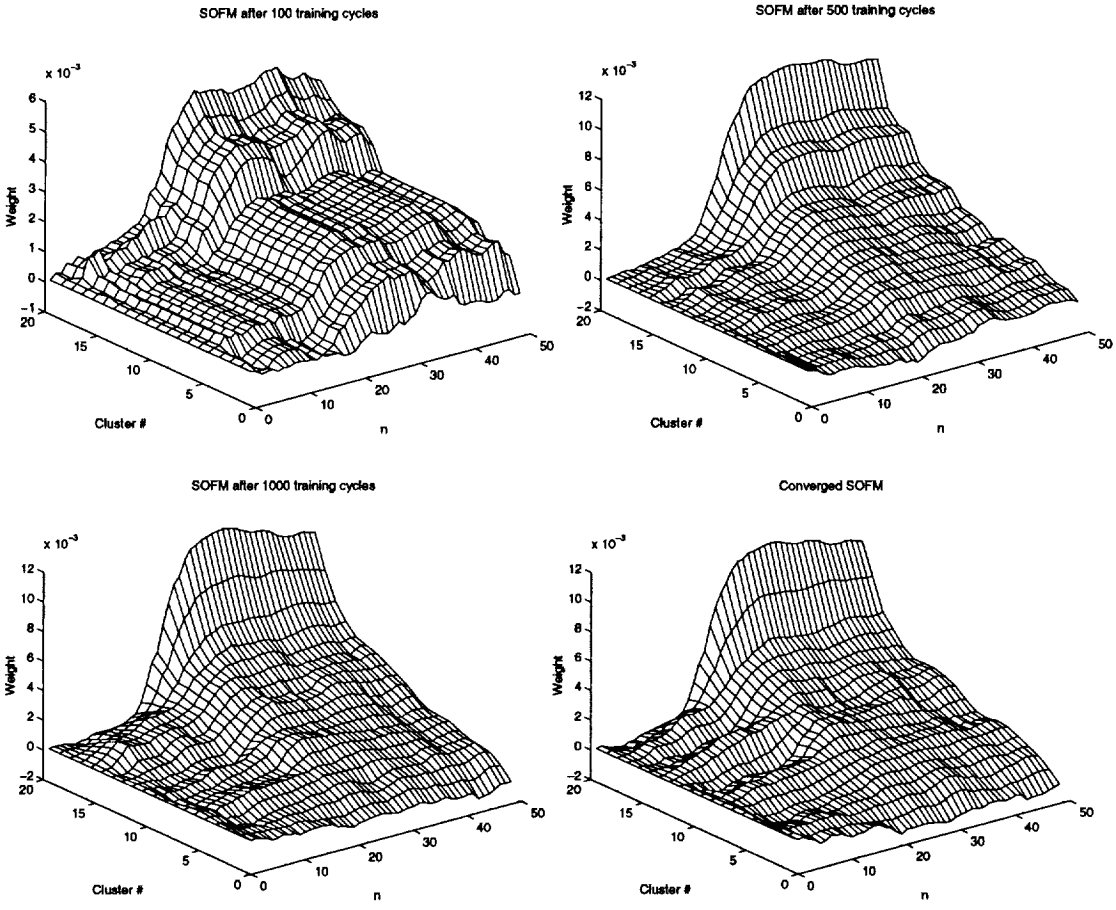


Figure 13. SOFM during various points in the training

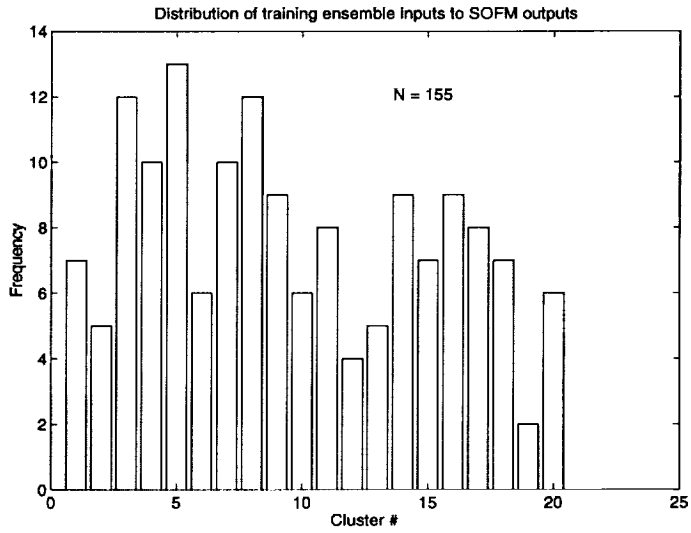


Figure 14. Distribution of training inputs among 20 converged SOFM clusters

Magnification Factor

The input distribution of the vectors **x**, or the multidimensional probability density function (pdf) of **x**, $p(\mathbf{x})$, is represented by the total N neurons in the output layer of the SOFM. The input vectors **x** are drawn from an n-dimensional input space X. The pdf of **x**, integrated over all of X, must equal unity:

$$\int_{-\infty}^{\infty} p(\mathbf{x})\, d\mathbf{x} = 1 \tag{8}$$

The corresponding density of neurons in the output layer of the SOFM is referred to as the *magnification factor,* $m(\mathbf{x})$, defined as the number of neurons in a small volume $d\mathbf{x}$ of the input space X. The integral of the magnification factor over the entire input space, must equal the total number of neurons N:

$$\int_{-\infty}^{\infty} m(\mathbf{x})\, d\mathbf{x} = N \tag{9}$$

For the SOFM to match the input density exactly, the magnification factor must be directly proportional to the input pdf:

$$m(\mathbf{x}) \propto p(\mathbf{x}) \tag{10}$$

Si and Lin [1997], have recently shown, for multidimensional input, the converged SOFM weights have a magnification factor proportional to $p(\mathbf{x})^{\frac{n}{n+2}}$. Kohonen [1995] makes the point that in most practical applications that the input data vectors have high dimensionality, on the order of dozens to hundreds, and compares the result to classical vector quantization (VQ), where the asymptotic point density is proportional to $p(\mathbf{x})^{\frac{n}{n+2}}$ as well. For this experiment, the input dimension n, is $n = 50$, so it was expected that the input distribution would be well matched by the locations of the output neurons of the SOFM. From a control viewpoint, this has the beneficial effect of providing a higher density of neurons in regions of the input space where the statistical frequency of input features is correspondingly higher, with fewer neurons assigned to regions of the input space with features of lower statistical frequency.

Applications of the SOFM

Three major practical application areas suggested by Kohonen [Kohonen, 1995] where the SOFM could be used effectively are:

1) Industrial and other instrumentation, for both monitoring and control
2) Medical applications, for diagnostic methods, prostheses, and modeling

3) Telecommunications, for allocation of resources to networks, transmission channel equalization, and adaptive equalization.

A survey of the diverse applications of the SOFM [Kohonen, 1995] highlights the following areas: machine vision and image analysis, optical character and script reading, speech analysis and recognition, acoustic and musical studies, signal processing and radar measurements, telecommunications, industrial and other real-world measurements, process control, robotics, chemistry, physics, electronic-circuit design, medical applications without image processing, data processing, linguistic and AI problems, mathematical problems, and neurophysiological research. The reported applications in process control were of interest, but, for the most part, the research focused on monitoring the process state rather than effecting some control action. Some general problems addressed in this area are: identification of process state [Kasslin, Kangas, and Torkkola, 1992], process error detection [Alander et al., 1991], and diagnosis of machine vibrations [Wu et al., 1991]. Some specific examples of industrial applications are: monitoring paper machine quality [Lampinen and Taipale, 1994], flow regime identification [Cai, Toral, and Qiu, 1993], grading of beer quality [Cai, 1994], and estimation of torque in switched reluctance motors [Garside et al., 1992]. In a more recent application to process control, [Matthews and Warwick, 1995] the SOFM was used for separating fault types and monitoring the process state. In [Warwick, 1996] the SOFM is proposed again as a classifier for fault indications as opposed to a system identification tool.

One of the most control-specific applications of the SOFM reported in the literature is the visuomotor control of a robot arm by [Ritter, Martinetz, and Schulten, 1992]. In this application, the SOFM is used as a look-up table, where the input pattern,

identified by the "winner", specifies an SOFM location associated with specified values of control parameters, which were learned adaptively.

The two dimensional coordinates, x_1 and x_2, of a target point in the image planes of two cameras were combined into a four-dimensional, stereoscopic input vector x and used as the input to the SOFM. A three-dimensional SOFM was used to form the spatial representation of the target point. The three joint angles, one about the vertical axis for motion in the horizontal plane, and two for motion in the vertical plane, comprise a configuration vector $\theta=[\theta_1, \theta_2, \theta_3]$. The basic goal of their approach was to find the transformation $\theta(x)$ that would bring the tip of the robot arm to the target point, where the cameras can get the observation x. The configuration vector is determined by a linearization about the origin determined by the "winner" location c:

$$\theta = A_c(x - m_c) + b_c. \tag{11}$$

Here b_c is the configuration vector corresponding to the location m_c, A_c is the 3x4 Jacobian matrix, m_c is from the weights of the SOFM winner, and (11) gives the first two terms of the Taylor series expansion of $\theta(x)$ around m_c. Linearization is carried out around m_c and is valid in the whole Voronoi set of x values around m_c. Ritter et al., developed a learning scheme where the control parameters A_c, b_c were updated simultaneously with the formulation of the SOFM. The importance of the SOFM in their problem was the discretization of the input space, in particular, the allocation of the configuration vectors, b_c, to regions of the input vectors, x, having a higher density of lattice points where the control must be more accurate.

For our application, the SOFM discretizes an n-dimensional space composed of output sequences of the system, *y(k), y(k-1), ... y(k-(n-1))*, which are considered to be the responses of the system to a prototype control input *u(k-1), u(k-2), ... u(k-m)*. Thus, the prototype input is the control parameter associated with all the nodes in the lattice, which is here, one-dimensional corresponding to the single control input to the system, u. In our application, the linearization is done around the "winner" to predict responses to candidate controls:

$$M_p = A_c(u_i - u_c) + M_c \ . \tag{12}$$

where M_c is the winner, A_c, is the Jacobian, derived directly from the SOFM, u_c is the control prototype associated with the SOFM and u_i is one of the candidate control sequences. In our application we replace the slow adjustment of control parameters by an external scheme, as in Ritter's application, with the ability to switch, at discrete intervals, among the discrete local linear models associated with each node in the SOFM. This highlights the difference between a slowly adaptive control scheme, and our application, which is designed to switch rapidly to accommodate abrupt changes in the system characteristics.

A Brief Review of Adaptive Control

The adaptive identification and control of dynamical systems has been extensively developed for linear time-invariant systems with unknown parameters over the past three decades. The development of adaptive control for linear systems is a logical consequence of the diversity of mathematical tools available for the analysis of the

properties of linear systems. The choice of parameterization of the plant model and the controller in such problems were naturally based on results from linear systems theory. In the 1980's, the theory of adaptive control focused on the design of stable adaptive control laws which are robust in the presence of unmodeled disturbances, time-varying parameters and unmodeled dynamics [Narendra and Annaswamy, 1989]. A good understanding exists for the design of stable adaptive controllers for linear systems with unknown parameters.

Two major approaches to the adaptive control of linear systems, *direct* and *indirect*, have developed over the past twenty years. The direct approach seeks to minimize some performance criteria, usually based on the error between the output of the system and some desired output, by direct adjustment of the controller parameters. The indirect approach attempts to explicitly identify the dynamics of the system to be controlled, and then modifies the parameters of the controller based on this identification. Both of these methods traditionally used a single, linear, parameterized model of the system being controlled, or *plant.* One of the major drawbacks of both these approaches, is the time required for adaptation of the controller parameters in the direct case, or the identification of the parameters of the plant in the indirect case, to achieve the desired control. This is particularly troublesome when the method is to be applied on-line to control processes whose dynamic behavior is known to change abruptly.

As a result of the shortcomings mentioned above, a more recent approach to the adaptive control of an uncertain linear time-invariant system (LTI), is the use of multiple models with switching [Narendra and Balakrishnan, 1997]. Although this was not the first time that the individual concepts of multiple models, with switching and on-line

tuning of some models, had been proposed, this framework proposed to improve the transient response of adaptive systems in a stable fashion [Narendra and Balakrishnan, 1994]. The recent results present the problem in the context of model reference adaptive control (MRAC) [Narendra and Annaswamy, 1989] of a LTI system, and the principle results are the proofs of stability for various assumptions on the coverage of the space $S \subset \Re^{2n}$ of the plant parameters by either the initial parameter values of a set of adaptive models or the parameters of a set of fixed models, and various combinations of both fixed and adaptive models. The multiple model and switching framework is quite general and applies to both linear and nonlinear systems, but stability results are only currently available for the linear time-invariant plants.

The development of nonlinear adaptive control has for the most part, paralleled the linear case, usually with even more restrictive assumptions about plant than the linear case. The usual approach is to perform a linearization of the plant model around some point in the state space, determine the localized characteristics of the linearized system, and the region in the state space where the linearization is valid.

Linear Adaptive Control

A single input-single output (SISO) linear time-invariant system with unknown parameters, described by the state equations:

$$\begin{aligned} x(k+1) &= Ax(k) + bu(k) \\ y(k) &= cx(k) \end{aligned} \tag{13}$$

corresponds to the case where some or all of the parameters of the matrix A and vectors b and c are unknown. Alternately, if the system is described by the n^{th} order difference equation:

$$y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + \sum_{j=0}^{n-1} \beta_j u(k-j) \tag{14}$$

where $u(k)$ and $y(k)$ represent the input and output respectively at time k, the parameters α_i and β_j are assumed to be unknown. The objective then is to determine the control input $u(k)$ so the output $y(k)$ behaves in some desired fashion.

The transfer function, $W_p(z)$, of the plant described by equation (14) is :

$$W_p(z) = \frac{\beta_0 z^{n-1} + \beta_1 z^{n-2} + \ldots + \beta_{n-1}}{z^n - \alpha_0 z^{n-1} - \ldots - \alpha_{n-1}} \tag{15}$$

The order of the system is n and if $\beta_0 \neq 0$, then the relative degree is one. If, however, $\beta_0 = \beta_1 = \beta_2 = \ldots = \beta_{d-2} = 0$ and $\beta_{d-1} \neq 0$, then the relative degree is d and the input $u(k)$ affects the output at time instants greater than or equal to $k+d$. It is best to first consider the case when the relative degree is one, then extend the results to the case when the relative degree is greater than one.

A bounded signal $y^*(k)$ is specified as the desired output of the plant and the input $u(\bullet)$ is to be determined. Alternately, $u(k)$ at instant k has to be chosen so that

$$\lim_{k \to \infty} \left| y(k) - y^*(k) \right| = 0. \tag{16}$$

In model reference adaptive control (MRAC), $y^*(k)$ is generally chosen as the output of a reference model. The simplest reference model that can be satisfied by (16) above is z^{-d} where d is the relative degree of the plant. For the case where the relative degree is one, the reference input *r(k)* to the reference model is $y^*(k+1)$ and is assumed to be known at time *k*.

For the non-adaptive problem, if the plant is described by equation (14) and the parameters α and β are known, the control law can be chosen as :

$$u(k)=\frac{1}{\beta_0}\left[-\sum_{i=0}^{n-1}\alpha_i y(k-i)-\sum_{j=1}^{n-1}\beta_j u(k-j)+y^*(k+1)\right] \tag{17}$$

and then the output $y(k)=y^*(k)$. The control input is merely a linear combination of n past values of the input and output as well as the desired signal at instant k+1, and that the output of the plant converges to the desired output in one instant.

For the adaptive case where the parameters α and β are assumed constant but unknown, the indirect approach can be employed and requires the estimation of the parameters α and β. If $\hat{\alpha}_i(k)$ and $\hat{\beta}_j(k)$ represent the estimates of α and β respectively, these can be used to compute the control input. However, it is no longer obvious that the overall system will be stable and that the condition (16) will be satisfied. This problem was resolved for both continuous-time and discrete-time systems in 1980 [Narendra, Lin, and Valavani, 1980; Morse, 1980]. However the stability of the overall system in the discrete case requires the following assumptions about the plant transfer function:

1) An upper bound on the order n is known
2) The relative degree of the plant is known
3) The sign of β_0 as well as an upper bound on the absolute value of β_0 are known
4) The zeroes of the plant transfer function are within the unit circle (minimum phase condition).

Given these assumptions, stable adaptive laws for the adjustment of the estimates $\hat{\alpha}_i(k)$ and $\hat{\beta}_j(k)$ result in a similar control law:

$$u(k)=\frac{1}{\hat{\beta}_0}\left[-\sum_{i=0}^{n-1}\hat{\alpha}_i y(k-i)-\sum_{j=1}^{n-1}\hat{\beta}_j u(k-j)+y^*(k+1)\right] \tag{18}$$

where the output $y(k)$ follows $y^*(k)$ asymptotically.

Control Using Multiple Models and Switching

The multiple model structure with switching has been proposed by [Narendra et al.; 1994, 1995] when the overall system is required to operate in multiple environments. Sudden changes in parameter values, failures of sensors or subsystems, and external disturbances taken to be the output of an unforced stable dynamical system, can be considered as different environments a control system may be required to cope with. In these cases, the need to use multiple models arises naturally, since a different mathematical model may be needed to represent the behavior of the plant in each of the environments.

The need for multiple models in the control of dynamic systems is further elaborated by [Narendra, 1996] as:

1) Many physical systems can be represented by interpolating between local models. Gain scheduling is the control paradigm based on this concept
2) Multiple models may be needed to detect different changes in the plant and initiate the appropriate control action
3) In some cases, all the information concerning the plant, such as the order or the relative degree, may not be available to compute the input. Multiple models may be needed to obtain the appropriate information
4) The advantages of individual models may be combined in a multiple model controller. One model may assure stability, while another heuristically designed may provide better performance. A proper combination of the two may result in a stable system with better performance.

The architecture of the Narendra's multiple model switching controller is shown in Figure 15. $I_1, I_2, \ldots, I_n$ are N predictive models of the plant which have been obtained by observing the system over a long period of time. $C_1, C_2, \ldots, C_n$ are the corresponding controllers, designed off-line and stored in memory. If the plant output is $y(k)$ and the output of model I_j is $\hat{y}_j(k)$, the output error is defined as $e_j = \hat{y}_j(k) - y(k)$. Based on some performance index $J(e_j)$, evaluated for $j = 1, 2, \ldots, N$, the model to be used at any instant is chosen. If $J_i(k) = \min_j J(e_j(k))$, the model I_i and the corresponding controller C_i are chosen at instant k. This corresponds to the switching part of the scheme. The implementation of the switching scheme employs some hysteresis to

prevent arbitrarily fast switching between models. In a more recent paper [Narendra and Balakrishnan, 1997], stability results for an all-fixed models controller was established for linear systems under some mild assumptions. In particular, it is shown that if there is at least one model that is sufficiently close to the actual plant and there is a non-zero waiting time between switches from one model to another, then the overall system is stable, given that each fixed model is stabilized by its corresponding controller.

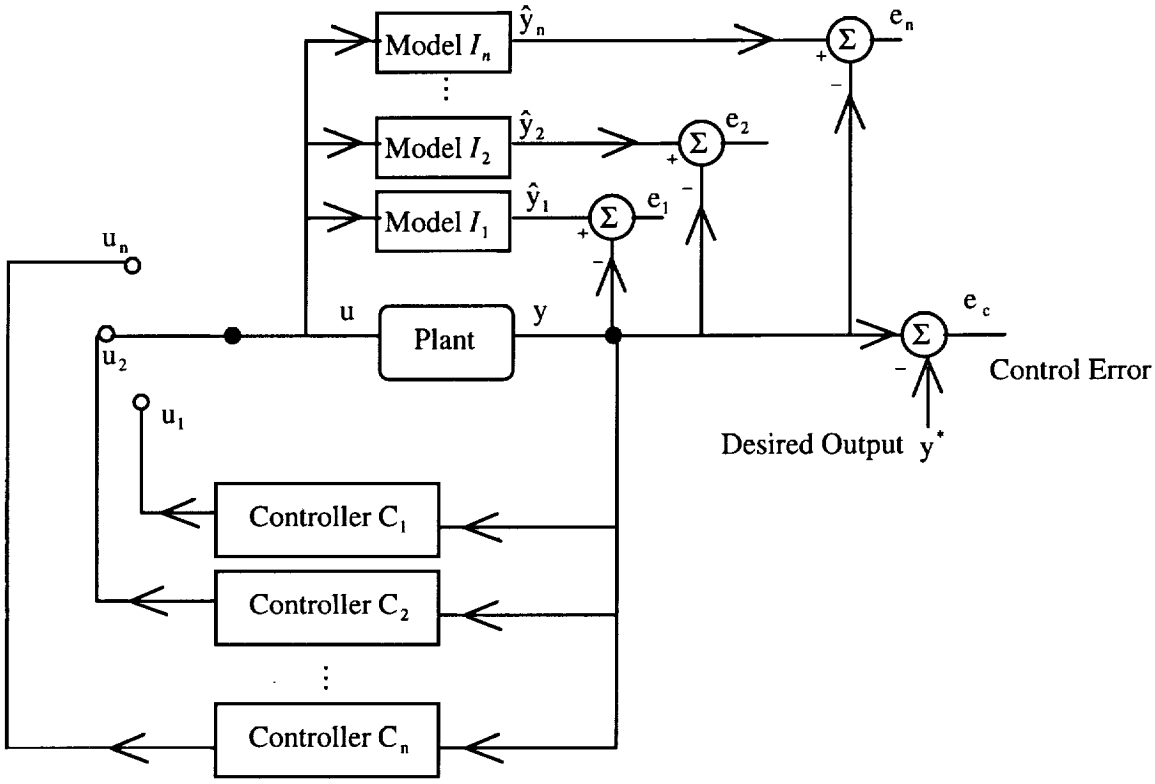


Figure 15. Structure of the multiple model control with switching

An even more recent paper [Narendra and Mukhopadhyay, 1997] introduces two classes of approximate non-linear input-output models which reduce the computational complexity of designing a controller based on the fact that the approximate models are

linear in the control input. This was essentially the approach taken in this experiment, where the converged SOFM provides multiple, approximate models of the input-output behavior of the plant for a given class of input. These approximate models were then used as the basis for linear predictions of the response to a set of control candidates to determine the control input that minimized the error between the predicted output and the desired output.

The development of these models begins by considering the representation of an arbitrary, discrete non-linear dynamical system using state equations:

$$\begin{aligned} \Sigma: \; x(k+1) &= f[x(k), u(k)] \\ y(k) &= h[x(k)] \end{aligned} \tag{19}$$

where $\{u(k)\}, \{x(k)\}$, and $\{y(k)\}$ are discrete-time sequences with $x(k) \in \Re^n$, $u(k) \in \Re$, $y(k) \in \Re$, $f: \Re^n \times \Re \rightarrow \Re^n$, $h: \Re^n \rightarrow \Re$, and $f, h \in C^\infty$. The origin is assumed to be an equilibrium state of (4), hence $f(0,0) = 0$. The linearization of Σ_L of Σ is described by the linear state equations:

$$\begin{aligned} \Sigma_L: \; x(k+1) &= Ax(k) + bu(k) \\ y(k) &= cx(k) \end{aligned} \tag{20}$$

where the $(n \times n)$ matrix A and the $(n \times 1)$ and $(1 \times n)$ vectors b and c are defined by

$$\left.\frac{\partial f(x,u)}{\partial x}\right|_{0,0} = A \qquad \left.\frac{\partial f(x,u)}{\partial u}\right|_{0,0} = b \qquad \left.\frac{\partial h(x)}{\partial x}\right|_{0} = c \tag{21}$$

Given this parameterization, the general state of knowledge about the system Σ can fall into one of the following categories:

1) f and h are known, and the state $x(k)$ is accessible

2) f and h are unknown, and the state $x(k)$ is accessible

3) f and h are unknown, and only the input $u(k)$ and the output $y(k)$ are accessible.

The third case is the one of interest here, where system identification and control have to be carried out using only input-output data.

Other Applications of Neural Networks for Control

Three recently reported neural network applications for control appeared in the July 1997 edition of the *IEEE Transactions on Neural Networks*. The first paper, "Reliable Roll Force Prediction in Cold Mill Using Multiple Neural Networks" [Cho, Cho, and Yoon, 1997] reported the use of multilayer perceptrons to predict the roll force and a corrective coefficient used to improve prediction accuracy by 30-50 % compared to an existing mathematical model used in the cold rolling mill process for steel. The second paper, "Dynamic Neural Control for a Plasma Etch Process" [Card, Snidermann, and Klimasauskas, 1997] described the use of a cascade (feedforward) neural network and a policy-iteration optimization scheme to provide suggested process setpoints for recovery from long-term drift in equipment used in the plasma etch process. The combined optimization scheme suggested "reasonable low cost solutions" for what were considered out-of-control situations. The third paper, "Neural Intelligent Control for a Steel Plant" [Bloch et al., 1997] suggests incorporating the skill of the human operators in neural models, at various levels of control. A feedforward multilayer perceptron is developed as a model of the annealing furnace, from which a static inverse model is derived. None of the three papers had any experimental results from actually employing the neural-based control to the targeted process.

The most specific reference citing the use of neural networks for wind tunnel control was [Buggele and Decker, 1994] where neural networks where used to interpret shadowgraph images, a type of flow visualization, in order to tune parameters in existing controllers. They concluded that their exercise was too complicated to demonstrate neural-net automation of wind-tunnel operations. Another reference citing the use of predictive control of Mach number at the National Aerospace Laboratory in Amsterdam, The Netherlands, [Soeterboek et al., 1991] demonstrated a 30-60% overall performance improvement over the conventional controller normally used. Their results were based on a *p*-step ahead prediction scheme, using a single operating point model (Mach 0.8), scaled to accommodate small variations in operating point (Mach 0.7 to 0.9).

In [Cooper et al., 1992], a vector quantizing neural classifier is used to identify process error due to both step and oscillating disturbances and adapt a single gain parameter in a simulated continuous stirred tank reactor (CSTR). Their approach demonstrated the ability of such a classifier to distinguish between the resulting error transients associated with these disturbances and adapt the gain of the closed-loop system to reduce the effect of the disturbances.

An overview of manufacturing applications of neural networks [May, 1994], reports positive results of researchers at DuPont Electronics and AT&T Bell Laboratories in plasma etch modeling for semiconductor manufacturing. Arc welding, machining operations, color printing, and linear accelerator beam positioning are given as examples of successful process control applications of neural network based control. "Neural nets are well-suited to process control since they can be used to build predictive models from multivariate sensor data generated by process monitors."

CHAPTER 3
MODELLING THE TUNNEL DYNAMICS

Introduction

In the opening chapter, it was stated that the task of controlling the Mach number in the tunnel was undertaken with a vision to capture the underlying dynamics of a nonautonomous system from observations of time-dependent, input-output data. The motivation for this approach came from previous work by Principe and Wang [1995], using the self-organizing feature map as the infrastructure for local dynamic modeling of chaotic time series. Their work focused on modeling autonomous systems, that is systems where the state trajectory evolves without an external, or exogenous input signal driving the trajectory from one region to another in the state space. That work is adapted here to provide localized predictions of the system response, p steps ahead, to a predetermined set of input or *control* sequences which will drive the system toward the desired region of operation.

The assumption is that the state of the underlying nonautonomous system can be described as a differential equation of the form:

$$\frac{d\,\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), u(t)) \tag{22}$$

where $\mathbf{x}(t)$ are the system states, $u(t)$, the *control signal,* is an *exogenous* input to the system, and f is the vector field that maps a Cartesian product of the state space, S, and

the control space, C, $S \times C \subset \Re^n \; x \; \Re$, to a tangent space $T \subset \Re^n$. If a closed-form solution for (22) exists, that is : Φ: $S \times C \rightarrow S$, then for a given initial condition **x**(0) and *u(t)* specified for all *t*, $\Phi(\mathbf{x}(0), u(t))$, represents a state-space trajectory of the system, or system flow.

For an autonomous system, there is no exogenous *u(t),* and the evolution of the system is assumed to be described by :

$$\frac{d\,\mathbf{x}(t)}{dt} = f(\mathbf{x}(t)) \tag{23}$$

Often, at this point the exogenous input *u(t)* is expressed as a function of the states:

$$u(t) = g(\mathbf{x}(t)) \tag{24}$$

whereby the nonautonomous system becomes autonomous. This is particularly useful for considering the stability characteristics of the system under the influence of a state-dependent, or state-feedback, signal *u(t)* as in (24) above. This will be elaborated upon in the appendix to gain some insight into the stability of the overall system. The approach in this chapter, however, will be to model the system response to a set of candidate control sequences applied as a function of time over a specified interval.

The representation of an arbitrary, discrete non-linear dynamical system using state equations was stated in Chapter 2, repeated here for convenience:

$$\begin{aligned} \Sigma: \; x(k+1) &= f[x(k), u(k)] \\ y(k) &= h[x(k)] \end{aligned} \tag{25}$$

where $\{u(k)\}, \{x(k)\}$, and $\{y(k)\}$ are discrete-time sequences with $x(k) \in \Re^n$, $u(k) \in \Re, y(k) \in \Re$, $f: \Re^n \times \Re \rightarrow \Re^n$, $h: \Re^n \rightarrow \Re$, and $f, h \in C^\infty$. Here f is a

map from the space of system states and input to the space of system states $\Re^n \times \Re \rightarrow \Re^n$, and *h* is a map from the space of system states to the output $\Re^n \rightarrow \Re$.

Our goal here is to determine the system output *y(k)*, over *p* steps into the future, in response to the application of a set of candidate control sequences $\mathbf{U}_{c_i}$ where :

$$\mathbf{U}_{c_i} = [u_{c_i}(k) \;\; u_{c_i}(k+1) \;\; \ldots \;\; u_{c_i}(k+p-1)\,] \tag{26}$$

is the *ith* candidate control sequence, and:

$$\mathbf{M}_{p_i} = [y_{p_i}(k+1) \;\; y_{p_i}(k+2) \;\; \ldots \;\; y_{p_i}(k+p)\,] \tag{27}$$

is the predicted response from the *ith* candidate control sequence.

Review of Local Dynamic Modeling with SOFM

As stated earlier, the previous work by Principe and Wang [1995] provided the starting point for the modeling architecture. Their objective was to construct a neural architecture capable of capturing the underlying dynamics of a chaotic time series. They employed the SOFM as the modeling infrastructure based on the following observations:

1) The SOFM is a localized representation of a signal constructed through competitive learning
2) The converged neural field bears a stronger global resemblance to the input space than other competitive learning, due to the neighborhood function
3) The positioning of each neuron is more strictly constrained by the overall statistical distribution of the signal, which helps to smooth out the irregular spacing of local data samples in the state space.

Their basic idea was to embed the given input space into a compact neural field through the Kohonen SOFM algorithm. Then a simple model estimation process was performed to construct the linearized local models for each response region. The global description of the dynamics was composed of all these local models pieced together. The whole process was composed of two separate procedures: the embedding process of the input space into the neural field followed by the local model estimation.

Their architecture was composed of three layers: input layer **x,** neural field layer *A,* and the layer of local linear models $\tilde{F}(\mathbf{x})$ as shown in Figure 16. The time series was embedded in a state space to create a state vector **x.** The function $\mathbf{i}^*(\mathbf{x})$ was realized by the SOFM. That is to say that the input was fully connected to the nodes of the second layer through a set of weight vectors $\mathbf{w}_i$, where the winner-takes-all neuron was identified by the competition. Each neuron in the neural field layer corresponded to a specific processor $\tilde{F}_i:[\mathbf{a}_i, b_i]$, which represented the linear approximation of the local dynamics.

In this architecture, the SOFM performed two major functions: the positioning of the local models in the state space, and the identification of the matched local model for the current input state **x**. The first function is accomplished during the training phase of the SOFM, while the second is accomplished during the modeling phase. The construction of the overall architecture was composed of three consecutive steps: reconstruction of the state space, mapping the state space in the neural field, and estimation of local linear predictors.

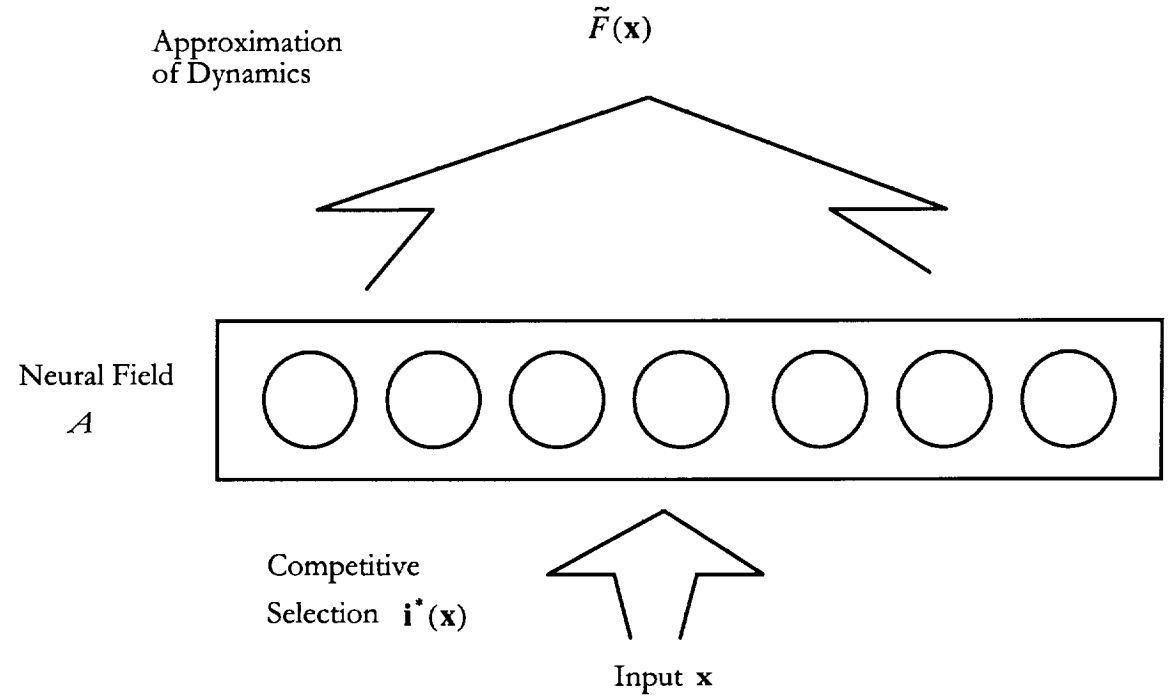


Figure 16. The SOFM-based Modeling Architecture for Time Series

Reconstruction of the state space from the training signal. Following the approach by [Takens, 1980], a sequence of $d+1$ dimensional state vectors $[\mathbf{x}(n)\ x(n+\tau)]$ was created from the given training time series, where $\mathbf{x}(n)=[x(n-(d-1)\tau), x(n-(d-2)\tau),\ldots,x(n)]$ and τ is the appropriate time delay where $d \geq d_A$ and d_A is the dimension of the underlying dynamical process.

Mapping the state space in the neural field. This step was accomplished via the Kohonen learning process. With each vector-scalar pair $[\mathbf{x}(n)\ x(n+\tau)]$ presented as the input to the network, the Kohonen algorithm adaptively discretizes the continuous input space $\mathbf{X} \subset \mathbf{R}^{d+1}$ into a set of disjoint cells $\mathbf{A}$ to construct the mapping $\Phi: \mathbf{X} \rightarrow \mathbf{A}$. This process continues until the learning rate decreases close to zero and the neighborhood function covers one unit. After learning, the neural field representation $\mathbf{A}$ of the input

space $\mathbf{X}$ via the constructed mapping relationship Φ is formed in terms of disjoint units topologically organized in the output space.

Estimation of the locally linear predictors. For each neuron $u_i \in \mathbf{A}$, its local linear predictor in terms of $[\mathbf{a}_i^T, b_i]$ is estimated based on $\alpha_i \subset \mathbf{A}$, which is a set of L_i neurons in the neighborhood of u_i including u_i itself. Each of them has a corresponding weight vector $[\mathbf{w}^T_{i_j}, w_{i_j}(d+1)]^T \in \mathbf{R}^{d+1}$ where $\mathbf{w}_{i_j}^T = [w_{i_j}(1), w_{i_j}(2), \ldots, w_{i_j}(d)]$. The local prediction model $[\mathbf{a}_i^T, b_i]$ is fitted in the least-square sense to the set of weights in α_i:

$$w_{i_j}(d+1) = b + \mathbf{a}^T \mathbf{w}_{i_j} \tag{28}$$

After the above construction procedure, a modeling network is obtained with a global functional map composed of a set of local linear equations

$$x(n+1) = \tilde{F}_i(\mathbf{x}(n)) = \mathbf{a}_i^T \mathbf{x}(n) + b_i \tag{29}$$

where i is the winner-take-all neuron identified by competition in (6).

Modifications for SOFM-based Predictive Control

From (25), consider the output of the nonlinear system Σ :

$$\begin{aligned} y(k) &= h[x(k)] \equiv \Psi_1[x(k)] \\ y(k+1) &= h[f(x(k), u(k))] \equiv \Psi_2[x(k), u(k)] \\ y(k+2) &= h[f(\, f(x(k), u(k)),\, u(k+1)] \equiv \Psi_3[x(k), u(k), u(k+1)] \\ &\cdots \quad \cdots \quad \cdots \\ y(k+n) &= h \circ f^n[.,.] \equiv \Psi_{n+1}[x(k), u(k), u(k+1), \ldots, u(k+n-1)] \end{aligned} \tag{30}$$

where f^n is an n-times iterated composition of f. Denoting the sequence $y(k+1), \ldots, y(k+n)$ by $Y_n(k)$ and the sequence $u(k), u(k+1), \ldots, u(k+n-1)$ as $U_n(k)$, (30) can be expressed as :

$$\Psi[x(k), U_n(k)] = Y_n(k) . \tag{31}$$

For SOFM-based predictive control, the thesis is that a set of feature maps can, collectively, be a global representation of these n-times iterated compositions of *f*, where an SOFM winner represents the localized response of the system to a prototype control sequence, belonging to a larger set of control sequences, the *candidate controls*. Thus, the embedded state space is mapped into a neural field corresponding to a prototype control.

The second major point in the thesis is that predictors that are locally linear in the control can be constructed from the SOFM winners. The construction of the locally linear predictors associated with the SOFM winners is essentially a linearization around the weights of the winner:

$$\mathbf{M}_{\mathbf{p}_i} = \Phi_p[x(k), \mathbf{U}_p] + \nabla\Phi_p[x(k), \|\mathbf{U}_p - \mathbf{U}_c\|_1] \tag{32}$$

where $\|\mathbf{U}_p - \mathbf{U}_c\|_1$ is the $L1$ norm of the difference between the prototype control, $\mathbf{U}_p$ and the candidate control, $\mathbf{U}_c$, and $\nabla\Phi_p$ is the Jacobian with respect to the control, extracted from the converged SOFM weights.

Ideally, perhaps, there would be an individual SOFM, Φ_i, for each candidate control, $\mathbf{U}_{\mathbf{c}_i} = [u_{c_i}(k) \;\; u_{c_i}(k+1) \;\; \ldots \;\; u_{c_i}(k+p-1)]$, and predictions of the tunnel

response, $\mathbf{M}_{\mathbf{p}_i} = [y_{p_i}(k+1) \quad y_{p_i}(k+2) \ \ldots \ y_{p_i}(k+p)\ ]$ would be made using the SOFM winners:

$$\mathbf{M}_{\mathbf{p}_i} = \Phi_i[x(k), \mathbf{U}_{\mathbf{c}_i}]\ . \tag{33}$$

This would not have explored the concept of being able to extract a model that was locally linear in the control from the SOFM and would have required excessive amounts of training data that was not available, i.e. an ensemble of responses for each candidate control over the entire operational range.

Thus the approach to modeling the tunnel dynamics evolved into a procedure consisting of two major components. First, the control input space was manually partitioned by the construction of significant prototype control vectors assumed to be capable of producing the general features of the desired wind tunnel response. Second, for each such partition of the control input space, a SOFM was constructed from an ensemble of tunnel dynamic responses, i.e. the resulting Mach number response, covering the operating range. Each ensemble of Mach number responses was extracted from over 20 hours of actual wind tunnel data, covering the entire operational range. Collectively, the SOFM(s) form an atlas of the global wind tunnel response due to the prototype control inputs.

The assumption here is that having an atlas for the system response to a set of control input prototypes provides a sufficiently complete modeling infrastructure, given the desired objective of predictively controlling the tunnel. There is no need to provide an infrastructure capable of modeling the response to all possible 3^p control sequences of length p, because it is assumed that the control inputs applied to the tunnel, at least in the PMMSC mode of operation, will come from the known set of candidate controls, which

are either the control prototypes themselves, or close enough to the prototypes, by design, so as to predict the tunnel response by local models constructed from the response embedded in the input neural field of the corresponding SOFM.

Partitioning the Control Input Space

The control input space was partitioned by the construction of prototype vectors. Experimentally, it was found that nine prototype vectors were required to achieve the desired control to the specified tolerance. Seven of the control prototypes were 50 sample periods in length, with two shorter prototypes which were 10 samples long. Figure 17 shows the seven 50-point control prototypes. The 10-point prototypes were composed of either all +1's or all -1's.

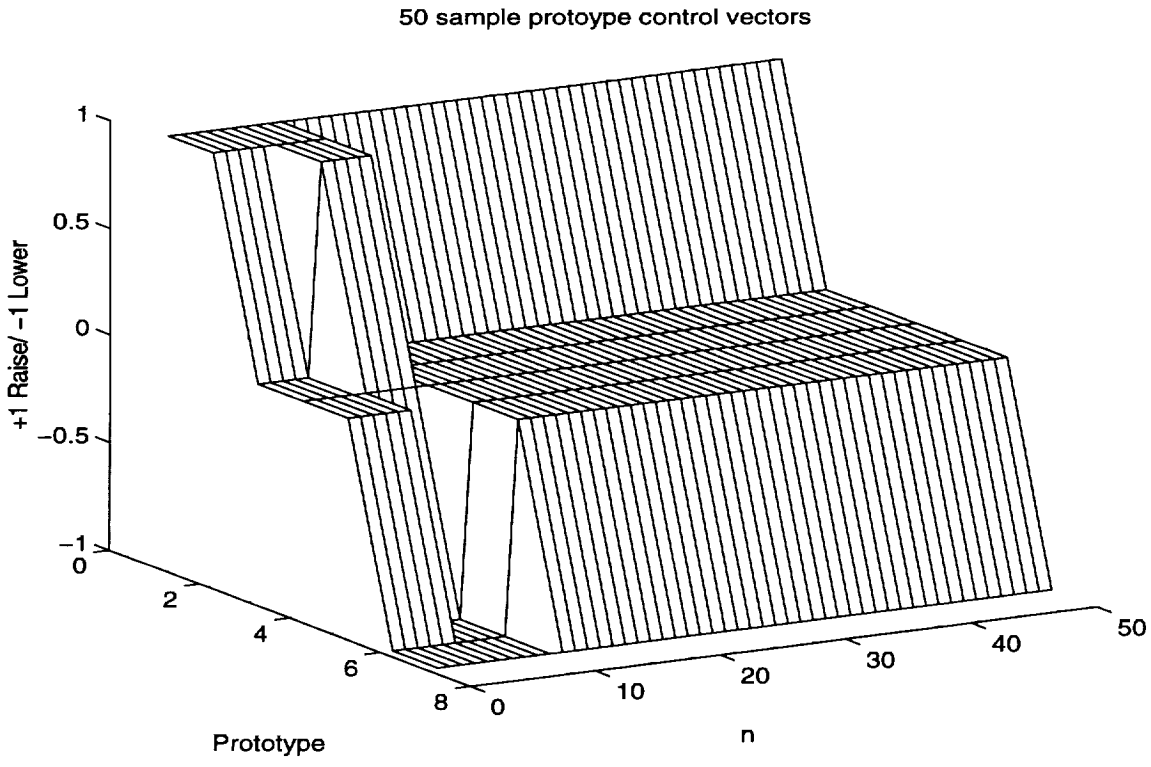


Figure 17. 50-point prototype control inputs

For convenience, the prototypes were assigned labels such as input_class_0, input_class_1, input_class_2, etc. Input_class_0, input_class_1, and input_class_2 are 50-point control sequences consisting of all zeroes, all +1's and all -1's, respectively. Table 3 lists the features of the prototype control vectors.

input_class	n	Composition	Control function
input_class_0	50	Fifty zeroes	Steady-state
input_class_1	50	Fifty +1's (Raise)	Ramp up
input_class_2	50	Fifty -1's (Lower)	Ramp down
input_class_3	50	Ten +1's, forty zeroes	End of Ramp up
input_class_4	50	Ten -1's, forty zeroes	End of Ramp down
input_class_5	50	6-9 zeroes, 1-4 +1's, forty zeroes	Positive correction
input_class_6	50	6-9 zeroes, 1-4 -1's, forty zeroes	Negative correction
input_class_7	10	Ten +1's	Positive transition
input_class_8	10	Ten -1's	Negative transition

Table 3. Prototype Control vectors

The idea here, as discussed in the introduction, was to partition the control input space by manually constructing prototype vectors for the control sequence. The goal of this partitioning was to provide a set of control inputs capable of ramping the tunnel from one operating point to another, regulating about a given operating point, rejecting disturbances, while eliminating control sequences known experimentally to be of no

practical interest, particularly when considering the desire to minimize control activity while regulating about an operating point. An alternating sequence of +1's and -1's might provide the desired regulation of the output, but would be highly undesirable in terms of control effort. This will be elaborated upon in the following chapters.

Ramping the tunnel from one operating point to another would be accomplished with input_classes_1, 2, 3, and 4. Input_classes_5 and 6 would be used for regulating about a given operating point as well as rejecting disturbances. Input_class_0 provides the control input for the ideal steady-state condition with no disturbance, requiring no control action over a 50-point sample interval. Input_classes_7 and 8 provide a transition from the zero-input class to the ramping inputs of input_classes_1 and 2, and provide identification of the tunnel response over a shorter, more recent interval of time.

Clustering the Mach Number Responses

For each of the control input classes, ensembles of Mach number responses resulting from the application of each control prototype were extracted from the wind tunnel test data. Next, each ensemble of responses was clustered using a SOFM. The SOFM imposes a topographic ordering of the output neural field corresponding to features of the input patterns, which are in this case, the Mach number responses, taken over the past *n* sample intervals. Collectively, the SOFM(s) were trained using data extracted from more than 20 hours of actual wind tunnel response data. Table 4 lists the number of exemplars for each class.

Input_classes_3 and _4 have the fewest number of exemplars because they only occur at the end of the transition from one set point to another. Input_classes_1, _2, _7, and _8 have the greatest numbers of exemplars due to the relatively long transition times from one operating point to another, requiring steady ramping up or down. Next in frequency of application is input_class_0, representing the most desirable, minimum control effort over the 50 sample interval (15 seconds) when the Mach number is within the desired tolerance. The remaining two input_classes, _5 & _6, represent prototype positive and negative corrections which provide disturbance rejection and regulation about a set point, with the desired features of the control sequence, i.e. minimum control effort and minimum number of switchings or transitions from one state to another.

Input class	# exemplars
0	10,158
1	15,332
2	13,464
3	41
4	31
5	155
6	198
7	17,393
8	16,694

Table 4. Training exemplars for each input class

The following figures (19 through 27) show ensembles of Mach number responses from the application each control prototype, and their corresponding SOFM. The Mach number response, **M**, is taken over the same n sample intervals as the application of the control prototype,

$$\mathbf{M} = \mathrm{M} - M(t-n); \tag{34}$$

$$\mathrm{M} = M(t), M(t-1),..., M(t-n); \tag{35}$$

and n is either 50 or 10. Thus, **M** represents the output of an *n-tap* delay line, where the value at the nth tap is subtracted from all the values in the delay line. The output at a single tap is shown in Figure 18. This is essentially a bank of comb filters which preprocesses the Mach number responses, particularly for removing the dc component, yielding the change in Mach number over the past n samples. Both the training samples and the on-line Mach number responses were preprocessed in this fashion.

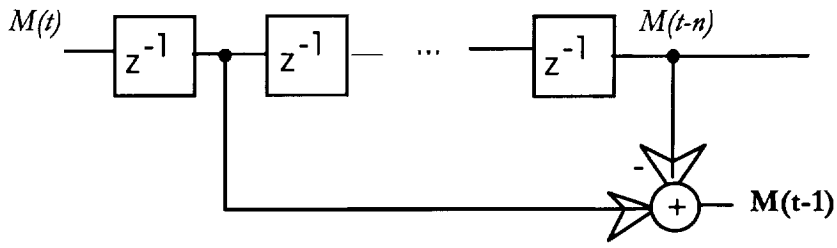


Figure 18. A single tap of the Mach number preprocessor

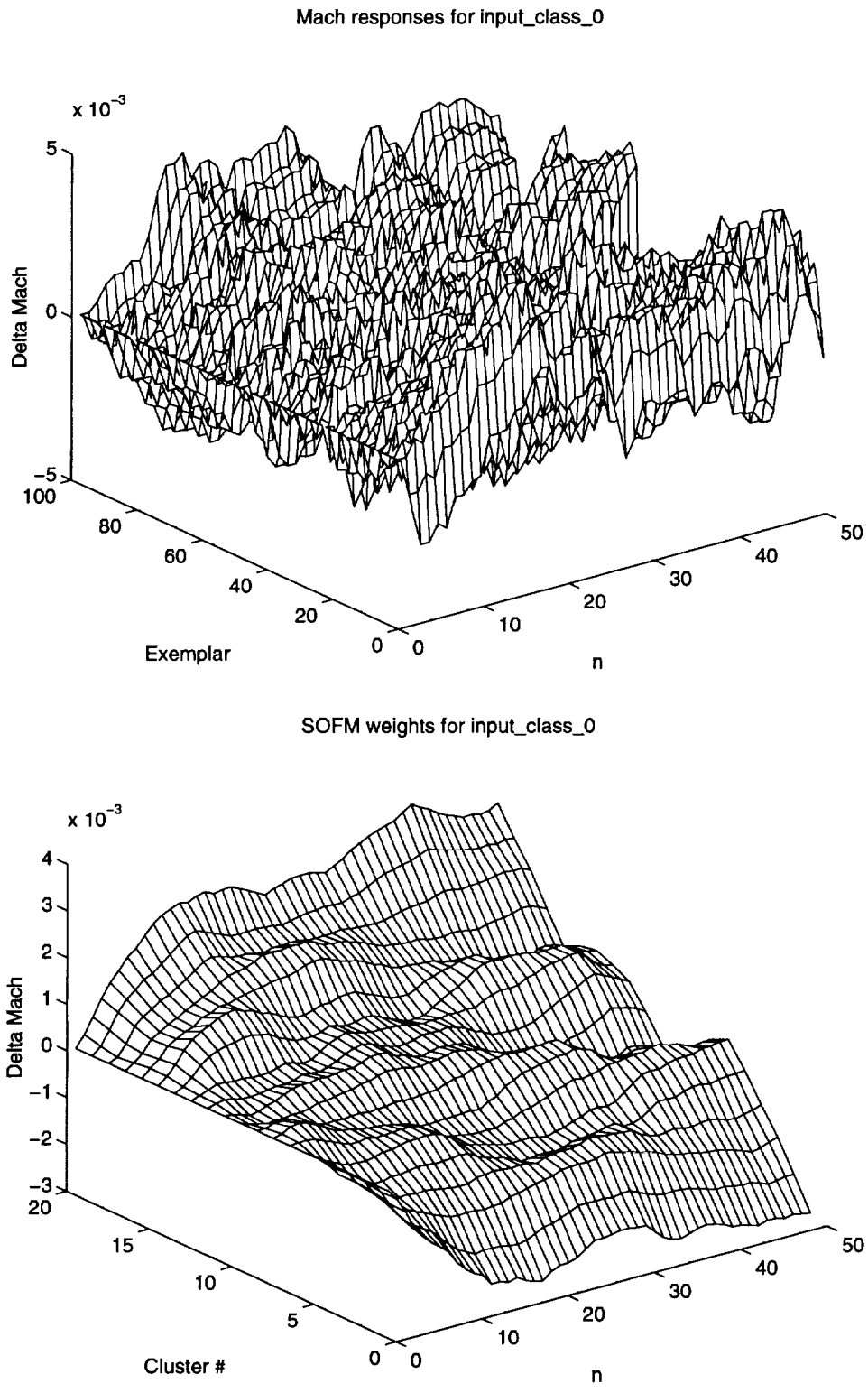


Figure 19. Mach number responses and corresponding SOFM for input_class_0

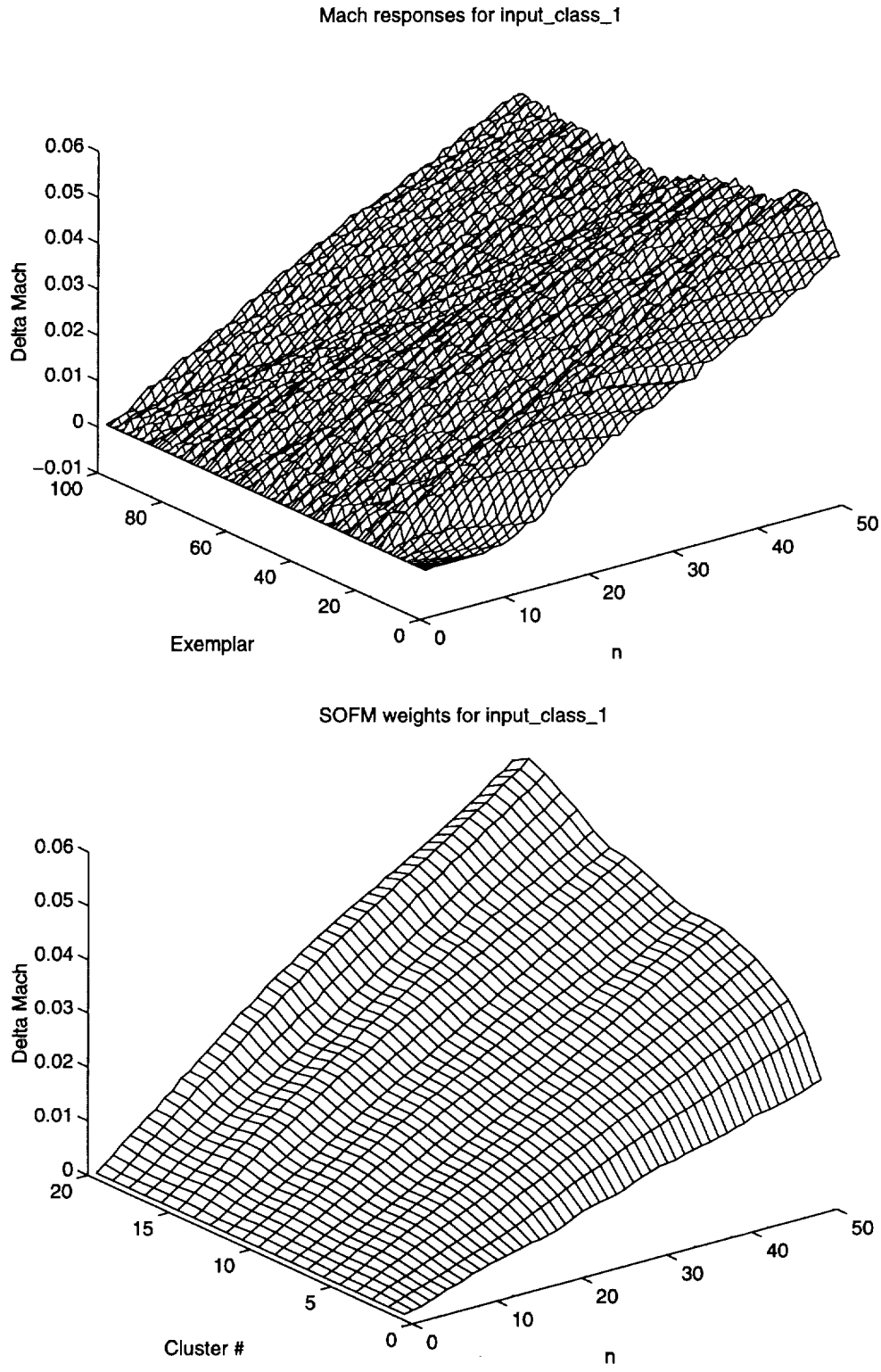


Figure 20. Mach number responses and corresponding SOFM for input_class_1

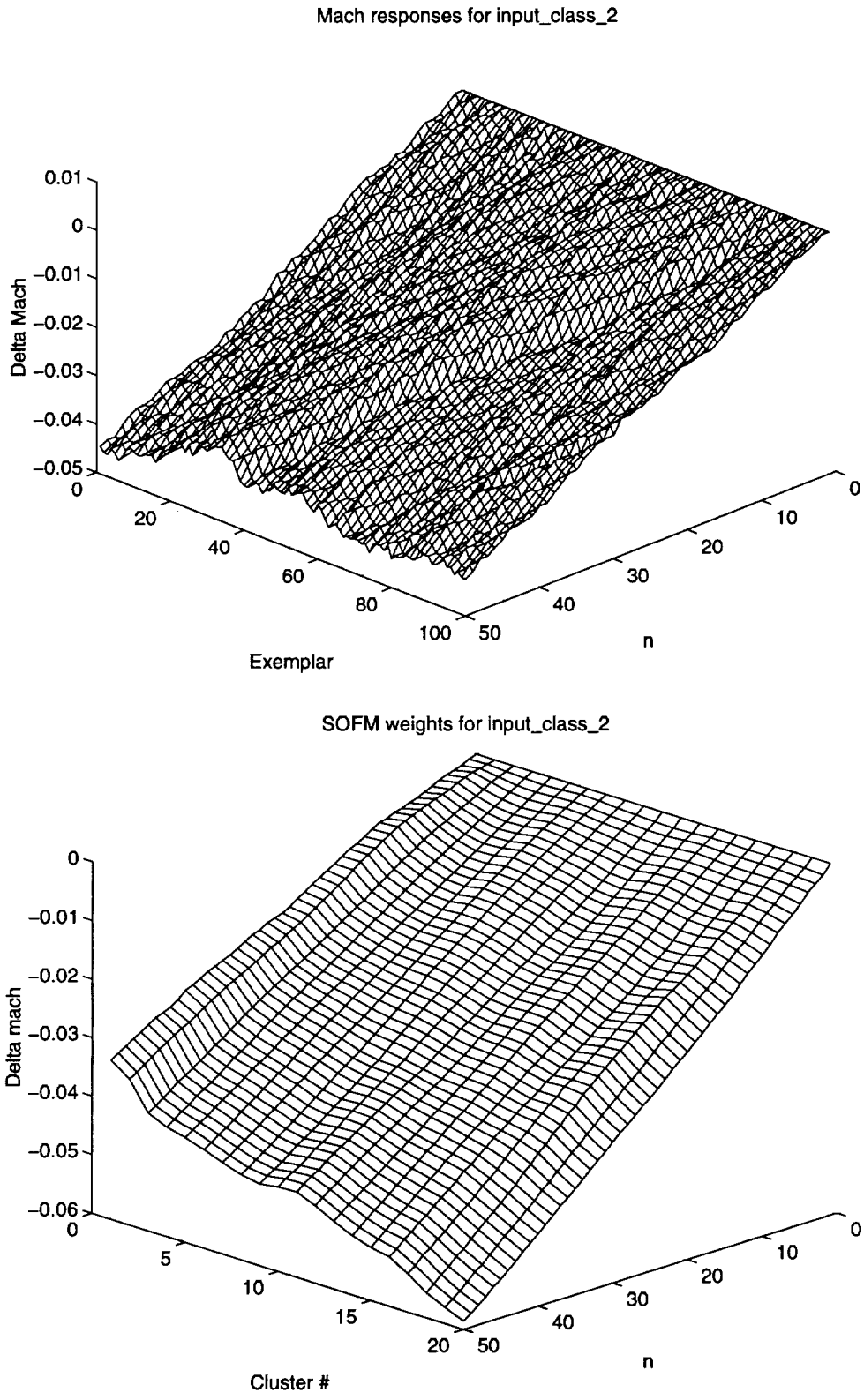


Figure 21. Mach number responses and corresponding SOFM for input_class_2

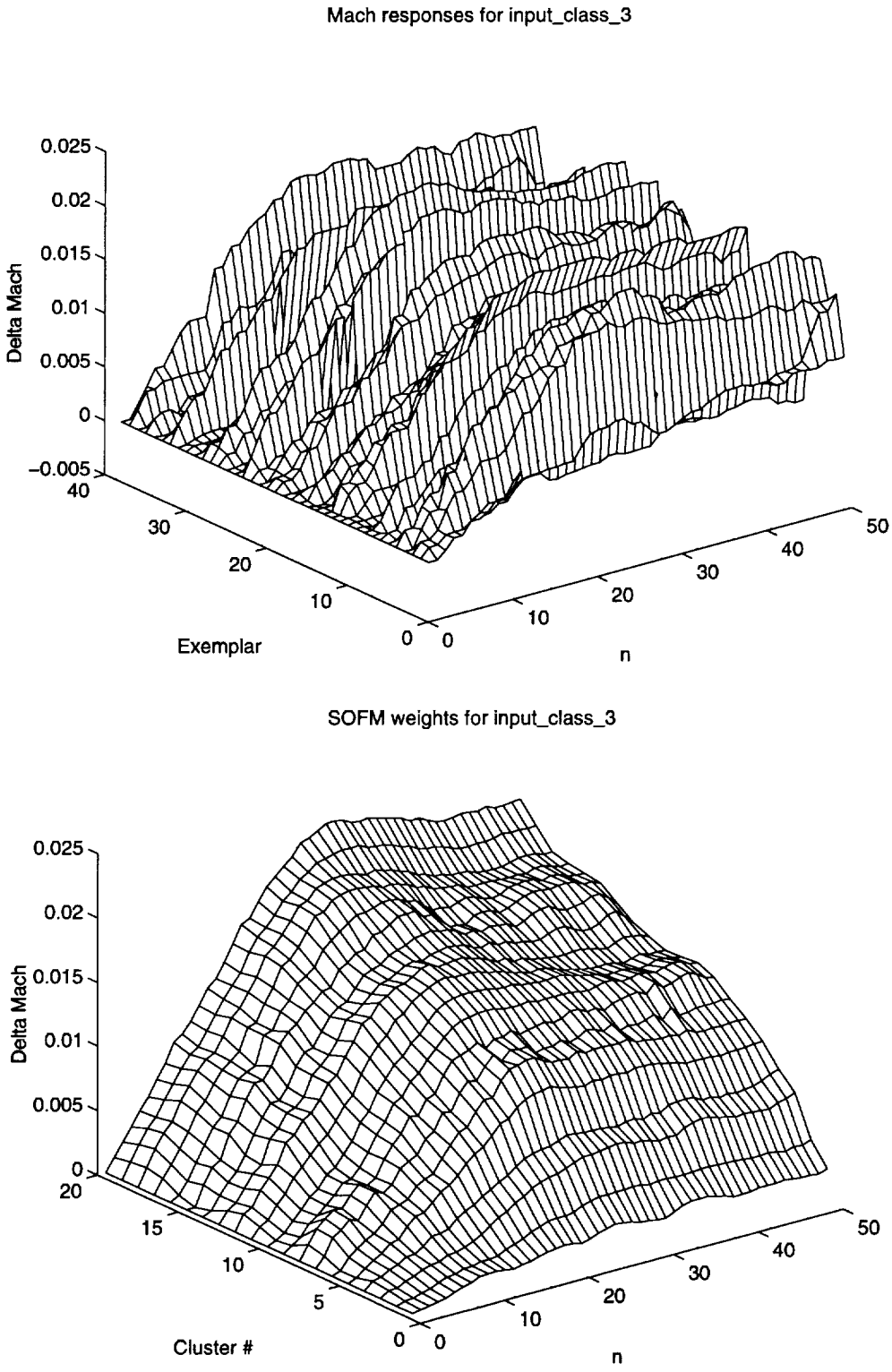


Figure 22. Mach number responses and corresponding SOFM for input_class_3

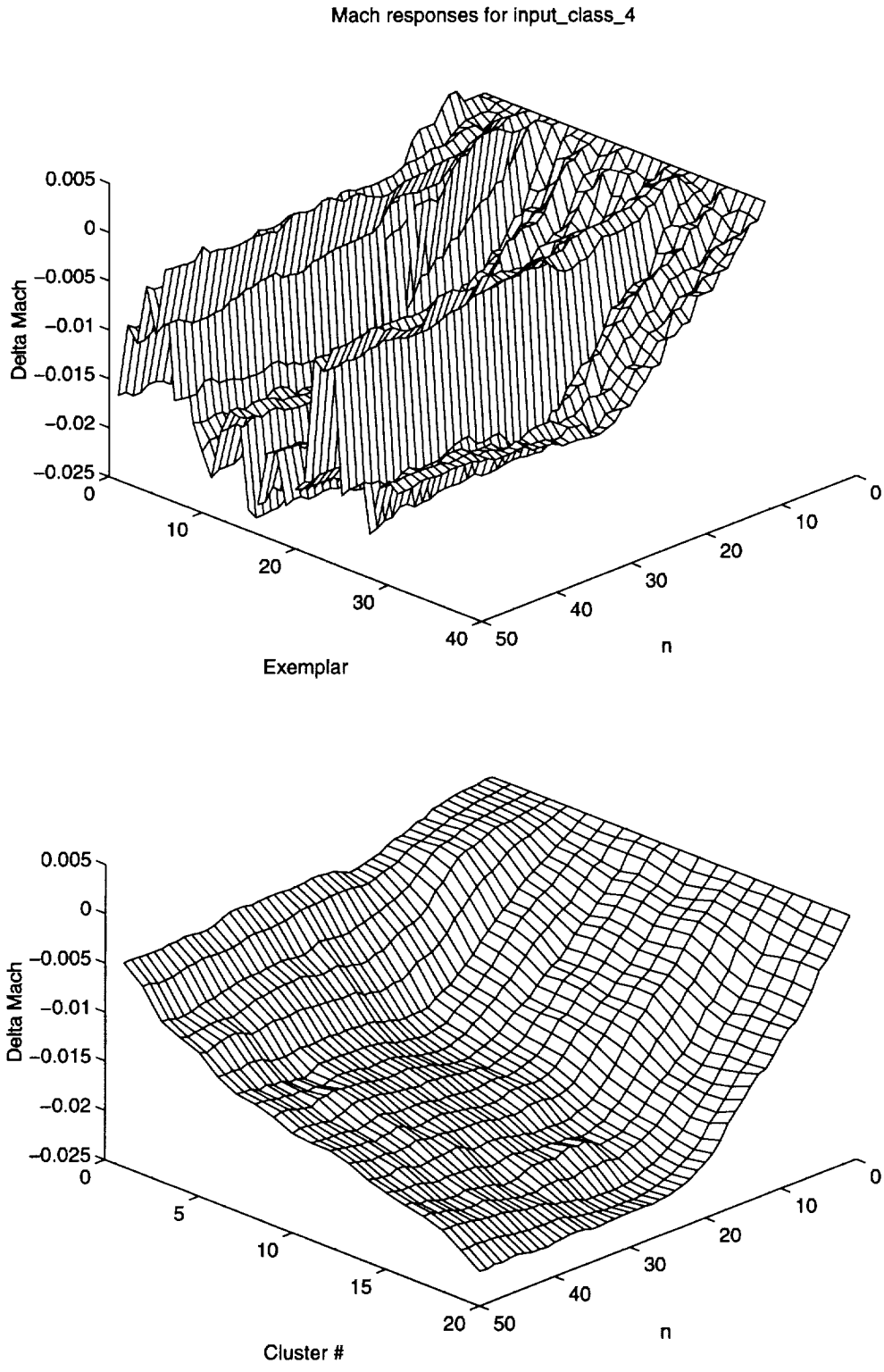


Figure 23. Mach number responses and corresponding SOFM for input_class_4

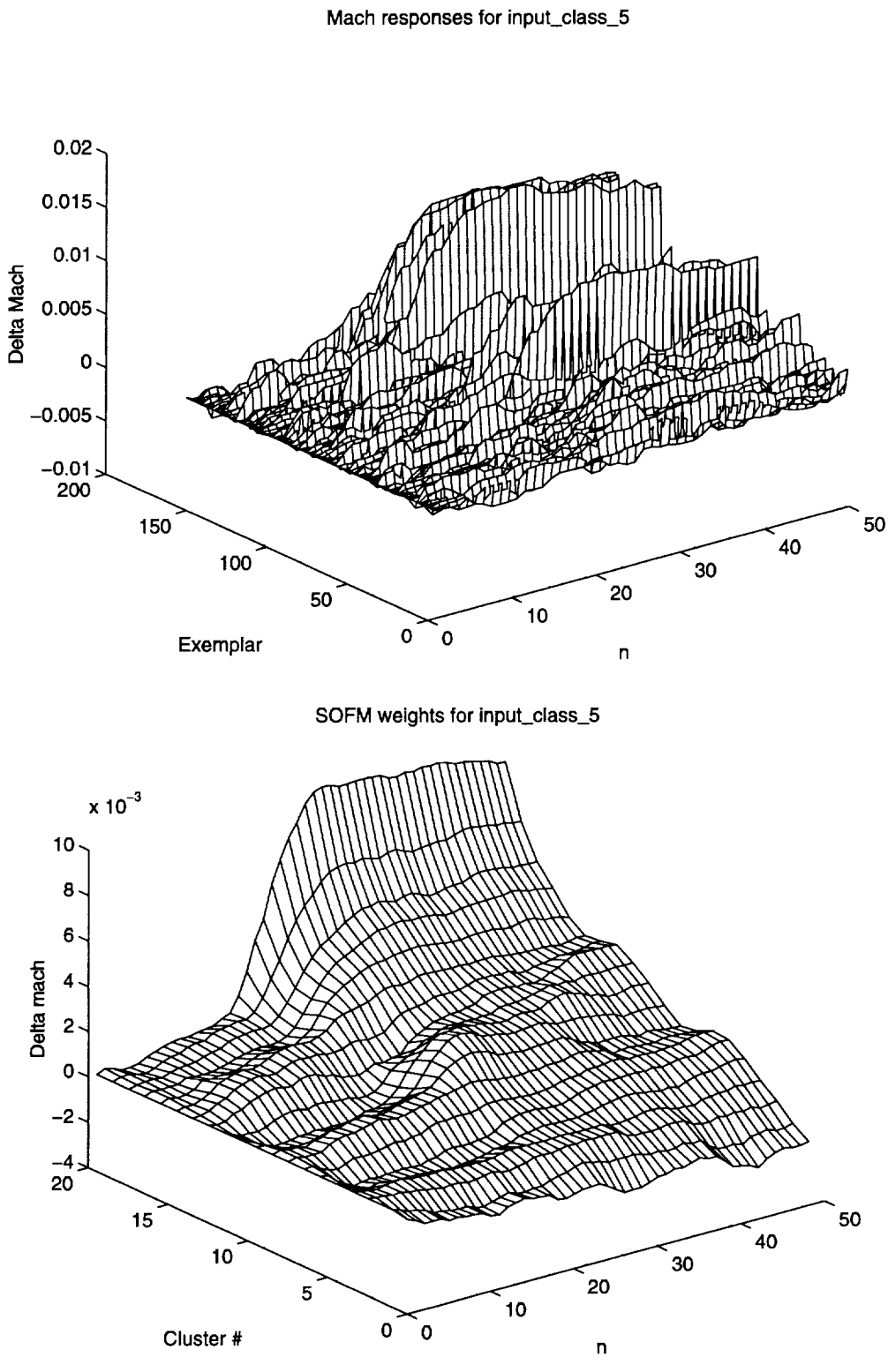


Figure 24. Mach number responses and corresponding SOFM for input_class_5

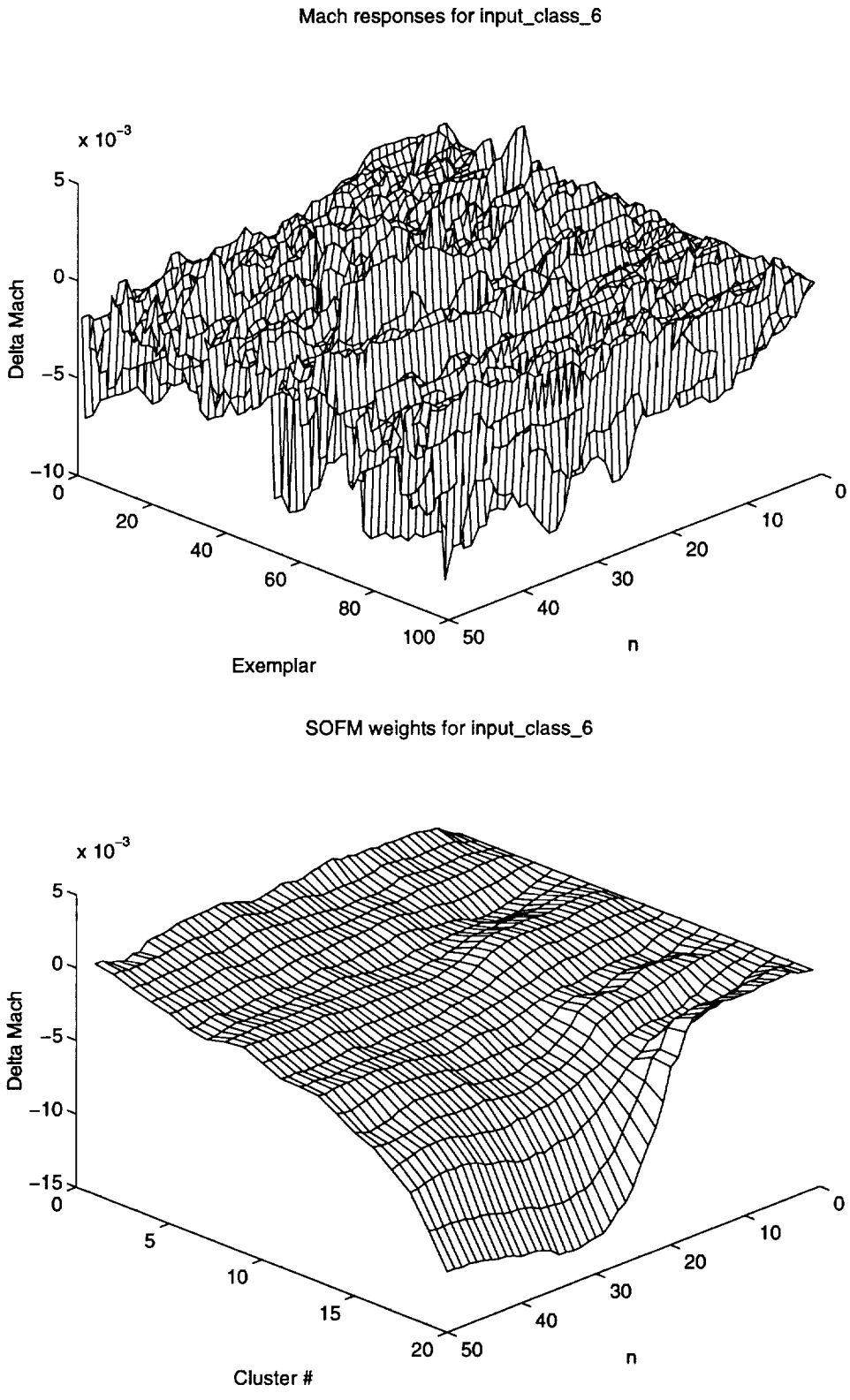


Figure 25. Mach number responses and corresponding SOFM for input_class_6

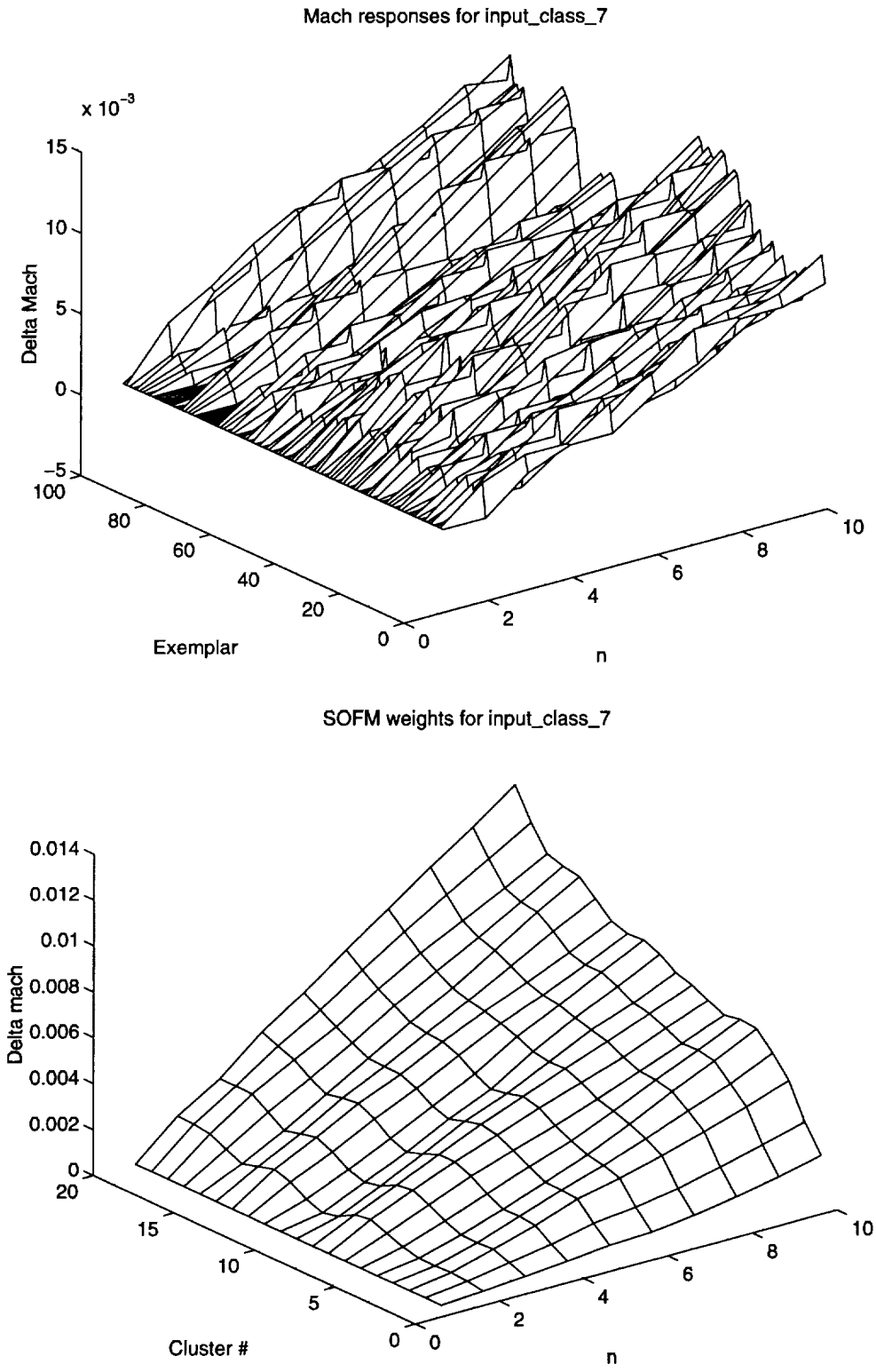


Figure 26. Mach number responses and corresponding SOFM for input_class_7

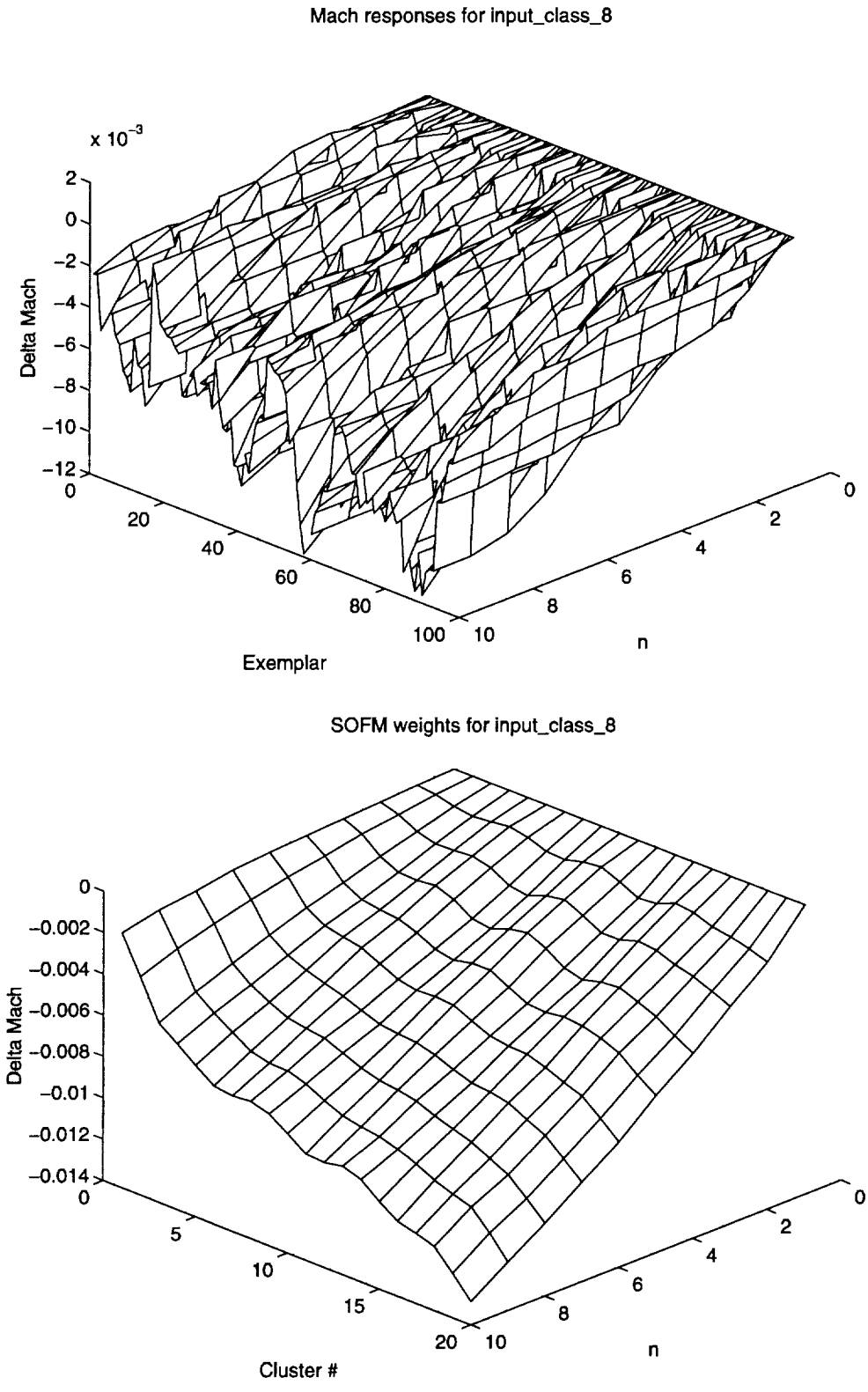


Figure 27. Mach number responses and corresponding SOFM for input_class_8

Convergence of the Input Neural Fields

The number of nodes for the SOFMs, each representing a cluster of the Mach number responses, was adjusted during the training phase to achieve an average separation between the adjacent converged neural input fields, or more simply, the weights of the SOFM. The topographic ordering imposed by the SOFM was key in this phase of the development. The number of nodes were adjusted so that the separation between the adjacent input neural fields corresponded to the desired goal of controlling the Mach number, based on 50 samples-ahead predictions, to better than the required 0.003 tolerance. Thus, the major focus was to determine the number of classes for the SOFMs for input_class_5 & _6, which provide the basis for regulation and disturbance rejection. Each of these SOFMs were trained with 155 and 198 exemplars. It was found experimentally that 20 nodes or clusters provided adequate separation based on considering the separation between the adjacent means of each neural field over the last 30 point interval:

$$\frac{1}{30}\left[\sum_{i=21}^{50} M_k^*(i, j+1) \; - \sum_{i=21}^{50} M_k^*(i, j) \right]. \tag{36}$$

Nodes were added to the SOFM until the mean separation, taken over the entire map, was well below 0.001 for input_classes_5 & _6, as listed in Table 5. The resulting 20 node SOFM structure was implemented for all the input_classes, and the resulting separations between adjacent input neural fields were considered adequate. The mean separation for input_class_0 was even less than the above classes, and the mean separations for the ramping input_classes SOFMs were deemed sufficient for the relatively coarser control required to move from one set point to another.

	$(10^3)\ \frac{1}{30}\left[\sum_{i=21}^{50} M_k^*(i,j+1) - \sum_{i=21}^{50} M_k^*(i,j)\right]$							$(10^3)\ \frac{1}{10}\sum_{i=1}^{i=10}(\bullet)$	
j	*k = 0*	*k = 1*	*k = 2*	*k = 3*	*k = 4*	*k = 5*	*k = 6*	*k = 7*	*k = 8*
1	0.6342	5.284	-1.352	1.648	-1.407	0.5741	-0.2934	0.6155	-0.7840
2	0.6194	3.111	-3.625	2.639	-2.557	0.6314	-0.4436	0.7627	-1.007
3	0.2973	1.686	-1.270	1.032	-1.249	0.2768	-0.4233	0.2835	-0.4472
4	0.1652	1.377	-0.8680	2.655	-1.547	0.3332	-0.3458	0.2418	-0.2451
5	0.2856	0.8588	-0.8589	1.438	-2.220	0.3767	-0.4339	0.2112	-0.1888
6	0.2339	0.9396	-0.7762	0.5972	-1.116	0.2678	-0.2206	0.1906	-0.2295
7	0.0565	0.6088	-0.7927	1.073	-0.4461	0.2143	0.0058	0.3037	-0.2699
8	0.0129	0.3392	-0.5479	0.8365	-0.2759	0.3067	-0.0531	0.1421	-0.1933
9	0.0166	0.8414	-0.1769	0.3579	-0.5014	0.4656	-0.4198	-0.089	-0.0416
10	0.1377	0.8744	-0.3810	0.1971	-0.5801	0.2794	-0.4187	0.0292	-0.1017
11	0.3549	0.8427	-0.8745	0.4618	-0.4409	0.0169	-0.1808	0.2533	-0.2718
12	0.3034	1.368	-0.6687	0.5598	-0.5765	0.0771	-0.2167	0.2644	-0.2841
13	0.1219	1.009	-0.5736	0.6931	-0.9616	0.1947	-0.5848	0.2335	-0.1611
14	0.0339	0.9442	-0.9577	0.6108	-0.7398	0.4424	-0.6323	0.1458	-0.0474
15	0.1218	0.6524	-0.7976	0.4443	-0.4548	0.6975	-0.5178	0.1386	-0.2038
16	0.4175	1.130	-0.7018	0.4792	-0.6299	0.7110	-0.7711	0.2827	-0.3107
17	0.3741	2.039	-1.055	0.3583	-0.8048	0.5912	-1.417	0.3475	-0.2851
18	0.4796	1.665	-2.076	1.221	-0.6797	1.554	-2.468	0.4746	-0.6025
19	0.5925	-0.8114	-1.757	1.390	-0.4480	2.735	-2.253	0.5246	-0.6308
m	0.2768	1.303	-1.058	0.9839	-0.9282	0.5656	-0.6363	0.2819	-0.3319
SD	0.2039	1.234	0.7670	0.7153	0.6259	0.6221	0.6809	0.2003	0.2550

Table 5. Difference between interval means of adjacent input neural fields

	$(10^3)\left[\frac{1}{30}\sum_{i=21}^{50} M_k^*(i,j)\right]$							$\left[\frac{1}{10}\sum_{i=1}^{10} M_k^*(i,j)\right]$ * 1000	
j	*k = 0*	*k = 1*	*k = 2*	*k = 3*	*k = 4*	*k = 5*	*k = 6*	*k = 7*	*k = 8*
1	-2.556	15.98	-22.28	2.456	-3.719	-1.570	1.031	0.7942	-0.6336
2	-1.922	21.26	-23.63	4.105	-5.126	-0.9963	0.7372	1.409	-1.417
3	-1.303	24.38	-27.25	6.745	-7.683	-0.3649	0.2936	2.172	-2.425
4	-1.006	26.06	-28.52	7.777	-8.933	-0.0881	-0.1297	2.456	-2.872
5	-0.8406	27.44	-29.39	10.43	-10.48	0.2452	-0.4755	2.698	-3.117
6	-0.5549	28.30	-30.25	11.87	-12.67	0.6219	-0.9095	2.910	-3.331
7	-0.3210	29.24	-31.03	12.47	-13.82	0.8897	-1.130	3.010	-3.536
8	-0.2645	29.85	-31.82	13.54	-14.26	1.104	-1.124	3.403	-3.806
9	-0.2519	30.19	-32.37	14.37	-14.54	1.411	-1.177	3.545	-4.000
10	-0.2349	31.03	-32.54	14.73	-15.04	1.876	-1.597	3.456	-4.041
11	-0.0973	31.90	-32.93	14.93	-15.62	2.156	-2.016	3.485	-4.142
12	0.2576	32.74	-33.78	15.39	-16.06	2.173	-2.197	3.738	-4.414
13	0.5610	34.11	-34.47	15.95	-16.64	2.250	-2.414	4.002	-4.698
14	0.6830	35.12	-35.04	16.64	-17.59	2.444	-2.998	4.236	-4.859
15	0.7169	36.01	-35.60	17.25	-18.33	2.887	-3.631	4.382	-4.906
16	0.8387	36.72	-36.78	17.70	-18.79	3.584	-4.149	4.520	-5.111
17	1.256	37.85	-37.50	18.18	-19.42	4.295	-4.919	4.803	-5.421
18	1.631	39.89	-38.55	18.54	-20.23	4.886	-6.337	5.151	-5.706
19	2.110	41.55	-40.63	19.76	-20.91	6.441	-8.806	5.625	-6.309
20	2.703	40.74	-42.39	21.15	-21.35	9.176	-11.06	6.150	-6.939

Table 6. Interval means of SOFM input fields

	$(10^3) \left\| M_k^*(i,j) \right\|$								
j	*k = 0*	*k = 1*	*k = 2*	*k = 3*	*k = 4*	*k = 5*	*k = 6*	*k = 7*	*k = 8*
1	16.78	92.73	129.3	14.83	22.78	9.669	8.075	3.195	2.594
2	12.90	123.3	136.9	24.13	30.09	6.997	4.947	5.456	5.658
3	8.921	141.6	158.5	39.41	44.14	2.808	2.318	8.344	9.363
4	6.794	151.9	166.3	46.01	51.61	1.048	1.322	9.629	10.92
5	5.625	160.1	171.6	62.03	61.78	3.099	3.247	10.74	12.02
6	4.136	165.2	176.3	70.51	76.33	4.611	6.642	11.26	12.96
7	2.839	170.6	180.4	74.66	82.47	5.177	7.177	11.58	13.58
8	2.623	174.0	185.1	81.11	85.71	6.713	6.448	12.51	14.13
9	2.933	176.0	188.6	85.83	88.76	8.406	6.576	13.17	14.82
10	2.632	180.9	189.5	88.40	91.33	10.55	8.931	13.21	15.38
11	1.902	185.9	191.9	90.99	94.44	12.09	11.24	13.65	16.03
12	2.070	190.6	197.3	94.72	97.32	12.26	12.45	14.40	16.64
13	3.504	198.4	201.0	96.88	100.6	13.35	14.24	14.92	17.24
14	4.774	204.4	204.2	100.3	106.1	14.36	17.37	15.58	17.88
15	5.547	210.0	209.8	104.7	111.2	16.69	20.30	16.41	18.44
16	6.890	213.9	214.6	107.5	115.1	20.09	23.23	17.22	19.27
17	8.854	220.4	219.2	110.9	119.0	23.92	28.27	17.93	20.19
18	10.63	231.9	225.3	113.3	123.4	27.28	36.12	18.92	21.10
19	13.57	241.6	236.9	119.0	126.7	35.86	49.03	20.78	23.18
20	17.47	236.9	247.1	126.3	128.2	50.95	61.49	22.89	25.40

Table 7. Euclidean norm of SOFM input neural fields

In order to quantify the topological ordering of the converged neural fields, Table 6 lists the mean taken over each 30 or 10 sample interval of the SOFM weights. Table 7 enumerates the Euclidean norm for all the converged neural fields as well. With the exception of SOFM_0, the norms steadily increase (or decrease) along the output field of the map. SOFM_0 displays increasing distance from the center of the map, outward, corresponding to the symmetry of the interval means about the center of the map shown in Table 6.

SOFM Selection for Local Model Identification

After the application of a candidate control, one of the nine SOFM is used to cluster the Mach number response, **M**, over the past n sample intervals. The selection of the SOFM is based on the minimum Euclidean norm between the control input history $\mathbf{U} = u(t-1),\ u(t-2),\ \ldots, u(t-m)$ and the set of prototype control vectors $\mathbf{U_i};\ i=1,n$:

$$input\_class\_i = \min_i \left\| \mathbf{U} - \mathbf{U_i} \right\|. \tag{37}$$

If more than one prototype control vector matches identically, i.e. $\left\| \mathbf{U} - \mathbf{U_i} \right\| = 0$ for more than one i, both SOFM(s) are excited with the appropriate length **M**. This can occur for SOFM_1 (or _2) and SOFM_7 (or_8), where the SOFM_7 (or 8) winner represents the response over the 10 most recent samples, while the SOFM_1 (or _2) winner represents the response over the past 50 samples. Additionally, the regulating control classes were clustered on a region of the control space, defined in Table 3, as opposed to a single point in the control space.

The SOFM metric for the winner is the minimum Euclidean norm between $\mathbf{M}$ and the SOFM prototype vectors, $\mathbf{M_i}$, for the SOFM selected by input_class :

$$mach\_class\_i = \min_{i=1:20} \left\| \mathbf{M} - \mathbf{M_i} \right\|. \tag{38}$$

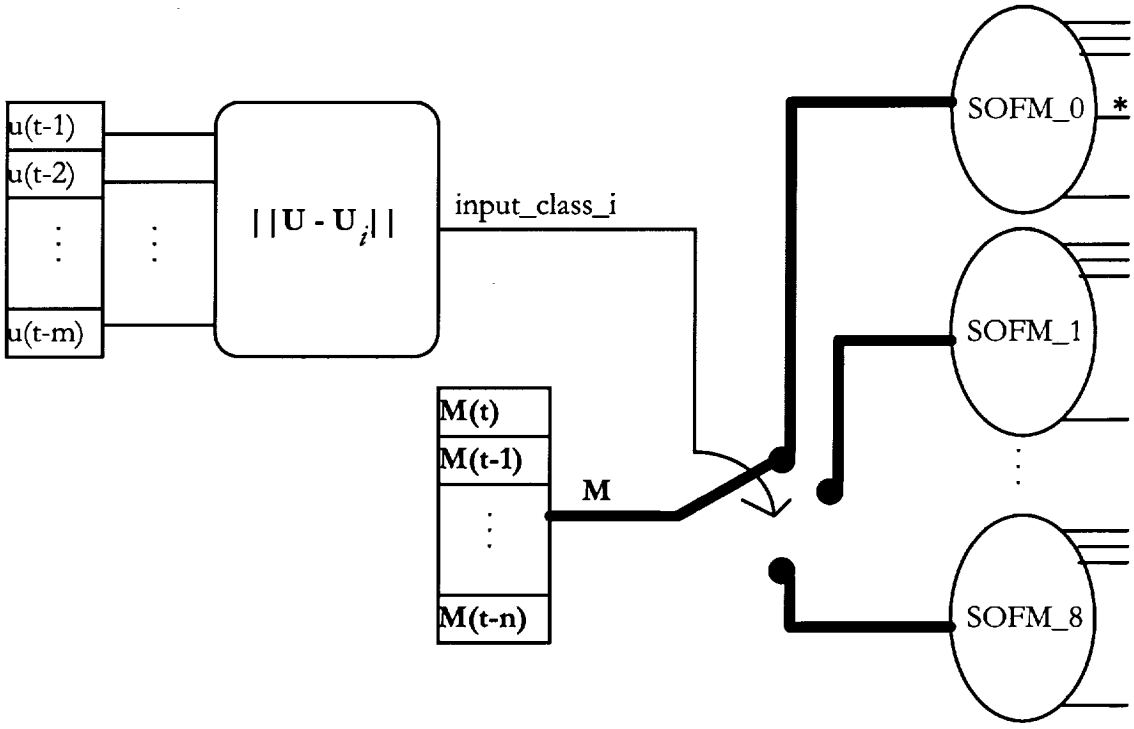


Figure 28. Selection of SOFM by input_class

Prediction of Tunnel Response Using Local Models

The Mach number responses are predicted by a linear model:

$$\mathbf{M}_{\mathbf{P}_c} = a_c \mathbf{A}^* + \mathbf{W}^*; \tag{39}$$

where $\mathbf{W}^*$ is the prototype response vector, or weights, of the winning node.

$\mathbf{A}^*$ is the least square approximation of the winner's prototype response to a d-sample delayed unit step sequence, $\mathbf{U}_r = [\underbrace{0,0,\ldots,0}_{d},\underbrace{1,1,\ldots 1}_{p-d}]$, where d represents the maximum relative degree or delay from input to output and p is the total number of samples ahead for which the prediction is made:

$$\mathbf{A}^* = b\mathbf{U}_r \tag{40}$$

where b is fit in the least square sense, or, alternately,

$$b = \mathbf{W}^*(\mathbf{U}_r)^{\Diamond} \tag{41}$$

and $^{\Diamond}$ denotes the Moore-Penrose pseudoinverse of the vector $\mathbf{U}_r$.

By inspection of the SOFM for all input classes, d was chosen conservatively to be greater than any observed delay, $d = 20$. A single constant, a_c, scales $\mathbf{A}^*$ based on the ratio of the $L1$ norm of the candidate control vector $\mathbf{U}_c$ and the $L1$ norm of the control sequence $\mathbf{U}$, producing the response **M:**

$$\begin{aligned} a_c &= \left\{ \frac{\|\mathbf{U}_c\|_1}{\|\mathbf{U}\|_1} - 1 \right\}, \ \|\mathbf{U}\|_1 \neq 0 \\ a_c &= \quad 0 \quad , \ \|\mathbf{U}\|_1 = 0 \end{aligned} \tag{42}$$

Thus, $a_c\mathbf{A}^*$ provides the difference in the predicted Mach number response due to the to the distance between the control sequence, $\mathbf{U}$, and the i^{th} candidate control sequence, $\mathbf{U}_c$. For the simplest case, $\mathbf{U}_c = \mathbf{U}$, the value of a is zero, and the Mach number response is predicted directly from the input neural field. This linear model is driven by the candidate control inputs, shown in Figure 29.

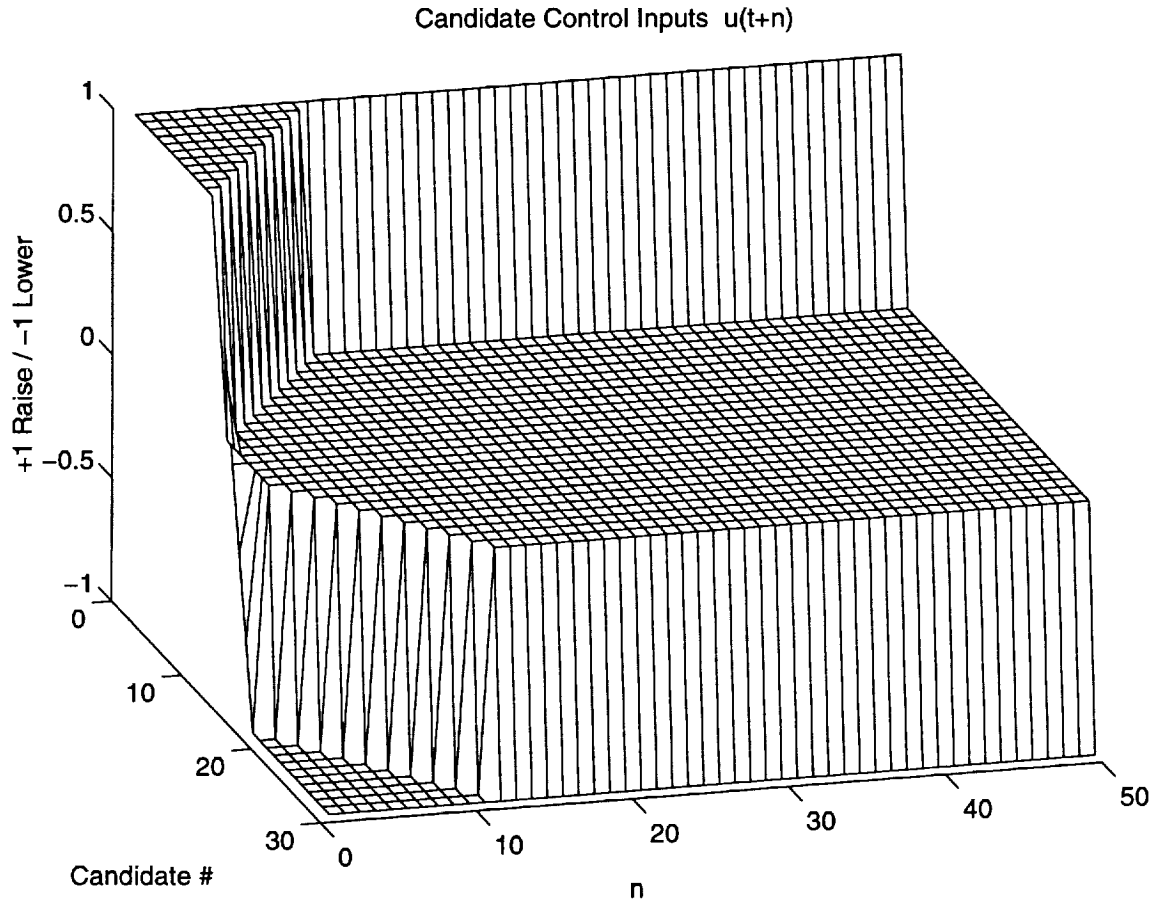


Figure 29. Candidate Control Sequences

Comparisons of the predictions of the Mach number to the actual tunnel responses as a result of the application of the candidate controls will be presented in Chapter 5, Experimental Results.

CHAPTER 4
PREDICTIVE CONTROLLER

Introduction

Given the model of the tunnel response developed in the previous chapter, the predictive controller evaluates the relative effectiveness of the candidate control inputs. The advantage of partitioning the control input space using a set of prototype controls becomes more apparent when compared to model-based predictive control (MPC) [Clarke, Mohtadi, and Tuffs, 1987]. In our method, predicted responses from a set of candidate control inputs can be extracted either directly from the SOFM's output neural field or from the derived local model. The controller then applies the control sequence which minimizes the error between the desired output and the predicted output over some finite number of steps into the future. The low computational cost of multi-step prediction by this method allows prediction for relatively long (50 samples ahead) control sequences, or *control horizon,* in the terminology of MPC, using relatively simple computing hardware. This is in contrast to MPC, which requires the inversion of an *NU x NU* matrix at each step for a control horizon *NU* steps into the future. A brief background of MPC is provided as a basis for comparison to SOFM-based predictive control using control prototypes.

Model Predictive Control Background

Most input-output model based predictive control schemes [Clarke, Mothadi, and Tufts, 1987], begin with the assumption of a linear model (ARMA, or Autoregressive-Moving Average) :

$$y(k) = \sum_{i=1}^{n} a_i y(k-i) + \sum_{j=1}^{m} b_{j-1} u(k-j) \tag{43}$$

with an additional disturbance term in moving average form :

$$d(k) = \xi(k) + \sum_{i=1}^{nc} c_i \xi(k-i) \tag{44}$$

where $\xi(k)$ is an uncorrelated random sequence.

Combining (43) and (44) and introducing the polynomials *A*, *B*, and *C* in the backward shift operator q^{-1}:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na}$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + \ldots + b_{nb} q^{-nb}$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + \ldots + c_{nc} q^{-nc}$$

yields

$$A(q^{-1}) y(k) = B(q^{-1}) u(k-1) + C(q^{-1}) \xi(k) \tag{45}$$

which is referred to in the literature as the CARMA (Controlled Auto-Regressive and Moving Average) model, a variation on the ARMAX (Auto-Regressive Moving Average with exogenous input) model.

A further refinement to the disturbance model to accommodate non-stationary disturbances such as random steps occurring at random times is :

$$d(k) = C(q^{-1})\xi(k) / \Delta \tag{46}$$

where $\Delta = 1 - q^{-1}$, the differencing operator. Combining (45) and (46) yields the CARIMA (Controlled Auto-Regressive Integrated Moving Average) model used in Generalized Predictive Control (GPC) :

$$A(q^{-1})y(k) = B(q^{-1})u(k-1) + C(q^{-1})\xi(k) / \Delta . \tag{47}$$

At this point it is useful to introduce a scalar cost function *J*:

$$J = \sum_{i=N_1}^{N_2} [\hat{y}(k+i) - w(k+i)]^2 + \sum_{j=1}^{Nu} \lambda(j)[u(k+j-1)]^2 \tag{48}$$

where :

$\hat{y}$ is the predicted response from the control input sequence u

N_1 is the beginning of the costing horizon;

N_2 is the end of the costing horizon;

N_u is the control horizon;

$\lambda(j)$ is a control-weighting sequence.

N_1, N_2, N_u, and $\lambda(j)$ represent tuning knobs which can be adjusted by the control designer to tailor the control action for the desired response characteristics. Rules of thumb provide some guidelines for initial selection. N_1 is usually picked to be greater than the largest anticipated time delay between the input *u(k)* and its response in the output *y(k)*. N_2 is determined by the longest settling time associated with the pulse or step response of the model. $N_u = 1$ is quite often chosen for open-loop stable non-minimum phase plants, but this often represents a compromise between the computational burden associated with longer control horizons.

The minimization of J, given a future set point sequence **w**, where :

$$\mathbf{w} = [w(t+1), w(t+2), \ldots, w(t+N)]' \tag{49}$$

leads to the control law :

$$\mathbf{u} = [\mathbf{G}^{\mathrm{T}}\mathbf{G} + \lambda \mathbf{I}]^{-1}\mathbf{G}^{\mathrm{T}}(\mathbf{w} - \mathbf{f}) \ . \tag{50}$$

The matrix **G** is of dimension $N \times NU$:

$$\mathbf{G} = \begin{bmatrix} g_o & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ . & .. & \cdots & . \\ \vdots & . & & g_0 \\ \vdots & . & & \vdots \\ g_{N-1} & g_{N-2} & \cdots & g_{N-NU} \end{bmatrix} . \tag{51}$$

This requires the inversion of an $NU \times NU$ matrix at each sample time, or at least for each identified change in the g parameters, which are the coefficients of the z-transform of the plant's step response. **f** is a linear combination of values of $u(t)$ and $y(t)$ up to time t.

SOFM-based Predictive Controller

The function of the SOFM-based predictive controller is to evaluate the relative effectiveness of the candidate control inputs in reducing the error between the desired Mach number and Mach number predicted by the current SOFM winners. This is done by evaluating the Euclidean norm of the difference between the last 30 points of the 50-points-ahead predicted Mach number responses and the desired Mach number set point :

$$err\_norm\_i = \left\| \mathbf{M}_{P_i}[21{:}50] - \mathbf{M}_{SP} \right\| \tag{52}$$

for all i candidate control sequences, as specified in the prediction section. The evaluation over the last thirty points of the prediction is to emphasize steady-state matching. The control sequence associated with the minimum norm of all i sequences is then applied as the control to the tunnel.

This is similar to the scalar cost function for GPC (48) :

$$J_p = \left[\sum_{i=N_1}^{N_2} [\hat{y}_p(k+i) - w(k+i)]^2 \right]^{1/2} \tag{53}$$

with $N_1 = 21$, $N_2 = 50$ and $\hat{y}_p$ is the predicted Mach response for the *pth* candidate control sequence. Both the constraints on the permissible values of the control (+1, 0, and -1) as well as the minimization of the control cost is embedded in the set of all p candidate control sequences with control horizon $N_u = 50$. The control $\mathbf{U}_p$ that generates $\hat{y}_p$ is selected for the minimum J_p.

In the set of candidates we included controls to ramp the set point up and down for large changes in operating point, as well as the regulating control sequences for disturbance rejection. The candidate control sequences, their associated SOFMs for prediction, and their control update parameters are listed in Table 8. The control update parameter for each candidate control determines whether the entire 50 sample control sequence is applied as the control, or just the first point in the sequence. Implicitly, this selection is done based on the error between the Mach number set point and the predicted responses. If the selected candidate control corresponds to either the two largest control efforts over the control horizon, or if the selected candidate control represents the

minimum control effort (i.e. zero) over the control horizon, the control sequence is updated by selection of the prediction-error minimizing control at the next sample period. For all other cases, the entire 50 sample selected candidate is applied. The two cases of regulating about an operating point and operating point changes illustrate the differences.

Candidate #	[u(t+1), u(t+2), ... u(t+50)]	SOFM	k
1	[(50) +1's]	1	1
2	[(11) +1's (39) zeros]	3	1
3	[(10) +1's (40) zeros]	3	50
4	[(9) +1's (41) zeros]	5	50
5	[(8) +1's (42) zeros]	5	50
6	[(7) +1's (43) zeros]	5	50
7	[+1 +1 +1 +1 +1 +1 (44)zeros]	5	50
8	[+1 +1 +1 +1 +1 (45) zeros]	5	50
9	[+1 +1 +1 +1 (46) zeros]	5	50
10	[+1 +1 +1 (47) zeros]	5	50
11	[+1 +1 (48) zeros]	5	50
12	[+1 (49) zeros)]	5	50
13	[0.66 (49) zeros]	5	50
14	[0.33 (49) zeros]	5	50
15	[50 zeros]	0	1
16	[-0.33 (49) zeros]	6	50
17	[-0.66 (49) zeros]	6	50

18	[-1 (49) zeros]	6	50
19	[-1 -1 (48) zeros]	6	50
20	[-1 -1 -1 (47) zeros]	6	50
21	[-1 -1 -1 -1 (46) zeros]	6	50
22	[-1 -1 -1 -1 -1 (45) zeros]	6	50
23	[-1 -1 -1 -1 -1 -1 (44) zeros]	6	50
24	[(7) -1's (43) zeros]	6	50
25	[(8) -1's (42) zeros]	6	50
26	[(9) -1's (41) zeros]	6	50
27	[(10) -1's (40) zeros]	4	50
28	[(11) -1's (39) zeros]	4	1
29	[(50) -1's]	2	1

Table 8. Candidate Control sequences and associated parameters

Operating Point Changes

The typical set point change is greater in magnitude than 0.1, which is several times greater than the largest Mach number change associated with any of the SOFM input fields. Set point changes of this magnitude produce the selection of either candidate control #1 (50 +1's) or #29 (50 -1's). These control sequences are updated at each sample interval, which means that the controller decides at each sampling instant whether to extend the series of raise or lower commands to achieve the desired set point. If the only selection was between the continued ramping associated with either SOFM_1 (ramp

up) or SOFM_2 (ramp down), and the next prototype control associated with SOFM_3 (end of ramp up) and SOFM_4 (end of ramp down), the transition between set points would indeed be rather coarse. The inclusion of candidates # 2 and #28 provide a one control-tick resolution between continued ramping and the transition to regulating about the desired set point. Ramping continues on until candidates #3 is selected over #2 or #27 is selected over #28 as the prediction-error minimizing control. These control sequences (#3 or #27) are applied for their entire 50-point duration, allowing for a smooth transition to regulation about the set point.

Regulating About an Operating Point

When actively regulating about an operating point, the entire 50 sample control sequence selected from the set of candidates, consisting of an active or non-zero segment of 1/3 to 10 sample periods, followed by the corresponding number of zeroes during the inactive segment, is applied as the control input for the next 50 sample periods. Thus, the selected candidate control is applied open-loop over the entire 50 sample control horizon, with the next control update occurring 50 sample periods later. The resulting 50-sample Mach number response is then input to the corresponding SOFM, and future predictions are made from the output neural field of the SOFM winner as described in Chapter 3.

If the sequence of all zeroes is selected, the control is updated at the next sample period. The 50 sample Mach number response is input to SOFM_0 for identification of the local dynamics by the SOFM_0 winner. Prediction and control sequence selection is performed at each sample period until an active (non-zero) control sequence is selected to regulate the Mach number.

CHAPTER 5
EXPERIMENTAL RESULTS

In Chapter 3 and 4, the SOFM-based modeling of the tunnel dynamics and the resulting predictive controller were developed. The control input space was manually partitioned by the use of prototype control sequences and SOFM's were trained to cluster the corresponding Mach number responses. Thus, the output neural field of each SOFM represents a collection of local models of the tunnel dynamics for the corresponding prototype control input. During the experiment, while actually controlling the tunnel with the PMMSC, the control inputs were chosen from the set of candidate control sequences, making the task of identifying the local dynamic model more straightforward than in the more general case of all allowable 3^p control sequences of length p.

Thus, the experimental results are composed of two parts. The first part is to look at the results of controlling the Mach number during actual experimental tests conducted in January 1996. These results will be compared to control of the tunnel with an existing gain-scheduled automatic controller as well as control by an expert human operator. The second part is to explicitly examine the results of modeling the tunnel dynamics with the control-input partitioned SOFM architecture. This will be accomplished by comparing the Mach number response predicted by the SOFM-derived local model to the actual response after application of the error-minimizing control sequence determined by the predictive controller.

Experimental Setup

The experimental setup consisted of a 486-33 MHz PC connected via a serial port to the existing control computer at the wind tunnel, referred to as the "tunnel micro". The tunnel micro is an early 1980's vintage 8086-based microcomputer. The existing automatic control implemented in the tunnel micro is a highly tuned but fixed table look-up of drive motor commands based on the error at a given Mach number [Capone et al., 1995]. The tunnel micro also communicates with the wind tunnel data acquisition system. The data acquisition system provides the Mach number measurements at a nominal sample interval of 0.3 seconds. Figure 30 shows a block diagram of the experimental setup.

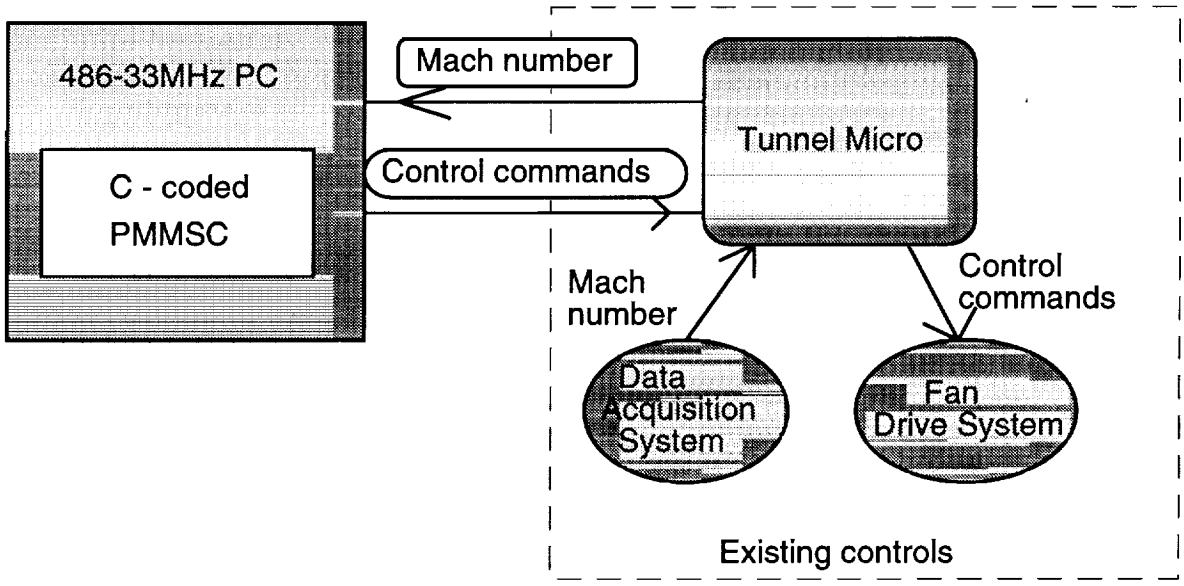


Figure 30. Experimental Setup

The PMMSC was implemented as a C program, compiled and run on the PC. The output of the program, at each sample interval, is a control command which is

communicated to the tunnel micro and then applied to the drive system for the tunnel fans. The control command can take on the values of; +1 to raise the tunnel fan RPM, -1 to lower the fan RPM, or zero to maintain fan RPM. Further, the command duration may be specified to be either the full sample interval, 0.3 seconds, or less than the full sample interval in 0.1 second increments, (i.e. either 0.1 or 0.2 second duration). This subdivision of the sample interval was required to provide finer control of the tunnel Fan RPM. Control inputs of less than 0.1 second duration are generally ineffective in producing a change in the tunnel fan RPM. Additionally, the PC was used to record the time histories of the tunnel state, control inputs, and PMMSC internal variables such as the predicted response and SOFM winning nodes.

Mach Number Measurements

The Mach number is computed from the a calibrated ratio of stagnation pressure to static pressure measured in the plenum surrounding the test section, as described earlier in Chapter 1, equation (1).

The most recent calibration of the wind tunnel was performed in 1990 [Capone, et. al, 1995]. This calibration used 30 static pressure measurements taken along the nominal 8-ft calibrated test section length (CSTL). Flow uniformity was parameterized by both the standard and maximum deviation of spatially local Mach number from a least-squares straight-line fit. The results of this calibration are listed in Table 9. In this table, the test section Mach number M_r, is the value of a least-squares straight-line fit to the Mach number data, corresponding to the midpoint of the test section. The standard

deviation is a measure of the average discrepancy along the test section length. The maximum deviation represents the worst departure from the least-squares fit along the selected length of test section. The document reporting the results of the calibration lists 2σ values, i.e. twice the positive square root of the variance.

M_r	2σ	σ_{max}
0.3015	0.000560	0.001088
0.4006	0.000754	0.001688
0.5014	0.000943	0.002158
0.6018	0.001152	0.002381
0.6544	0.001291	0.002833
0.7030	0.001388	0.003085
0.7537	0.001415	0.003350
0.7795	0.001478	0.003413
0.8000	0.001422	0.003015
0.8284	0.001481	0.003552
0.8555	0.001514	0.003632
0.8809	0.001576	0.003756
0.9038	0.001428	0.003700
0.9304	0.001584	0.003749
0.9579	0.001539	0.003684
0.9816	0.001422	0.003211

Table 9. Standard and maximum deviation of Mach number during calibration

From this table, it can be seen that both the standard and maximum deviation of Mach number measured along the CTSL vary significantly over the subsonic range. This spatial variation corresponds to the temporal variation of steady-state Mach number measurements. This is illustrated by taking the standard deviation of a time series of Mach number measurements calculated from the calibrated ratio of stagnation pressure to plenum static pressure under steady conditions during operational tests. In Table 10, M is the mean value of 200 consecutive Mach number measurements taken under steady

conditions with no control input applied. The standard deviation is the sample standard deviation:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(M(i)-\overline{M}\right)^2 ; \sigma = \sqrt{\sigma^2} ; \tag{54}$$

where 2σ is used for direct comparison to the calibration results.

M	2σ
0.2979	0.000546
0.3968	0.000520
0.5999	0.000908
0.8014	0.001820
0.8518	0.001497
0.8850	0.001977
0.9003	0.001577
0.9504	0.001822
0.9819	0.002652

Table 10. Statistics of time histories of steady state Mach number measurements

Experimental Results of Controlling the Mach Number

Experimental results were obtained while controlling the wind tunnel with the PMMSC at several subsonic Mach numbers. These tests were conducted during the period of January 10th through January 23rd, 1996.

Figure 31 shows the wind tunnel Mach number being controlled by the PMMSC for a period of three hours, during a normal operational tunnel run, where aerodynamic research data was being taken. Mach number set points of 0.95, 0.9, 0.85, and 0.6 are

shown as dashed lines. The PMMSC regulated the steady-state Mach number to within the research requirement of 0.003 of the set point during the interval shown. PMMSC commands are shown with magnitudes less than one for control commands whose duration was less than the 0.3 second sampling period. Control pulses of 0.1 second are shown with magnitude 0.33 and 0.2 second pulses are shown with magnitude 0.66.

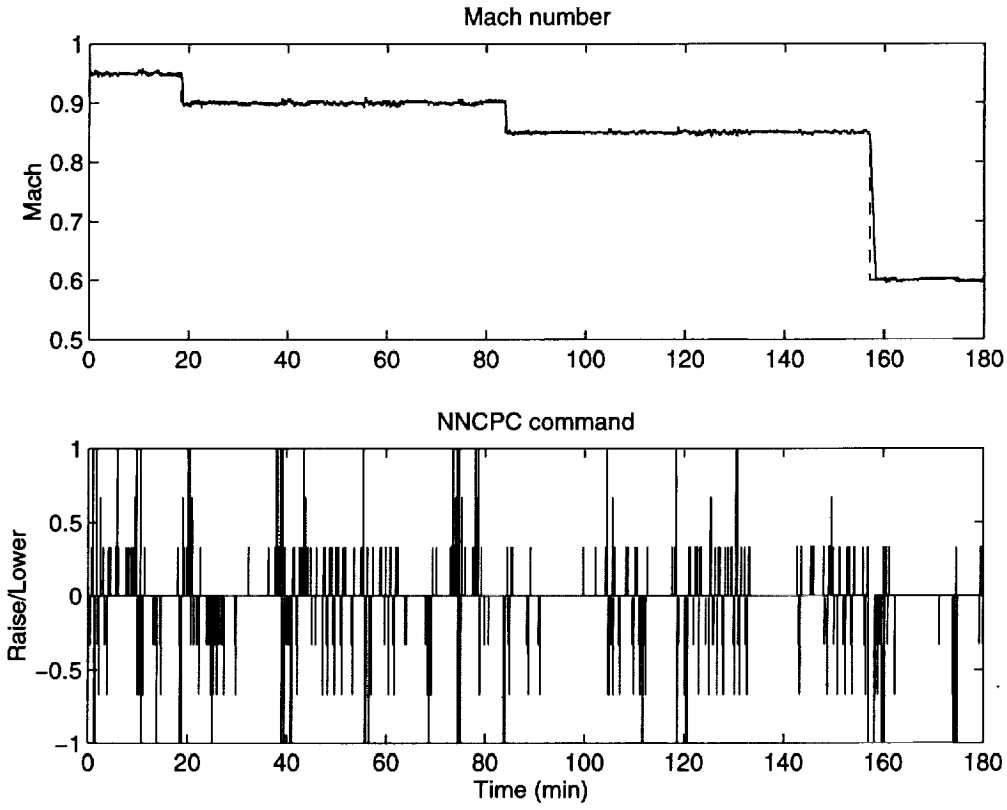


Figure 31. Mach number controlled by PMMSC* during a three hour test

*The PMMSC was previously referred to as NNCPC, so this acronym appears in some of the plots.

During these tests, the aircraft model attitude was varied to achieve the desired aerodynamic research data. Figure 32 shows typical variations of model attitude at each Mach number.

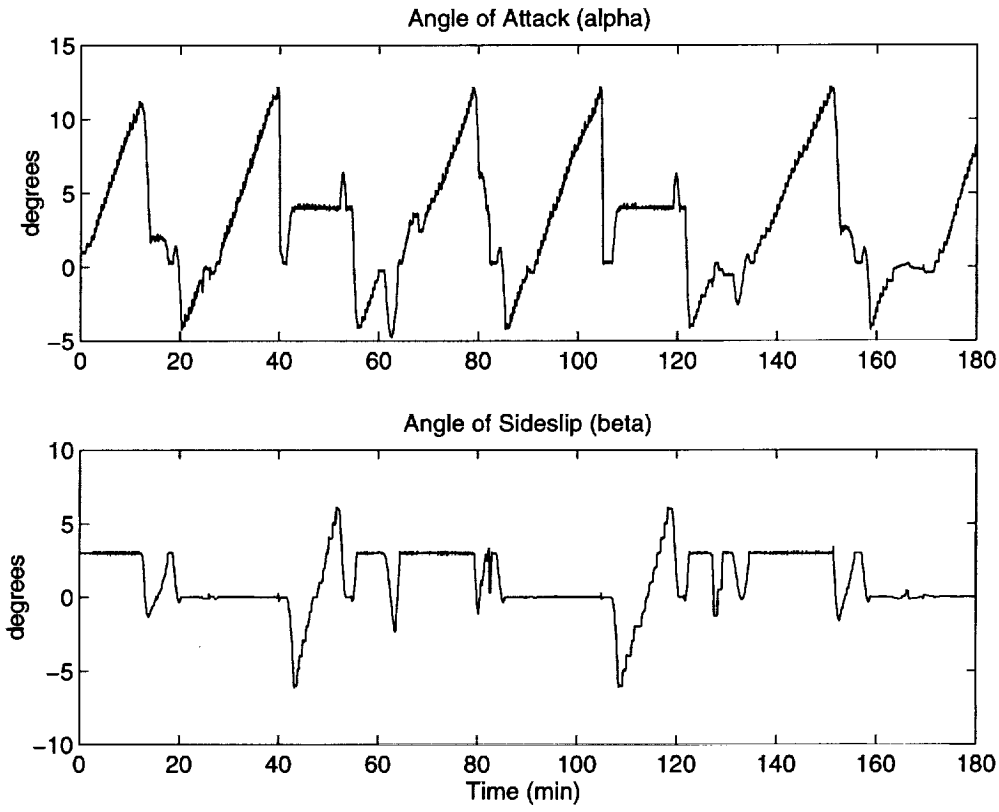


Figure 32. Variations of angle-of-attack and angle-of-sideslip during test

The variations are of two general types, referred to as an "alpha sweep" or "beta sweep". During an alpha sweep, the model angle-of-attack, or "alpha", is stepped through some range, from -4 degrees to +12 degrees in this test, while maintaining a constant angle-of-sideslip, or "beta". During a beta sweep, the model angle-of-sideslip is stepped through some range, from -6 to +6 degrees for this test, while maintaining a constant alpha. Mach number must be within 0.003 of the desired Mach number to satisfy the research requirements. The variation in model attitude produced some modest

(< 0.001/degree) disturbance in the tunnel Mach number, particularly at angles-of-attack above five degrees, although the onset of this disturbance was dependent on the test Mach number. Figure 33 shows the SOFM winning nodes for positive and negative corrections as determined by the PMMSC during the run. Figure 34 shows the Fan RPM and tunnel temperature during the run.

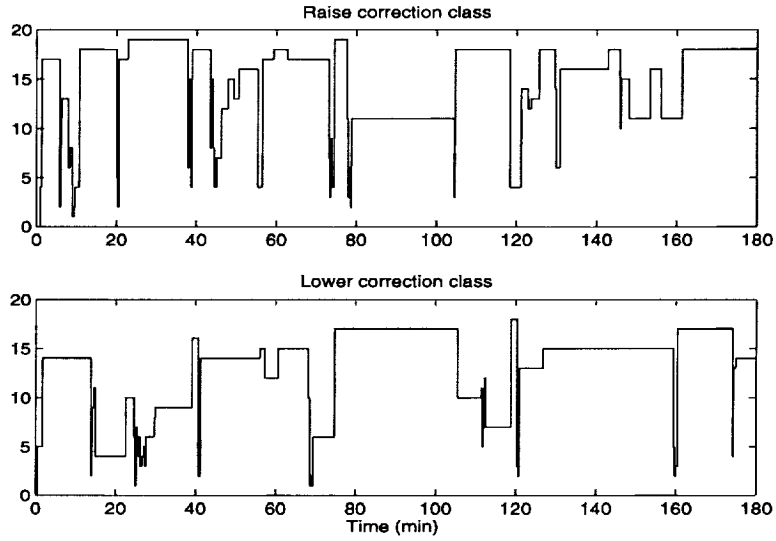


Figure 33. Winning nodes for SOFM_5 and SOFM_6 during test

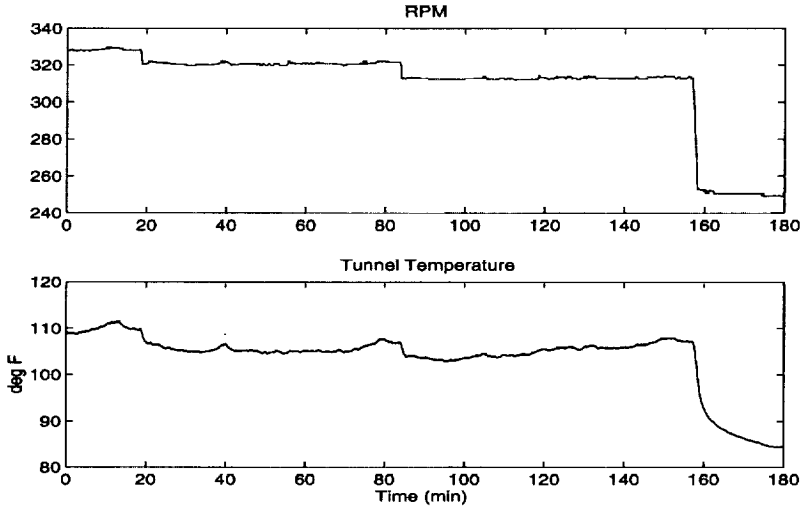


Figure 34. Fan RPM and Tunnel temperature during test

In order to illustrate the operation of the PMMSC, a shorter interval of the run is shown in Figure 35. Figure 35 shows the Mach number being controlled to a set point of 0.85 over a 15 minute interval. The angle of attack, alpha, is being steadily increased during this interval, while beta is maintained at zero. Fan RPM and tunnel temperature are steady as shown. At t = 98, the increases in alpha begin to cause the Mach number to drop. At t=99, a short duration (0.1 sec) raise correction brings the Mach number back to within tolerance. Further increases in alpha result in another decrease in Mach number. Another small raise correction minimizes the error. At t=104 a longer duration corrective pulse is applied after the Mach number response from the previous short duration pulse is classified as much less effective than the previous corrections. This is seen where the raise correction SOFM winner changes from 11 to 3. The corresponding raise correction SOFM winner change is shown in Figure 35. The raise correction SOFM winner corresponding to the Mach number response to the longer pulse is node 18.

A rather large decrease in angle-of-attack, from 12 degrees to near zero, causes the Mach number to jump up even further. Successive lower corrective pulses bring the Mach number back, while changing the SOFM winner of the lower corrective response, seen in Figure 35.

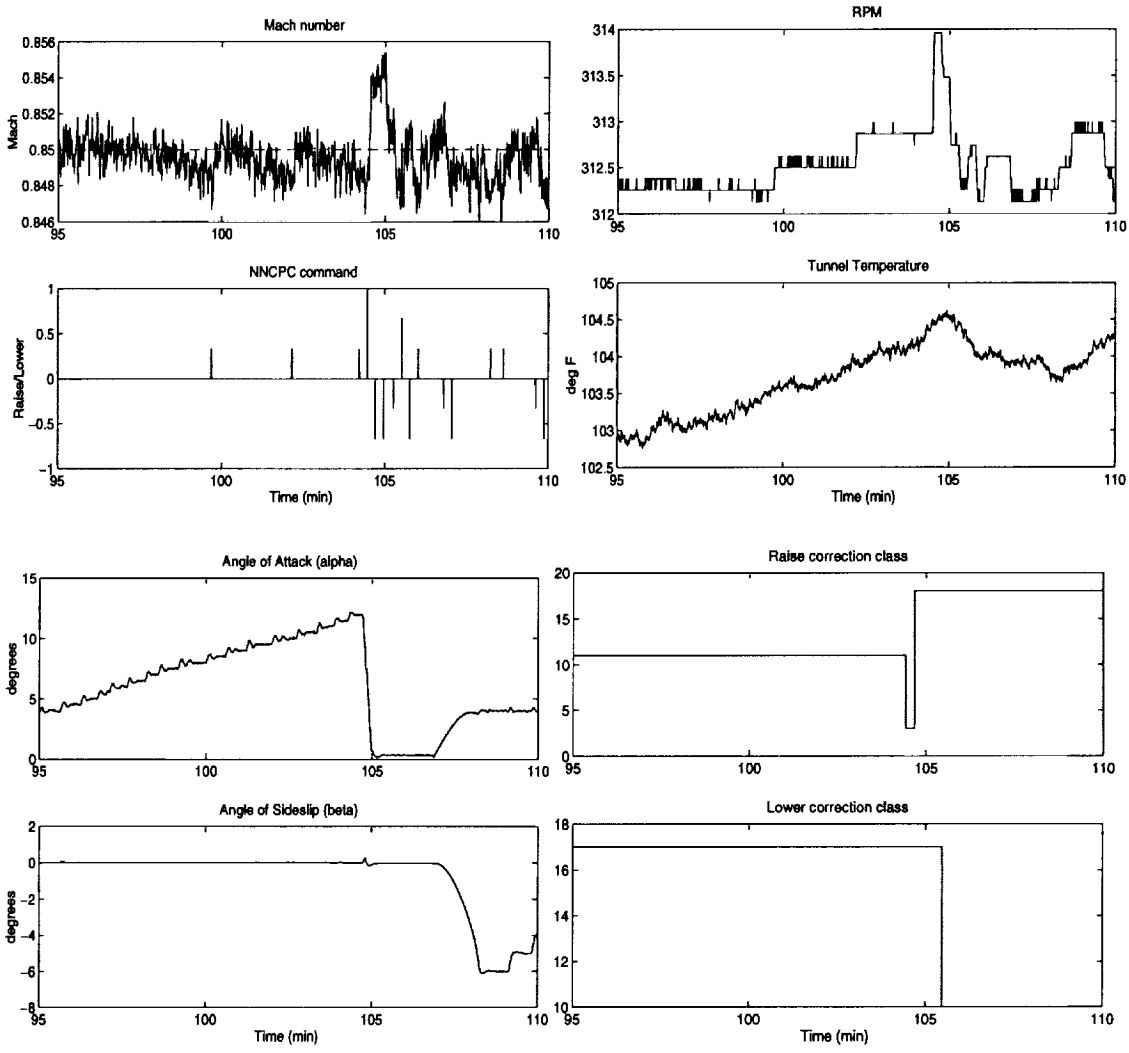


Figure 35. A 15 minute interval of the test

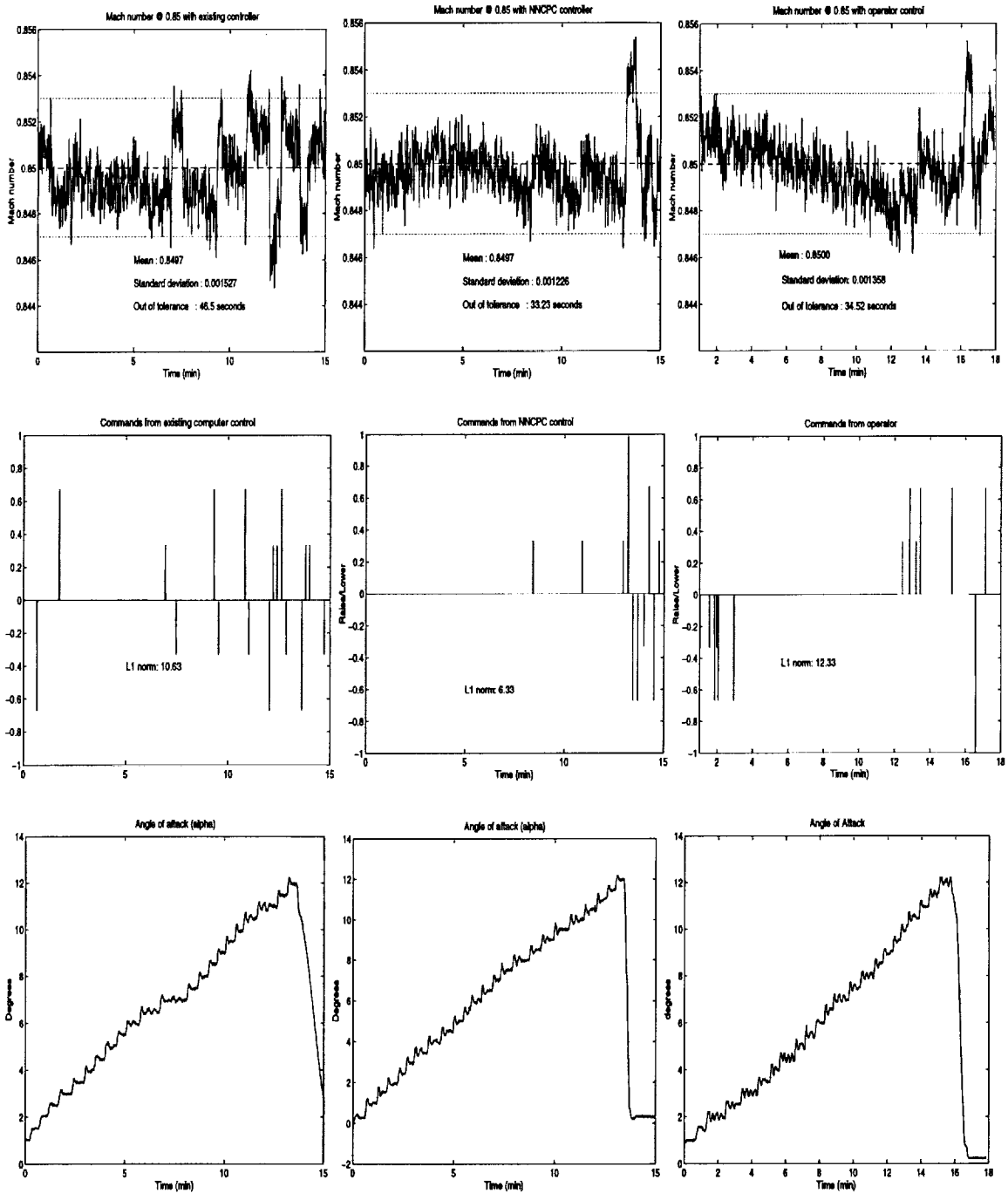


Figure 36. Comparison of PMMSC to existing control and expert operator

Comparison of PMMSC to Existing Controller and Expert Operator

Figure 36 compares the performance of the existing scheduled control, an expert operator, and the PMMSC under similar conditions over a nominal fifteen minute interval. Mach number, control commands, and test model angle-of-attack (disturbance) are shown for the existing control, an expert operator, and PMMSC control.

Derived metrics to quantify the comparisons between the three cases are the *time out of tolerance* and the $L1$ norm of the control input, u. The *time out of tolerance* is cumulative sum of time that the measured Mach number deviates beyond the required tolerance of 0.003 :

$$\begin{aligned} &\textit{time out of tolerance} = \sum_{k=1}^{k=N} \alpha \Delta t(k) \; ; \\ &\text{where} \quad a=0 \;\; \textit{if} \;\; \left| M(k) - M_{sp}(k) \right| \leq 0.003 \; ; \\ &\text{and} \quad \alpha = 1 \qquad \textit{otherwise} \; ; \\ &t \ni \left(t(0), t(1), \ldots, t(N) \right) ; \\ &\Delta t(k) = t(k) - t(k-1). \end{aligned} \tag{55}$$

The $L1$ norm of the control commands is :

$$L1[u] = \sum_{k=0}^{k=N} \left| u(k) \right| . \tag{56}$$

For this particular model, the angle-of-attack begins to mildly disturb the Mach number at approximately 5 degrees. Table 11 lists the reduction in the standard deviation of the Mach number, time out of tolerance, control effort, and time required to complete the sweep through the desired range of angle-of-attack while maintaining Mach number

steady for this comparison. For this particular test condition, the PMMSC performs slightly better than the expert operator, but with much less control effort and less time to complete the alpha sweep. Compared to the existing automatic control, the PMMSC maintains the Mach number within the desired tolerance much better with less control effort, completing the alpha sweep in less time, which is the most important figure of merit for the utilization of the facility.

	Existing controller	Expert Operator	PMMSC	% Reduction Auto / manual
Mean	0.8497	0.8500	0.8497	--
SD	0.001527	0.001358	0.001226	20 / 10
Time out of tolerance	46.5 s	34.52 s	33.2 s	29 / 4
L1 norm [u]	10.6	12.33	6.3	40 / 49
Alpha sweep	886 s	930 s	806 s	9 / 13

Table 11. Comparison of existing automatic control, expert operator, and PMMSC control

An additional metric on the control, the *control density,* ξ , was calculated by taking the sum of the absolute value of the control over a 50 sample sliding window:

$$\xi(k) = \xi[u(k)] = \sum_{i=0}^{i=49} |u(k-i)|. \tag{57}$$

The control density is used to compare the sparseness of the control between the PMMSC, the existing controller, and an expert operator, shown in Figure 37. This quantity measures the accuracy of the present control input, so in the PMMSC case it is a measure of the local linear models to predict the tunnel dynamics. The PMMSC is clearly the most sparse, but allows for increased density of the control when demanded by external disturbance, similar to the variation in control density employed by the expert

operator. This is in contrast to the existing automatic control, with fixed gains for a particular operating point resulting in a narrow range of control density.

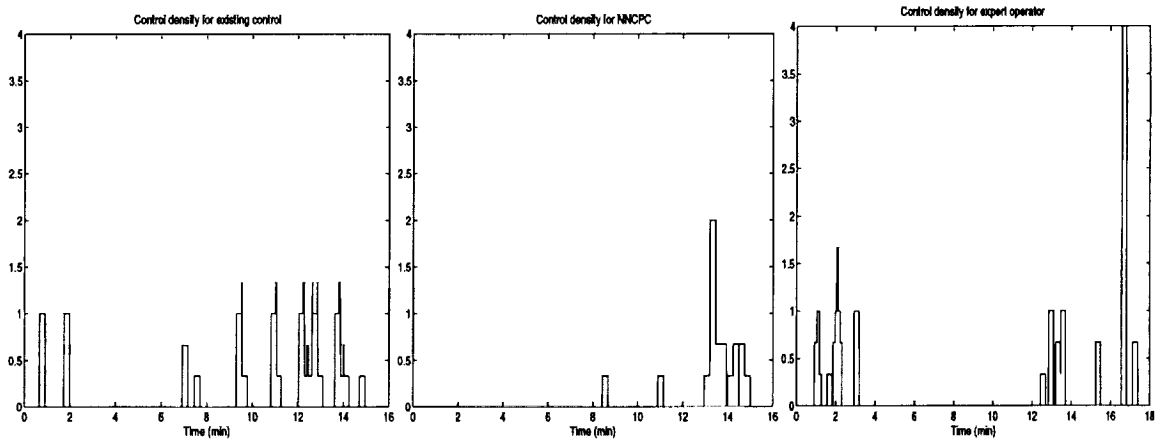


Figure 37. Comparison of Control Densities

Figure 38 compares the results of controlling the Mach number to several different set points over a nominal 28 minute interval. Mach number set points of 0.95, 0.9, and 0.6 are common to all three controllers. The PMMSC controls the Mach number to 0.85 versus 0.8 for the operator and existing controller. This difference is minimal and still provides a reasonable basis for comparison of the controllers. The angle-of-attack was varied extensively during all three runs. Again, the PMMSC maintains the Mach number within tolerance for a higher percentage of the time, with less expenditure of control effort. Table 12 lists the figures for time out of tolerance and control effort for the three runs. The PMMSC reduces the time out of tolerance on the order of 15-20 percent compared to the existing controller or an expert operator. The control effort is reduced by 12 percent compared to the existing controller, and 20 percent compared to an expert operator.

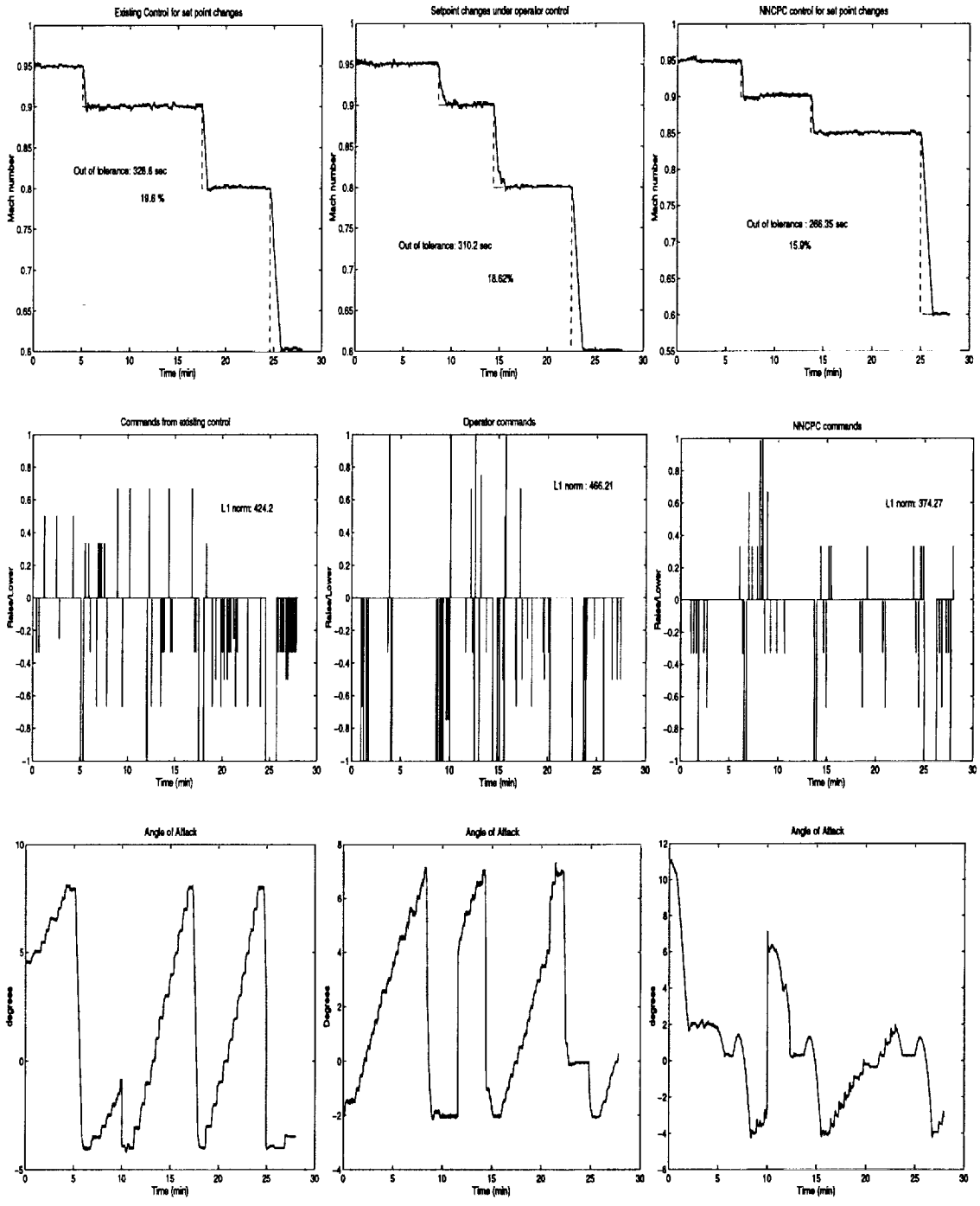


Figure 38. Comparison for controlling to several different set points

	Existing controller	Expert Operator	PMMSC	% Reduction Auto / manual
Out of tolerance	329 s	310 s	266 s	19.1 / 16.5
L1 norm [u]	424.2	466.2	374.3	11.7 / 19.7

Table 12. Comparison for controlling to several different set points

The differences in the control density for the three cases are illustrated in Figure 39. The variation in the control density is greatest for the expert operator and least for the existing controller. The PMMSC falls between the two cases in terms of variation of the control density, while requiring less overall control effort to provide less time out of tolerance.

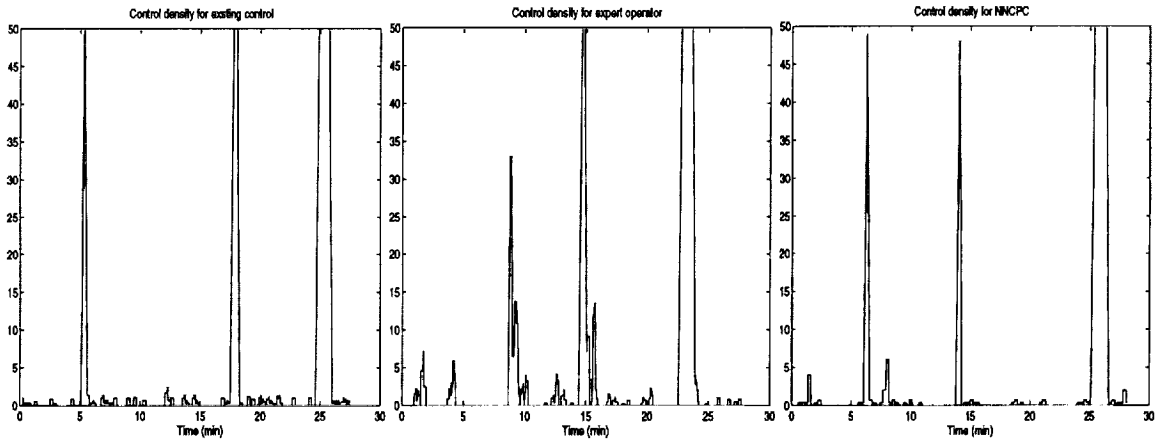


Figure 39. Comparison of Control Densities during set point changes

Experimental Results of Modeling the Tunnel Dynamics

Although the results of controlling the tunnel using the PMMSC imply some degree of success at modeling the tunnel dynamics, in this section we will explicitly compare the predicted Mach number responses to actual responses recorded during the experimental testing. This will provide some insight into the relation between the prediction error and the actual error observed while controlling the tunnel with the control sequence that was selected based on minimizing the predicted error.

The experimental results presented in this section are based on all the predicted and actual responses for the three hour control test shown earlier in Figure 31. This test consisted of rather lengthy segments where the Mach number was controlled to within the 0.003 tolerance at M=0.95, 0.90, 0.85, and 0.6. The control inputs during this test are predominately from input_class_0 (all zeroes), input_class_2 (ramp down), input_class_4 (end of ramp down), input_class_5 (small positive correction), input_class_6 (small negative correction), and input_class_8 (negative transition). A second control test, shown in Figure 45, provides results from predicting responses to input_class_1 (ramp up), input_class_3 (end of ramp up), and input_class_7 (positive transition). Table 13 lists the distribution among input classes for the two tests. Then, for each input_class, the relative frequency of the associated SOFM winners were determined and displayed in the corresponding histogram plots, Figures 40 through 44.

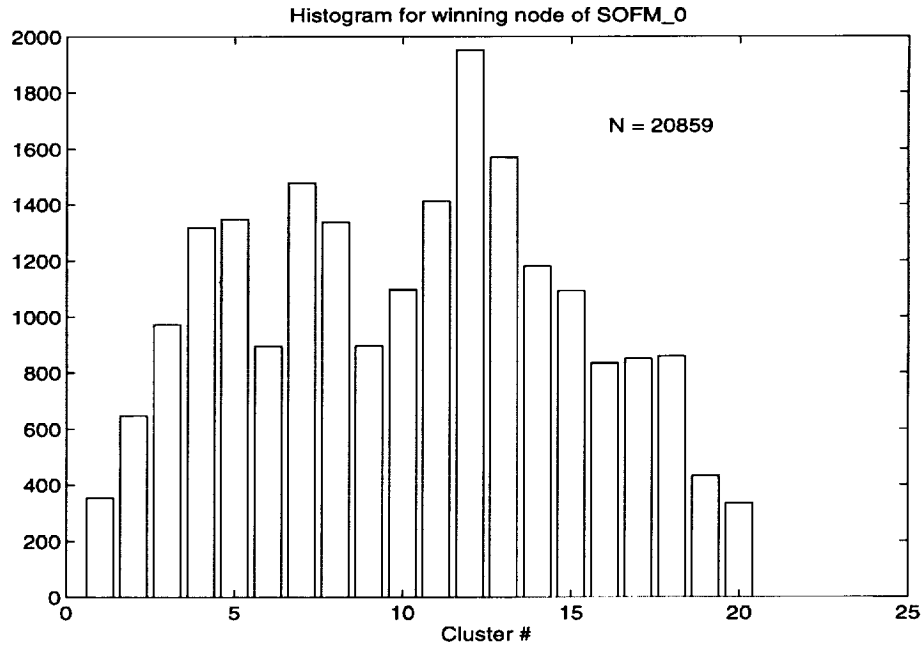


Figure 40. SOFM_0 winning nodes

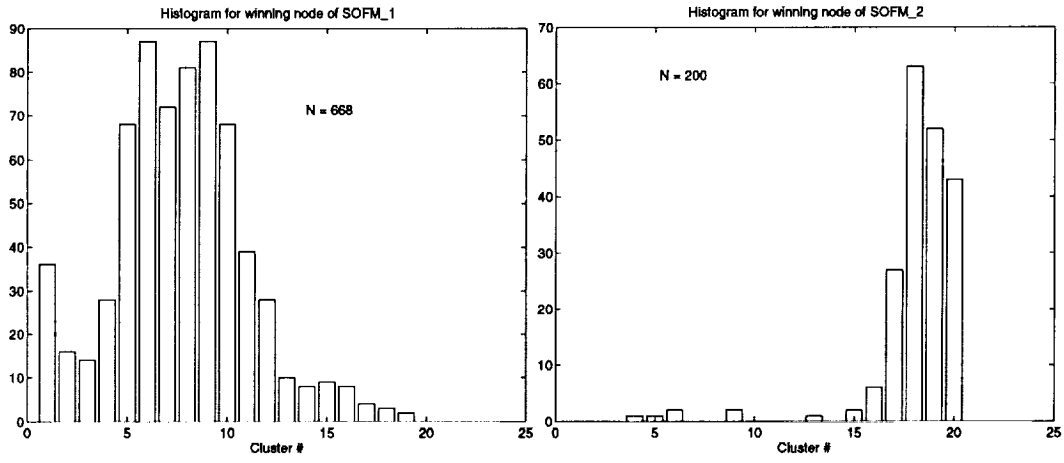


Figure 41. SOFM_1 and SOFM_2 winning nodes

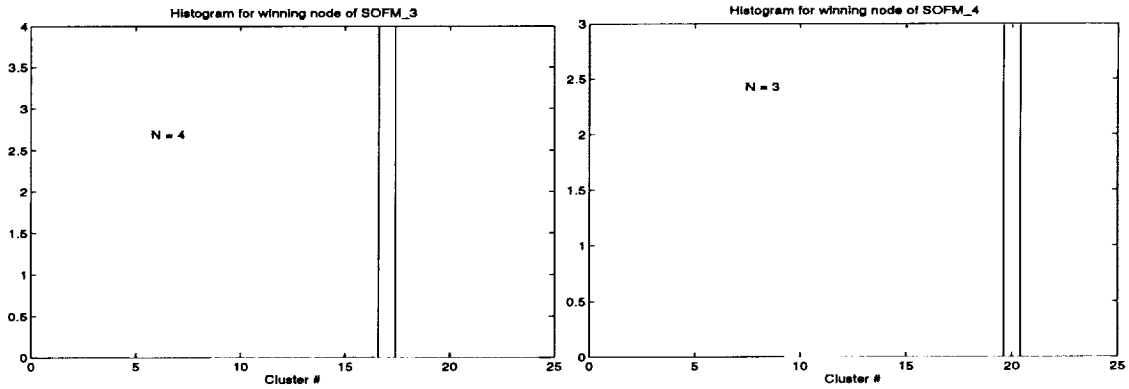


Figure 42. SOFM_3 and SOFM_4 winning nodes

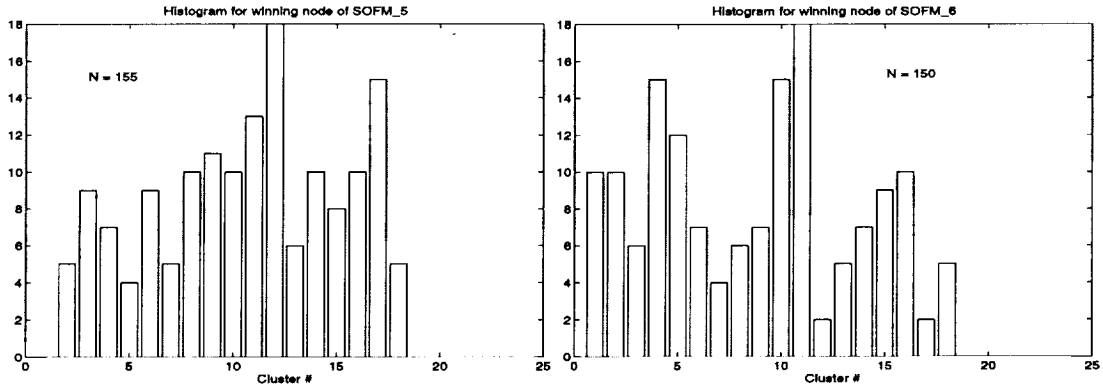


Figure 43. SOFM_5 and SOFM_6 winning nodes

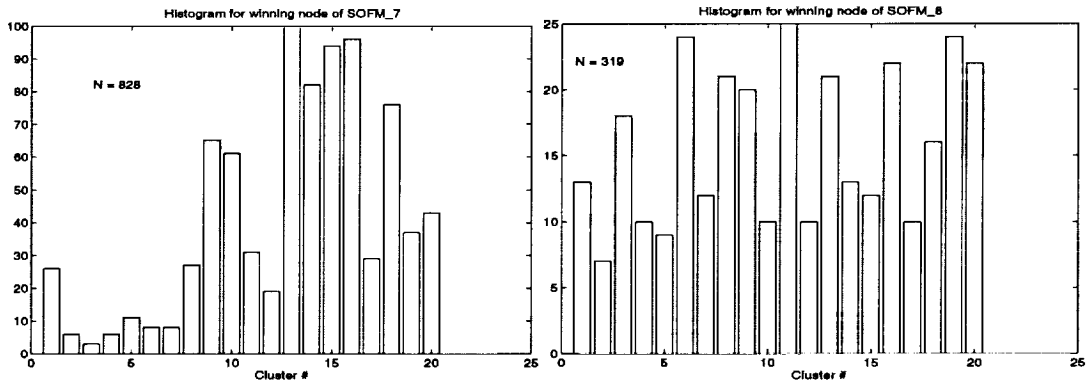


Figure 44. SOFM_7 and SOFM_8 winning nodes

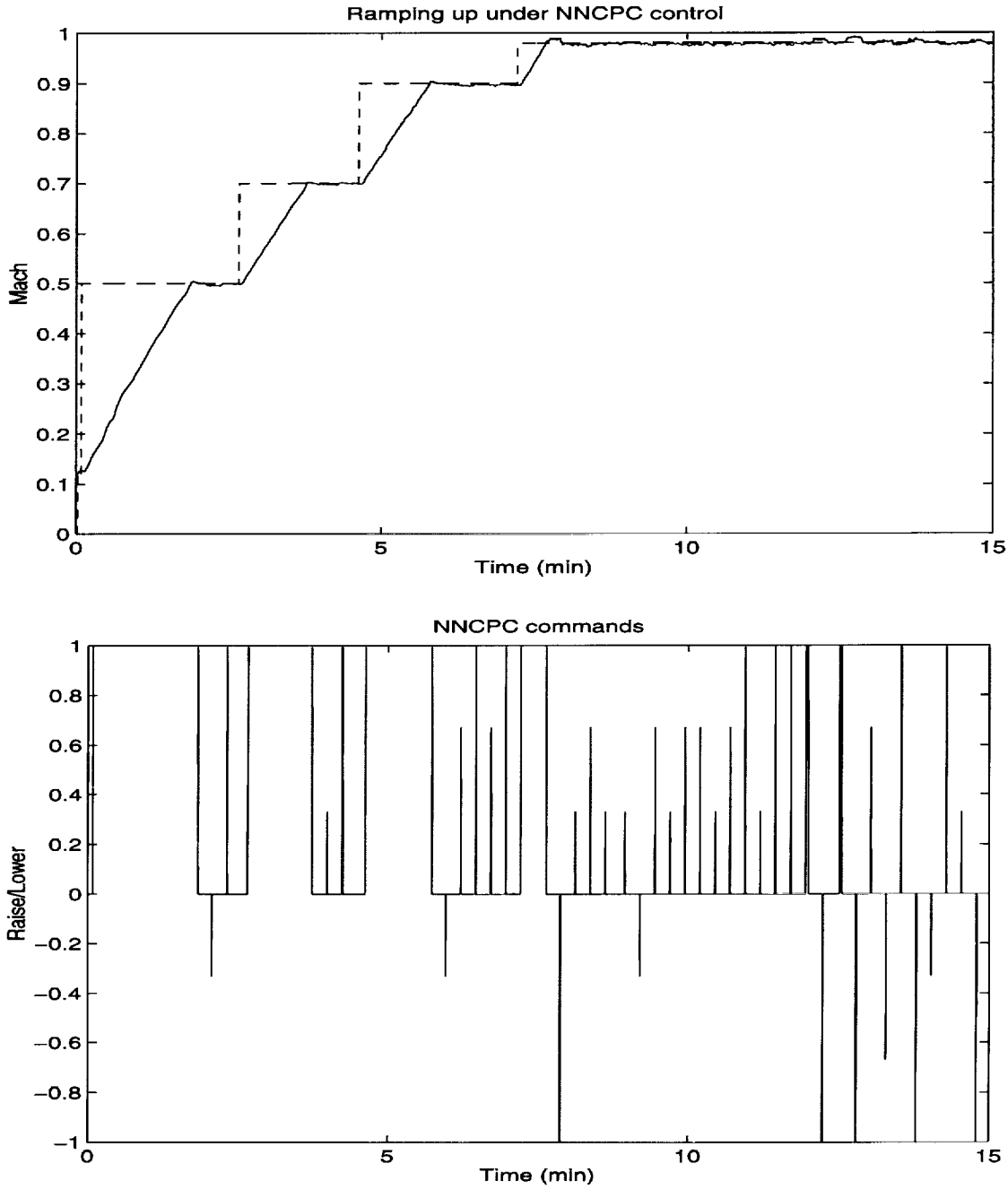


Figure 45. Ramping up with PMMSC control

input_class	Figure 31.	Figure 45.
0	20859	64
1	0	668
2	200	0
3	2	4
4	3	0
5	155	27
6	150	10
7	2	828
8	319	0

Table 13. Distribution among input_classes for Figures 31 and 45.

As a measure of the error between the predicted and actual responses to the control input sequence selected by the predictive controller, an average of the multi-step prediction error over the last 30 steps of the prediction was calculated by:

$$\xi = \frac{1}{30}\sum_{n=21}^{n=50}\left|\mathbf{M}(k+n)-\mathbf{M}^{*}(k+n)\right| \quad . \tag{58}$$

This measure is the average absolute value of the step-by-step prediction error over the last 30 prediction steps. This is the same interval over which the predictive controller evaluated the responses to candidate controls, thus providing a direct measure of the difference between the predicted response and the actual response. The average

absolute value of the error is a more intuitive choice over the Euclidean norm here, given the small absolute value of the control tolerance.

The SOFM's associated with input_classes_7 and _8 are only used to predict the Mach number response over the next ten sample periods, so the average multi-step prediction error is modified to cover only the first ten points of the predicted response:

$$\xi_{10} = \frac{1}{10}\sum_{n=11}^{n=20}\left|\mathbf{M}(k+n)-\mathbf{M}^{*}(k+n)\right| . \tag{59}$$

Figures 46 through 54 show ensembles of predicted and actual Mach number responses for all the input_classes. Additionally, the average multi-step prediction error for each prediction is also shown. The mean value and standard deviation for taken over all predictions for each input_class, are listed in Table 14.

input_class	N	Mean	SD	Control Function
0	10 000	0.000947	0.000401	Steady State
1	668	0.0065	0.0041	Ramp up
2	200	0.0039	0.0030	Ramp down
3	4	0.0046	0.0018	End of ramp up
4	3	0.0031	0.0019	End of ramp down
5	155	0.0018	0.0012	Positive correction
6	150	0.0017	0.0012	Negative correction
7	828	0.0014	0.0010	Positive transition
8	319	0.0016	0.0010	Negative transition

Table 14. Multi-step prediction errors for all input_classes

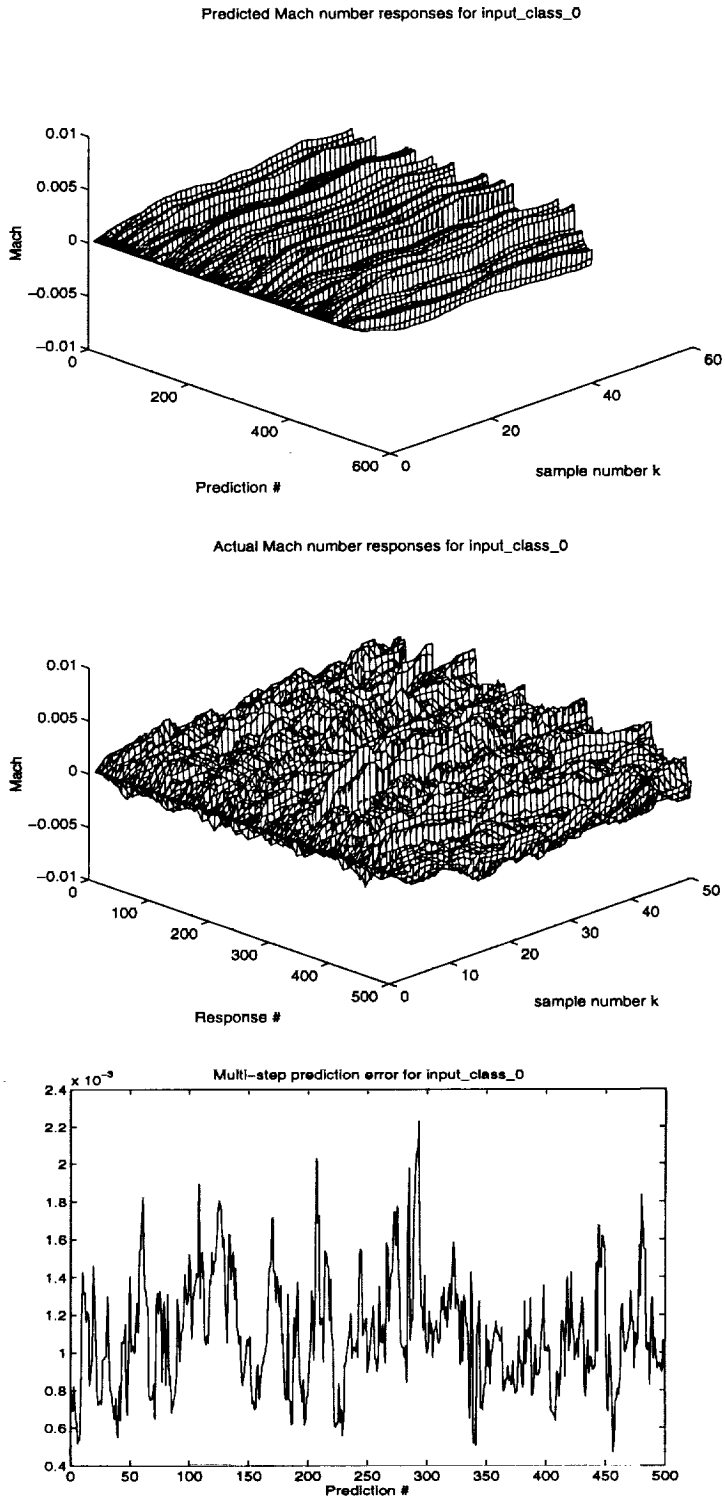


Figure 46. Predictions, responses and prediction error for input_class_0

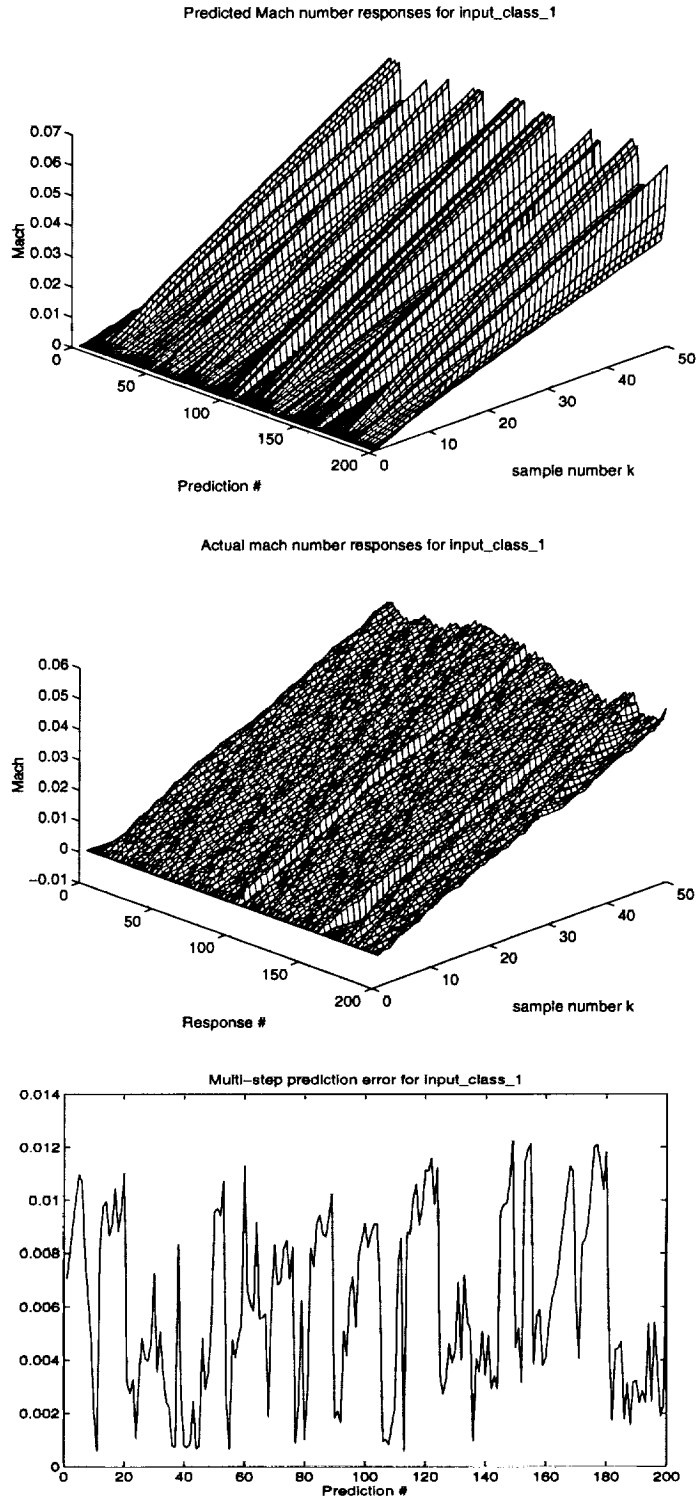


Figure 47. Predictions, responses and prediction error for input_class_1

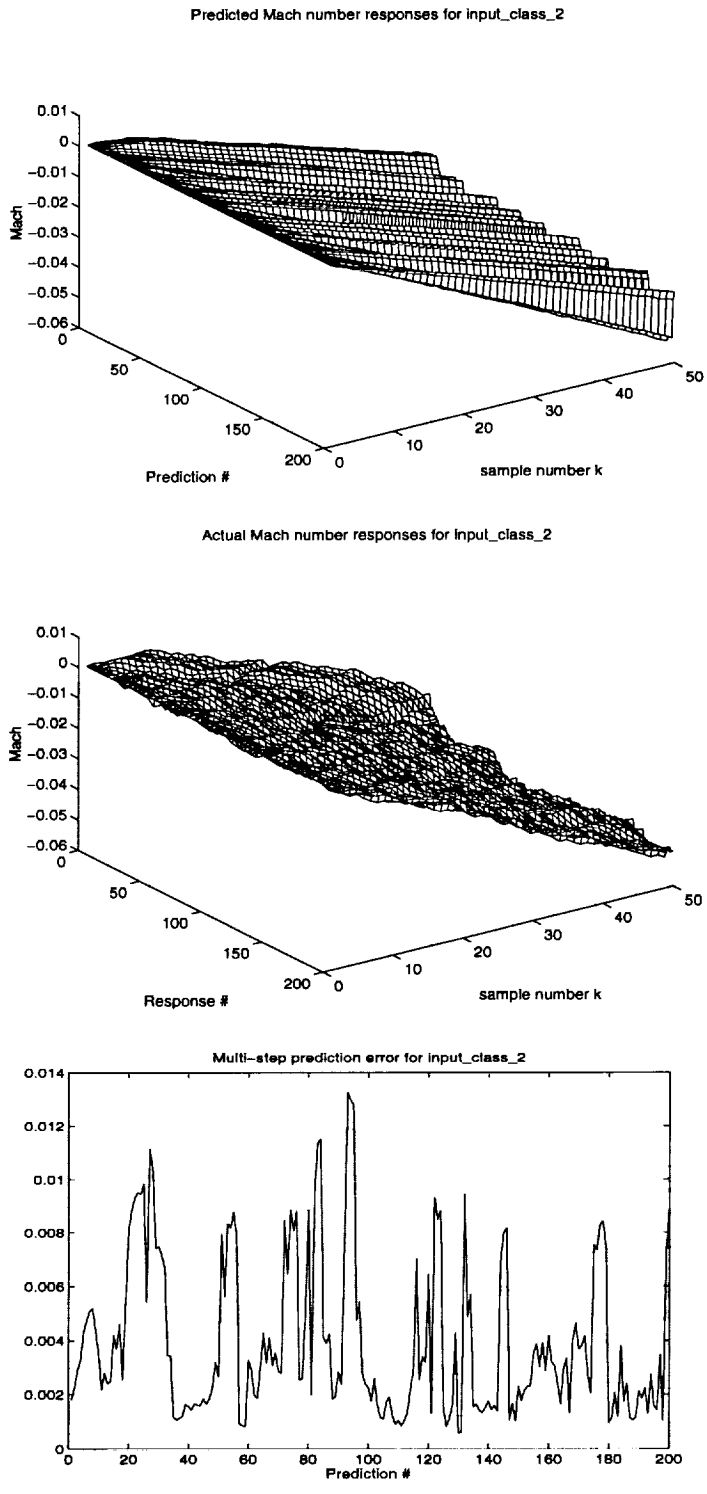


Figure 48. Predictions, responses and prediction error for input_class_2

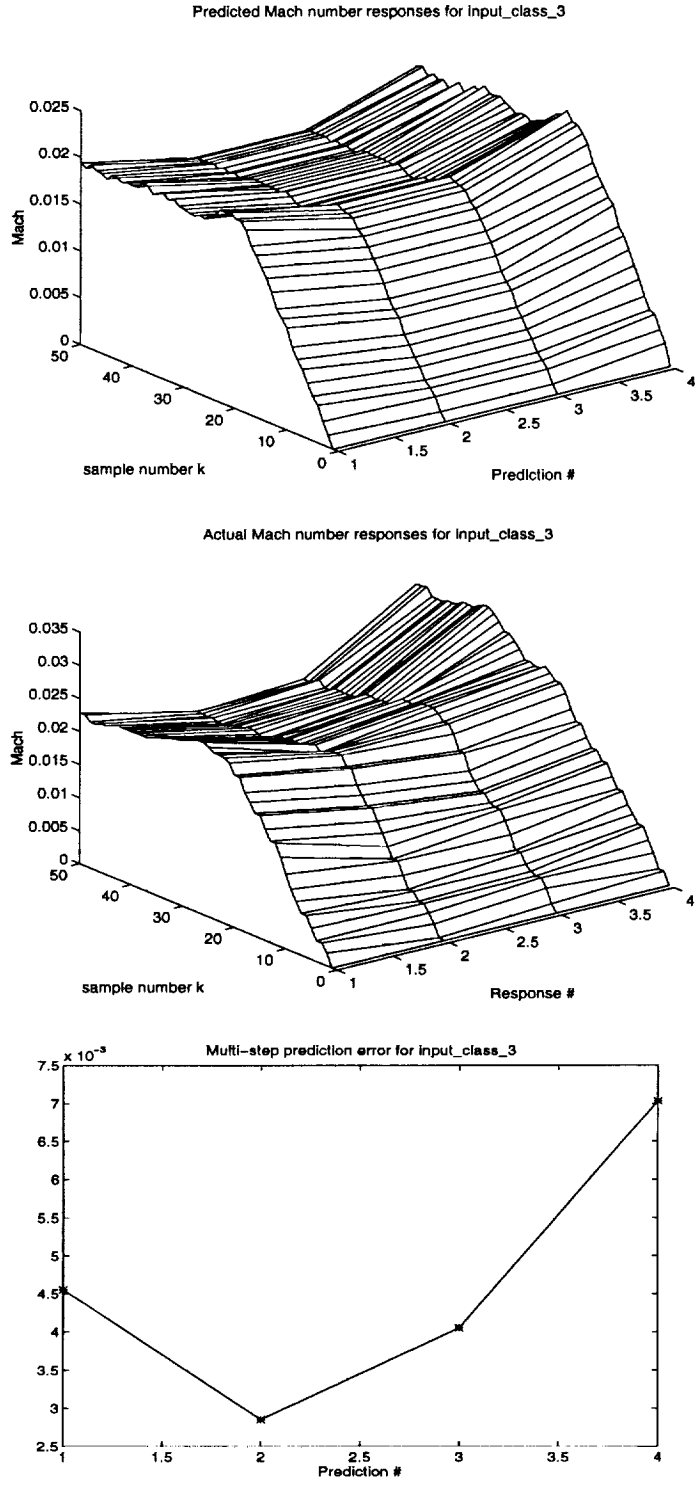


Figure 49. Predictions, responses, and prediction error for input_class_3

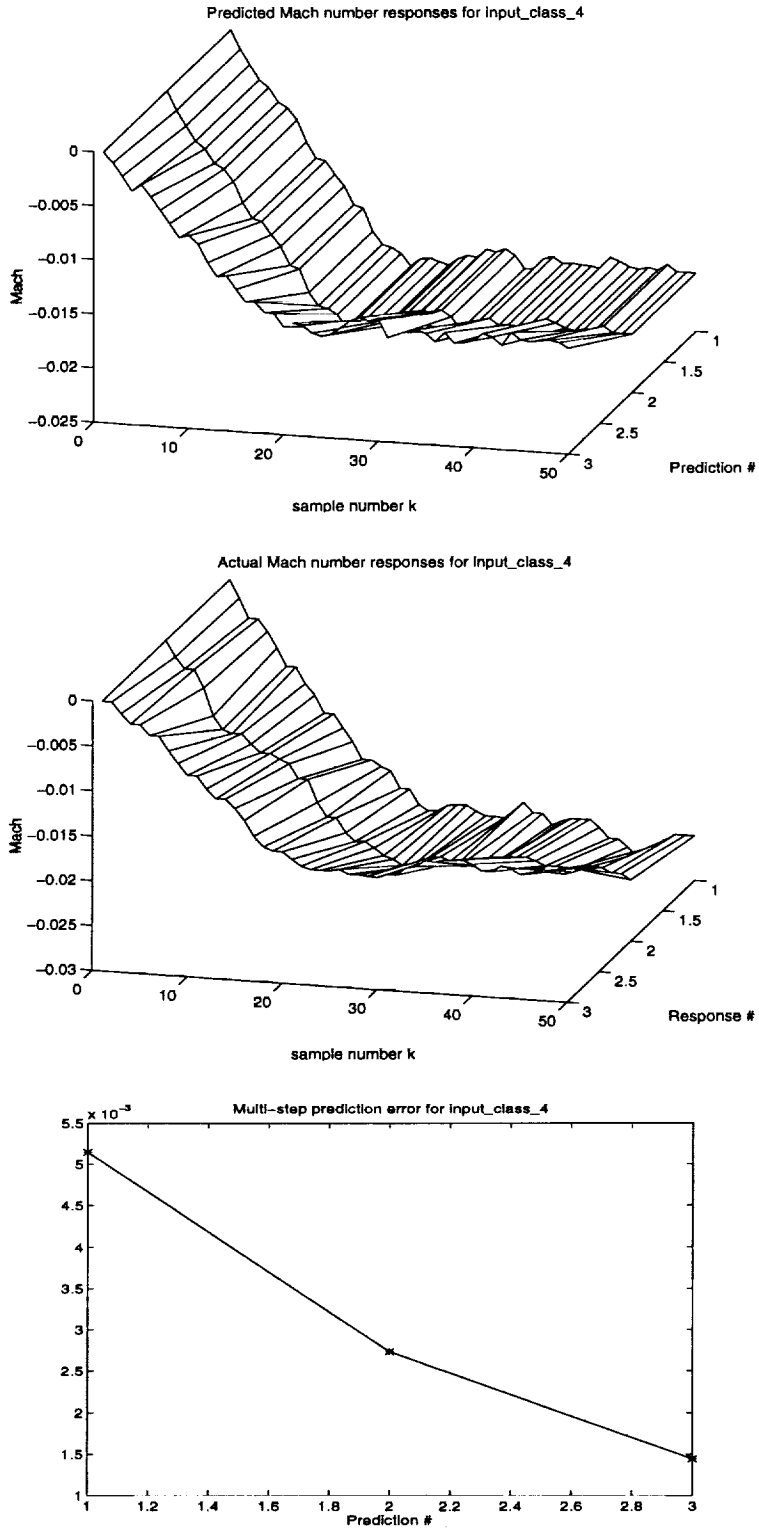


Figure 50. Predictions, responses, and prediction error for input_class_4

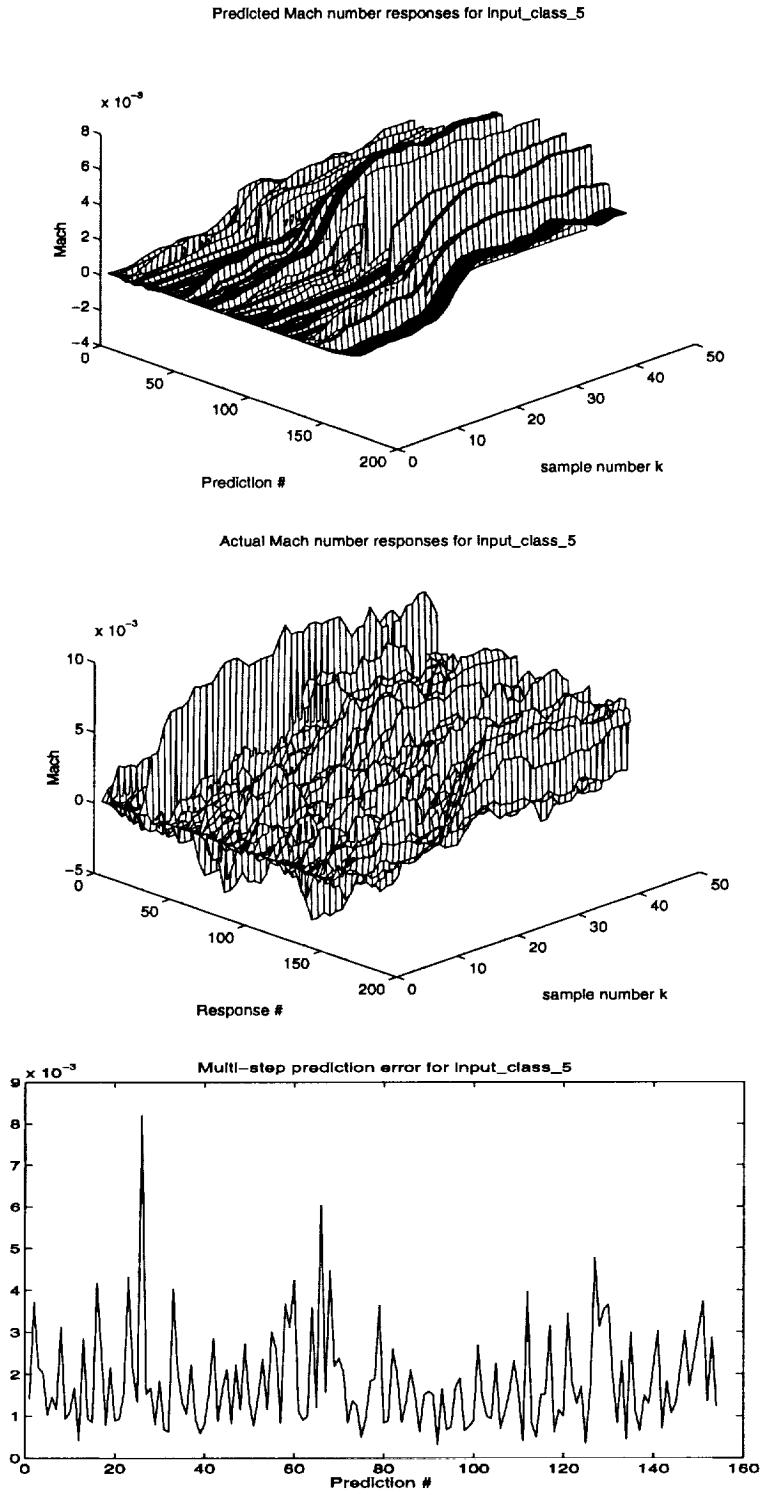


Figure 51. Predictions, responses, and prediction error for input_class_5

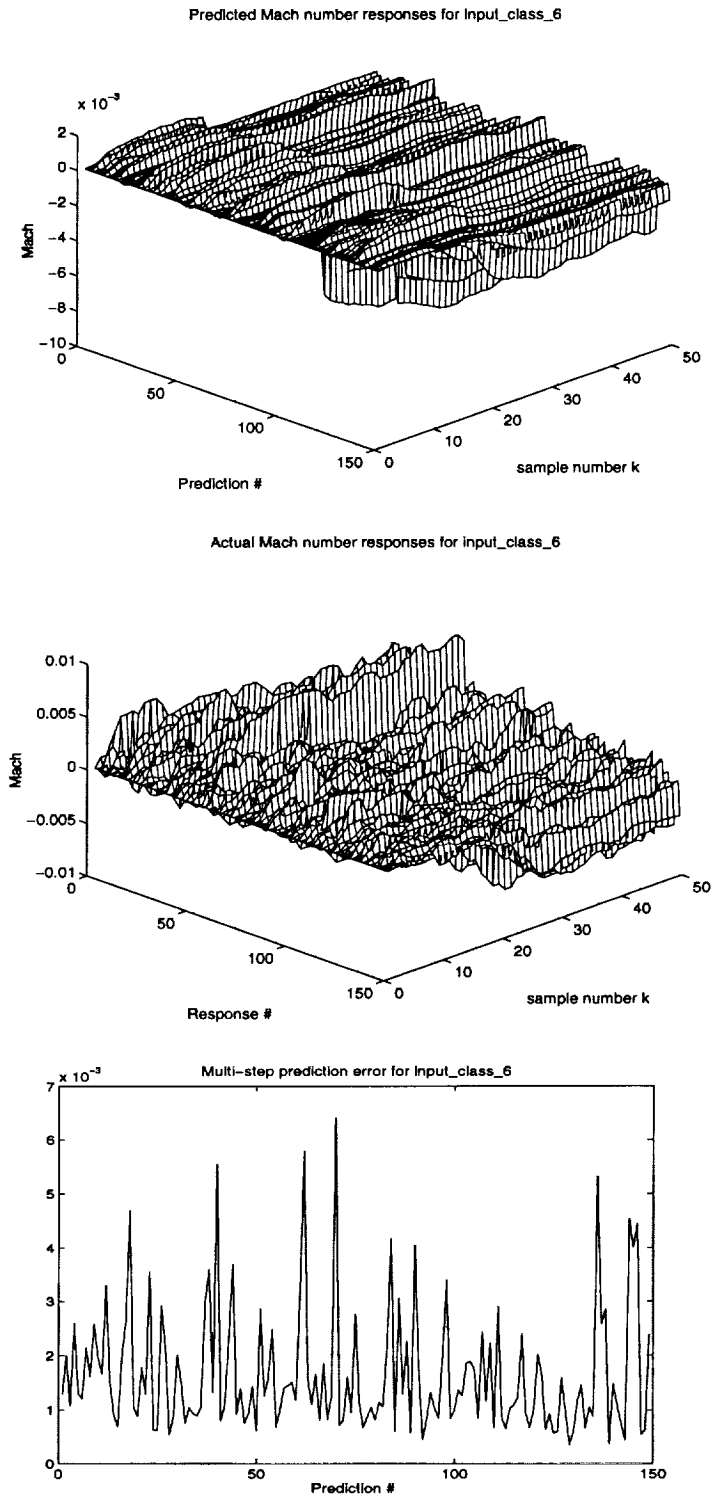


Figure 52. Predictions, responses and prediction error for input_class_6

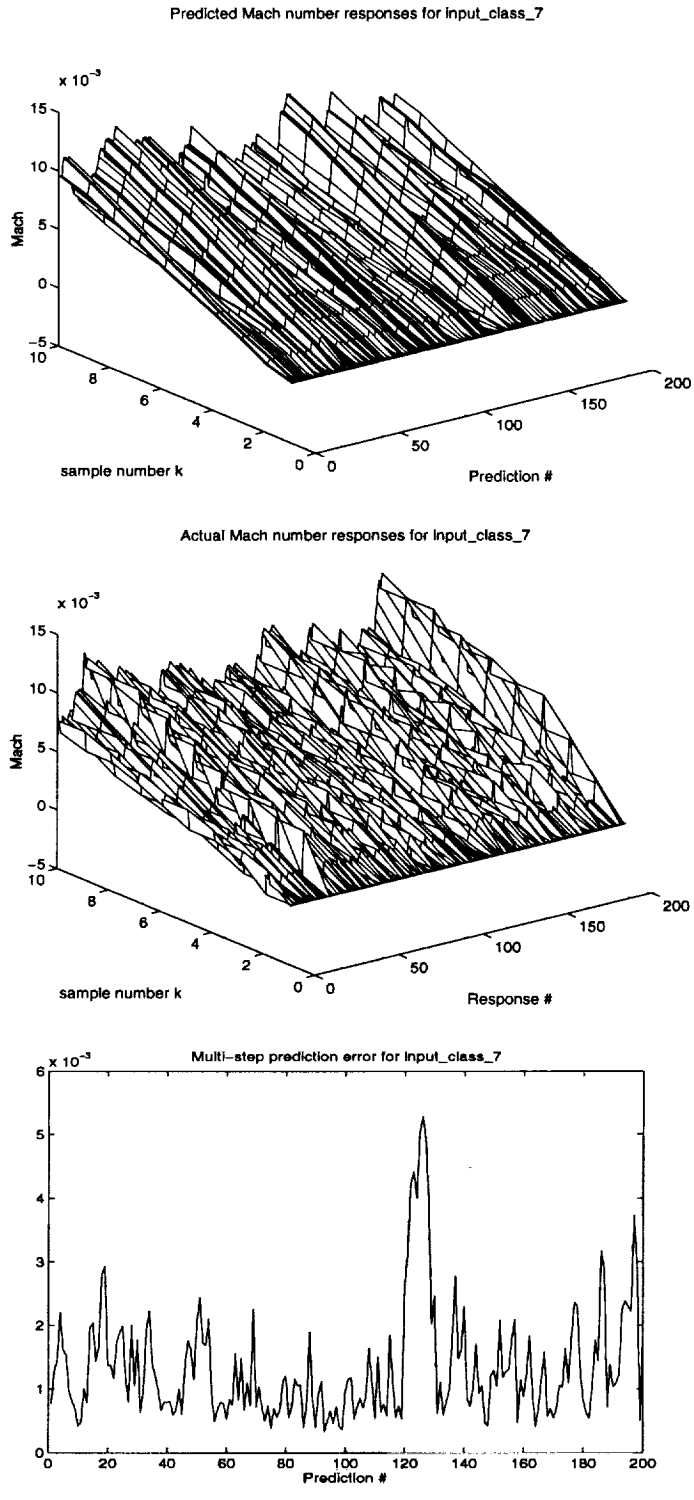


Figure 53. Predictions, responses, and prediction error for input_class_7

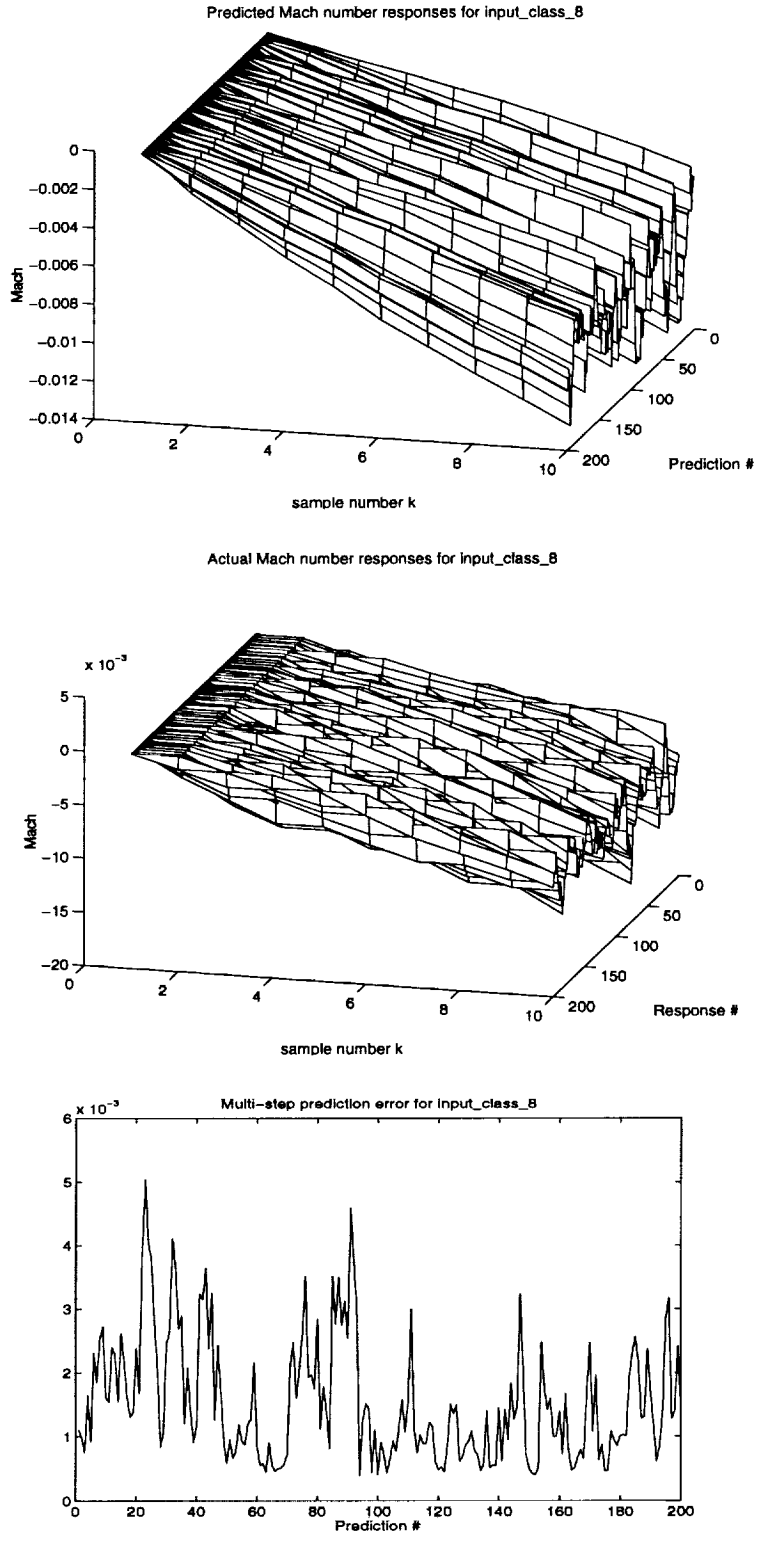


Figure 54. Predictions, responses, and prediction error for input_class_8

Figures 55 through 57 show typical predictions, the actual Mach number response, and the corresponding control input. The examples show both ramping and regulating cases at several Mach numbers. The Mach number is shown in solid and the predicted value as a dotted line. The goal of the linear model is to predict the steady state response of the tunnel, so the plots can be effectively divided into an initial transient phase corresponding to the first twenty samples and the following steady state phase, corresponding to the last 30 samples, which is the interval which the predictive controller evaluates the candidate sequences. The examples shown are illustrative of the performance found during operation, and we observe that the local linear models are predicting well the resulting responses.

Although examples of typical predictions are illustrative, consideration of the multi-step prediction errors and their corresponding statistics provides more insight into the overall accuracy of the predictions and consequently the accuracy of the control provided by the PMMSC. The predictions for both the regulatory cases, input_class_5 and input_class_6, have multi-step prediction errors with a mean plus one standard deviation less than the required control tolerance of 0.003. Figure 43, the histogram for these SOFM, indicates a good distribution of winners across these maps. Taken together with the steady state control tolerance achieved by the PMMSC, we can conclude that the locally linear predictors and consequently, the underlying discretization of the tunnel dynamics by the corresponding SOFM provides both sufficient resolution and coverage of state and control spaces.

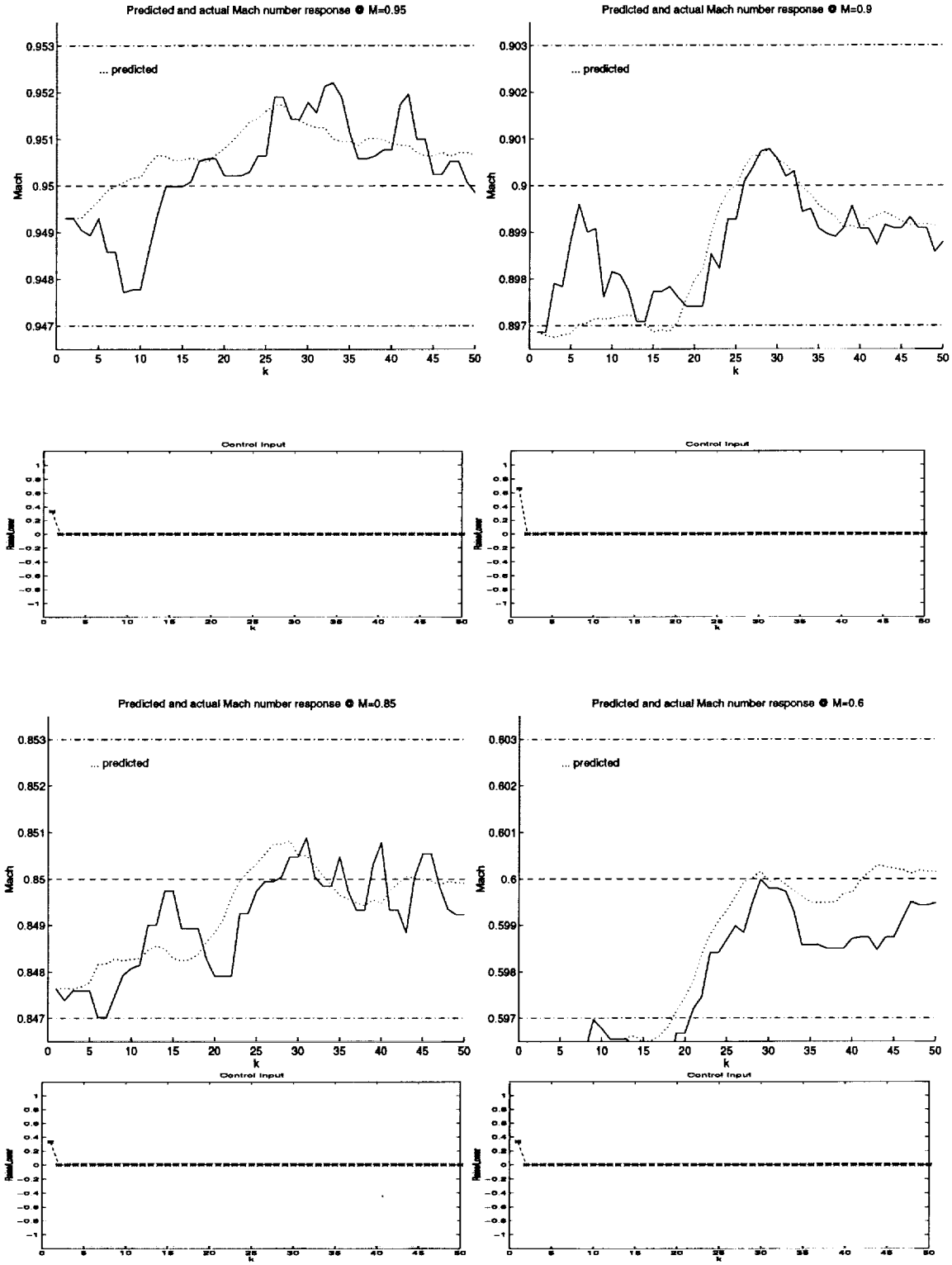


Figure 55. SOFM Predictions of Mach number in set point regulation

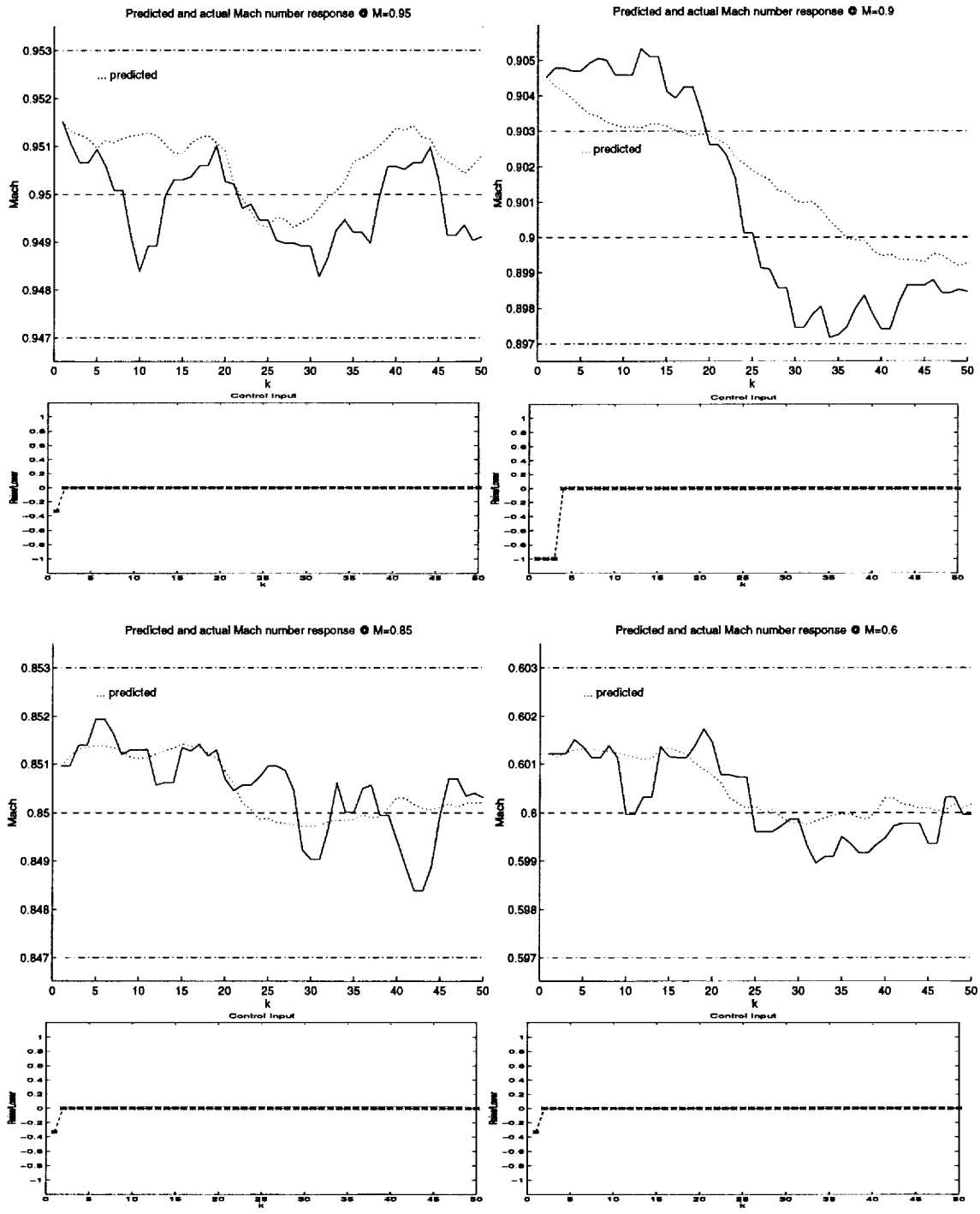


Figure 56. SOFM Predictions of Mach number in set point regulation

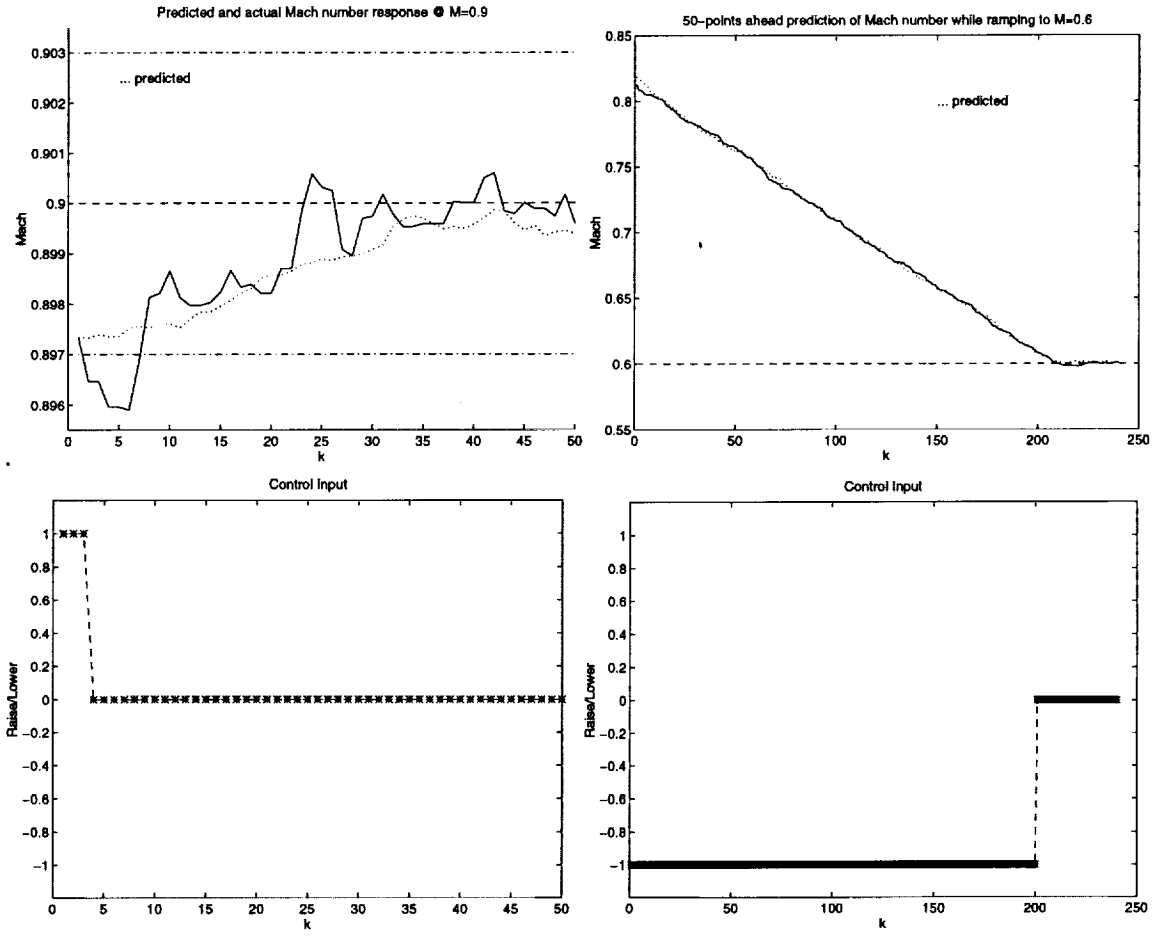


Figure 57. SOFM Predictions of Mach number in ramping

CHAPTER 6
CONCLUSIONS AND FUTURE RESEARCH

Conclusions

Modeling and controlling systems with a wide range of dynamic characteristics is a rich problem with many possible approaches. In this research, the method of local linear modeling based on the self-organizing feature map has been extended to a control framework as an approach to this problem.

The SOFM based modeling method was employed to develop a set of models which, collectively, described the system dynamic characteristics over the entire range of operation, but individually, represented the response of the system in some restricted region of both the state and control spaces of the system. The extension of the method allowed us to predict the system response to a small, but effective set of inputs, using the model which best describes the local dynamics. The input corresponding to the prediction that best satisfied the requirements at the output was then applied as the control. The overall result was the development of a controller, the PMMSC, which predicted the system response by switching to the best available model.

Two problems which naturally arise from this approach are: how to guarantee that the collection of models adequately cover all the dynamic regimes of the system, and how to select the model which best describes the local dynamical regime. Our SOFM based local linear modeling approach addresses both the problems with a computationally

efficient method. The SOFM guarantees that the repertoire of dynamics used for training are represented by the collection of local models and serves to identify the local dynamic regime. In a sense, the diverse plant dynamics are captured in a compact table look-up of linear models.

The combination of the locally linear models and a small set of candidate control sequences provided a computationally efficient method for multi-step predictive control. This was contrasted with generalized model predictive control, particularly in an environment which requires switching between models.

For this application, the fundamental control problem is one of regulating the tunnel Mach number to within tolerance of the desired set point under nonstationary loads. There were several characteristics of this problem that made it an attractive candidate for application of our method. The first was that the open-loop plant was stable, so the focus was on improved regulation. Secondly, there was a wealth of data to train a locally linear dynamic model of the tunnel under different dynamic conditions. Third, the control input is quantized to three values, which allowed for a meaningful clustering on a small set of representative control prototypes that were derived from the experiential knowledge of the tunnel.

The PMMSC was implemented on inexpensive computing hardware and used to control the wind tunnel to within the strict research requirements for three separate runs of three hours. The performance of the PMMSC was compared to both the existing controller and expert human operators by several metrics. The PMMSC provided improved performance with decreased control effort over both the existing controller and expert human-in-the-loop control.

Future Research

In this research, the method of SOFM based local linear modeling has been extended to a control framework. However, the resulting predictive, multiple model switching controller developed for this application does not represent a general control architecture, which points to directions for further work.

1. The control space for this application was partitioned by the construction of representative prototypes. This represented the incorporation of *a priori* knowledge about the operation of the system. A clustering of the inputs using the SOFM, would enhance the generality of the method
2. The plant being controlled in this application was stable. The method should be investigated for stabilization of an unstable plant, at least in simulation studies. This would open the possibility of identifying stabilizing and destabilizing manifolds of the control space, under the first suggestion for further work
3. Incorporation of an on-line adaptive model should be investigated. These could use the best SOFM based model as a starting point to speed the up the convergence
4. The SOFM algorithm could be implemented on-line and used to reformulate the maps based on accumulated knowledge as the system explores operating regions not covered in the training data.

APPENDIX
STABILITY CONSIDERATIONS

An approach for considering the stability of the overall system is to analyze the stability characteristics of a simple system with bang-zero-bang ($-\Gamma$, 0, $+\Gamma$) input, controlled using feedback and a combination of simple nonlinear functions. The system, considered in discrete time formalism, consists of an integrator, d pure delays, and a nonlinear controller implemented by:

1) a symmetric dead zone with zero output for inputs in the closed interval [$-\varepsilon$, $+\varepsilon$]

$$\begin{aligned} f(x) &= x \text{ if } x > \varepsilon \\ f(x) &= x \text{ if } x < -\varepsilon \\ f(x) &= 0 \text{ if } |x| \leq \varepsilon \\ &\textit{with } \varepsilon > 0 \end{aligned} \tag{60}$$

2) a *signum* function

$$\begin{aligned} f(x) &= \Gamma \text{ if } x>0 \\ f(x) &= -\Gamma \text{ if } x<0 \\ f(x) &= 0 \text{ if } x=0 \\ &\textit{with } \Gamma > 0 \end{aligned} \tag{61}$$

The system is shown in block diagram format in Figure 58.

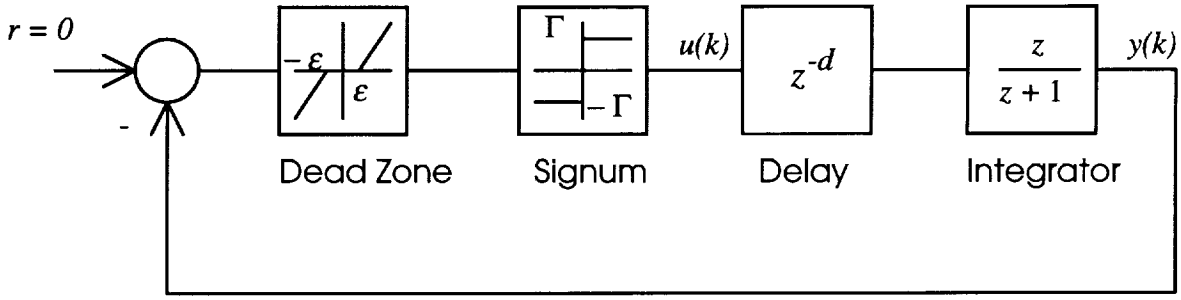


Figure 58. A simple nonlinear system with feedback

Consider the system with no delays, $d = 0$. The nonlinear control law yields the system equations:

$$\begin{aligned} y(k+1) &= y(k) - \Gamma \quad \text{if } y(k) > \varepsilon \\ y(k+1) &= y(k) + \Gamma \quad \text{if } y(k) < -\varepsilon \\ y(k+1) &= y(k) \qquad \text{if } |y(k)| \le \varepsilon \end{aligned} \tag{62}$$

In order to prove the stability of the system, a simple candidate Lyapunov function of the system is chosen:

$$V[\,y(k)\,] = \frac{1}{2}\, y^2(k) \tag{63}$$

For stability about the origin, $y = 0$, we require that [LaSalle, 1986]:

$$\begin{aligned} &1) \quad V[y] > 0 \text{ for all } y,\ y \neq 0 \\ &2) \quad V[y] = 0 \text{ for } y = 0 \\ &3) \quad \Delta V \le 0 \text{ along the trajectory of (62) for all } y \end{aligned} \tag{64}$$

where

$$\Delta V \equiv V[y(k+1)] - V[\,y(k)] \tag{65}$$

Conditions 1) and 2) are true by inspection for (64). Condition 3) needs to be considered in three distinct cases corresponding to (62):

Case 1: $y(k) > \varepsilon$

$$\Delta V = \frac{1}{2}\left[y(k) - \Gamma\right]^2 - \frac{1}{2}\left[y(k)\right]^2$$

$$\Delta V = \frac{1}{2}y^2(k) - y(k)\Gamma + \frac{1}{2}\Gamma^2 - \frac{1}{2}y^2(k)$$

$$= \frac{1}{2}\Gamma^2 - y(k)\Gamma$$

Thus $\Delta V \le 0$ if $y(k) \ge \frac{1}{2}\Gamma$, which implies that $\varepsilon \ge \frac{1}{2}\Gamma$

Case 2: $y(k) < -\varepsilon$

$$\Delta V = \frac{1}{2}\left[y(k) + \Gamma\right]^2 - \frac{1}{2}\left[y(k)\right]^2$$

$$\Delta V = \frac{1}{2}y^2(k) + y(k)\Gamma + \frac{1}{2}\Gamma^2 - \frac{1}{2}y^2(k)$$

$$= \frac{1}{2}\Gamma^2 + y(k)\Gamma$$

Thus $\Delta V \le 0$ if $y(k) \le -\left(\frac{1}{2}\Gamma\right)$, which implies that $\varepsilon \ge \frac{1}{2}\Gamma$

Case 3: $|y(k)| < \varepsilon$

$$\Delta V = \frac{1}{2}\left[y(k)\right]^2 - \frac{1}{2}\left[y(k)\right]^2$$

$$\Delta V = 0$$

Thus $\Delta V \le 0$ if $\varepsilon \ge 0$

Thus this system, with *d=0,* will be stable and within the bound *e* of the origin if the discretized control input at each instant is either $+\Gamma, -\Gamma,$ or 0, and *e* is greater than $\frac{1}{2}\Gamma$.

Now we consider the cases of a single delay $d = 1$. The nonlinear control law yields the system equations:

$$\begin{aligned} y(k+2) &= y(k+1) + \Gamma \quad if \;\; y(k) > \quad \varepsilon \\ y(k+2) &= y(k+1) - \Gamma \quad if \;\; y(k) < -\varepsilon \\ y(k+2) &= y(k+1) \qquad if \;\; |y(k)| \le \; \varepsilon \end{aligned} \tag{66}$$

For this case the Lyapunov candidate function includes the additional state associated with the delay:

$$V[\,y(k)\,] = \frac{1}{2}\, y^2(k) + \; \frac{1}{2}\, y^2(k+1) \tag{67}$$

and

$$\begin{aligned} \Delta V &\equiv V[y(k+1)] - V[y(k)] \\ &= \frac{1}{2} y^2(k+2) - \; \frac{1}{2} y^2(k) \end{aligned} \tag{68}$$

Again we evaluate the three distinct cases corresponding to (65), with the additional consideration for the added delay in the system:

Case 1: $y(k) > \varepsilon$

If $y(k-1) > \varepsilon$

$y(k+2) = y(k) - 2\Gamma$

If $|\,y(k-1)\,| \le \varepsilon$

$y(k+2) = y(k) - \Gamma$

If $y(k-1) < -\varepsilon$

$y(k+2) = y(k)$

For these three cases , the one that sets the minimum lower bound on ε

$$\Delta V = \frac{1}{2}\left[y(k) - 2\Gamma\right]^2 - \frac{1}{2}\left[y(k)\right]^2$$

$$\Delta V = \frac{1}{2} y^2(k) - 2y(k)\Gamma + 2\Gamma^2 - \frac{1}{2} y^2(k)$$

$$= 2\Gamma^2 - 2y(k)\Gamma$$

$$= 2\Gamma(\Gamma - y(k))$$

Thus $\Delta V \le 0$ if $y(k) \ge \Gamma$, which implies that $\varepsilon \ge \Gamma$

Case 2: $y(k) < -\varepsilon$

If $y(k-1) < \varepsilon$

$y(k+2) = y(k) + 2\Gamma$

If $|y(k-1)| \leq \varepsilon$

$y(k+2) = y(k) + \Gamma$

If $y(k-1) > \varepsilon$

$y(k+2) = y(k)$

Again, the lower bound on ε is deterrmined by:

$$\Delta V = \frac{1}{2}[y(k)+2\Gamma]^2 - \frac{1}{2}[y(k)]^2$$

$$\Delta V = \frac{1}{2}y^2(k) + 2y(k)\Gamma + 2\Gamma^2 - \frac{1}{2}y^2(k)$$

$$= 2\Gamma^2 + 2y(k)\Gamma$$

$$= 2\Gamma(\Gamma + y(k))$$

Thus $\Delta V \leq 0$ if $y(k) \leq -\Gamma$, which implies that $\varepsilon \geq \Gamma$

Case 3: $|y(k)| \le \varepsilon$

If $y(k-1) < -\varepsilon$

$y(k+2) = y(k) + \Gamma$

If $|y(k-1)| \le \varepsilon$

$y(k+2) = y(k)$

If $y(k-1) > \varepsilon$

$y(k+2) = y(k) + \Gamma$

$$\Delta V = \frac{1}{2}[y(k)+\Gamma]^2 - \frac{1}{2}[y(k)]^2$$

$$\Delta V = \frac{1}{2}y^2(k) + y(k)\Gamma + \frac{1}{2}\Gamma^2 - \frac{1}{2}y^2(k)$$

$$= \frac{1}{2}\Gamma^2 + y(k)\Gamma$$

$$= \Gamma(\frac{1}{2}\Gamma + y(k))$$

Thus $\Delta V \le 0$ if $y(k) \le -\frac{1}{2}\Gamma$, which implies that $\varepsilon \ge \frac{1}{2}\Gamma$.

Thus this system, with $d=1$, will be stable and within the bound ε of the origin if the discretized control input at each instant is either $+\Gamma, -\Gamma$, or 0, and ε is greater than Γ.

By induction, for the case of d delays we get the result that:

$$\Delta V = \frac{1}{2}[y(k) \pm (d+1)\Gamma]^2 - \frac{1}{2}[y(k)]^2$$

where $-$ is for $y(k) > \varepsilon$ and $+$ is for $y(k) < -\varepsilon$, then

$$\Delta V = \frac{1}{2}y^2(k) \pm (d+1)y(k)\Gamma \pm \frac{1}{2}(d+1)^2\Gamma^2 - \frac{1}{2}y^2(k)$$

$$= \frac{1}{2}(d+1)\Gamma^2 \pm (d+1)y(k)\Gamma$$

$$= (d+1)\Gamma\left\{\frac{d+1}{2}\Gamma \pm y(k)\right\}$$

Thus $\Delta V \le 0$ if $|y(k)| \ge \frac{d+1}{2}\Gamma$, which implies that $\varepsilon \ge \frac{d+1}{2}\Gamma$

The general result is that if :

$$\varepsilon \geq \left(\frac{d+1}{2} \right) \Gamma \tag{69}$$

then the system will be stable and $|y(k)| \leq \varepsilon$ as $k \rightarrow \infty$, i.e. the system will converge to a region around the origin bounded by ε.

For a system with long delays, this can impose a large loss of precision in the control of the output in order to guarantee stability. A strategy to circumvent this loss of precision while still providing stability is to apply sequences of control commands composed of m sample intervals of $\pm \Gamma$, followed by d sample intervals of zero input, for a total sequence length of $(m + d)$. Thus control sequences are determined every $(m+d)$ samples, and applied open-loop over the $(m+d)T$ interval, where T is the sample interval duration. Then if at time k, the controller selects a control sequence :

$$[\underbrace{\pm(\Gamma, \Gamma, \ldots \Gamma)}_{m \text{ samples}}, \quad \underbrace{0,0,\ldots 0}_{d \text{ samples}}] \tag{70}$$

then the output $y(k + m+d)$ is:

$$y(k+m+d) = y(k) \pm m\Gamma \tag{71}$$

Similar to the earlier result (69), the system controlled in this fashion will be stable to within ε of the origin if :

$$\varepsilon \geq \left(\frac{m}{2} \right) \Gamma \tag{72}$$

where $m = 1, 2, 3, \ldots$.

LIST OF REFERENCES

Alander, J.T., Frisk, M., Holmström, L., Hämäläinen, A., and Tuominen, J., (1991). "Process Error Detection using Self-Organizing Feature Maps," *Artificial Neural Networks*, Vol. II, pp. 1229-1232.

Bloch, G., Sirou, F., Eustache, V., Fatrez, P., (1997). "Neural Intelligent Control for a Steel Plant," *IEEE Transactions on Neural networks,* Vol. 8, Number 4, pp. 910-918.

Buggele, A. E., and Decker, A. J., (1994). *Control of Wind Tunnel Operations Using Neural Net Interpretation of Flow Visualization Records.* NASA Technical Memorandum 10668.

Cai, S., Toral, H., and Qiu, J., (1993). "Flow Regime Identification by a Self-Organizing Neural Network," *Proc. ICANN'93, Int. Conf. on Artificial Neural Networks*, p. 868.

Cai, Y., (1994). "The Application of the Artificial Neural Network in the Grading of Beer Quality," *Proc. WCNN'94, World Congress on Neural Networks*, Vol. I, pp. 516-520, Lawrence Erlbaum.

Card, J.P., Snidermann, D.L., and Klimasauskas, C., (1997). "Dynamic Neural Control for a Plasma Etch Process," *IEEE Transactions on Neural networks,* Vol. 8, Number 4, pp. 883-901.

Cho, S., Cho, Y., Yoon, S., (1997). "Reliable Roll Force Prediction in Cold Mill Using Multiple Neural Networks," *IEEE Transactions on Neural networks,* Vol. 8, Number 4, pp. 874-882.

Clarke, D.W., Mohtadi, C., and Tuffs, P.S. (1987). "Generalized Predictive Control - Part I. The Basic Algorithm," *Automatica*, Vol 23, No. 2, pp. 137-148.

Cooper, D. J., Megan, L., Hinde, R.F., (1992). "Disturbance Pattern Classification and Neuro-Adaptive Control," *IEEE Control Systems Magazine*, Vol.12, Number 2, pp. 42-48.

Garside, J.J., Brown, R.H., Ruchti, T.L., Feng, X., (1992). "Nonlinear Estimation of Torque in Switched Reluctance Motor Using Grid Locking and Preferential Training Techniques on Self-Organizing Neural Networks," *Proc. IJCNN'92, Int. Joint Conf. on Neural Networks*, Vol. II, pp. 811-816.

Haykin, Simon, (1994). *Neural Networks: A Comprehensive Foundation.* New York: Macmillan College.

Hopfield, J.J., (1982). "Neural Networks and Physical Systems with Emergent Collective Computational Abilitites," *Proceedings of the National Academy of Sciences of the U.S.A.*, Volume 81, pp.3088-3092.

Jackson, E.A., (1989). *Perspectives of nonlinear dynamics.* Cambridge: Cambridge University Press.

John, James E.A., (1984). *Gas Dynamics.* Newton, MA : Allyn and Bacon, Inc.

Kasslin, M., Kangas, J., Simula, O., (1992). "Process State Monitoring Using Self-Organizing Maps," *Artificial Neural Networks*, 2, Vol. II, pp. 1531-1534.

Kohonen, T., (1990). "The Self-Organizing Map," *Proceedings of the IEEE*, Vol. 78, No. 9.

Kohonen, T., (1995). *Self-Organizing Maps,* Berlin, Heidelberg, Germany: Springer-Verlag.

Lampinen, J., and Taipale, O., (1994). "Optimization and Simulation of Quality Properties in Paper Machine with Neural Networks," *Proc. ICNN'94, Int. Conf. on Neural Networks*, pp. 3812-3815.

LaSalle, J.P., (1986). *The Stability and Control of Discrete Processes.* New York: Springer-Verlag.

Matthews, C.P., and Warwick, K., (1995). "Practical Application of Self Organising Feature Maps to Process Modelling," *Proceedings of Engineering Applications of Neural Networks.*

May, Gary S., (1994). "Manufacturing IC's the Neural Way," *IEEE Spectrum*, Vol. 31, No. 9.

Mercer, C.E., Berrier, B.L., Capone, F.J., Grayston, A. M., Sherman, C.D., (1984). *Computations for the 16-Foot Transonic Tunnel - NASA Langley Research Center.* NASA Technical Memorandum 86319.

Morse, A.S., (1980). "Global Stability of Parameter Adaptive Systems," *IEEE Transactions on Automatic Control*, Vol.25, pp. 433-439.

Motter, M., and Principe, J. C., (1994). "A Gamma Memory Neural Network for System Identification," *Proceedings of IEEE International Conference on Neural Networks*, Vol. 5, pp. 3232-3237.

_____., (1995). "Classification and Prediction of Wind Tunnel Mach Number Responses using both Competitive and Gamma Neural Networks," *Proceedings of 1995 World Congress on Neural Networks*, Vol. 2, pp. 25-29.

Narendra, K.S., and Annaswamy, A., (1989). *Stable Adaptive Systems*. Englewood Cliffs, New Jersey: Prentice Hall.

Narendra, K.S. and Balakrishnan, J., (1994). "Intelligent Control using Switching and Tuning," *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, June 13-15.

Narendra, K.S., Balakrishnan, J., (1997). "Adaptive Control Using Multiple Models," *IEEE Transactions on Automatic Control*, Vol.42, No. 2, pages 171-187.

Narendra, K.S., Balakrishnan, J., and Ciliz, M.K., (1995). "Adaptation and Learning Usng Multiple Models, Switching, and Tuning," *IEEE Control Systems Magazine*, Vol.15, No. 3.

Narendra, K.S., Li, S., and Cabrera, J.B.D., (1994). "Intelligent Control using Neural Networks,' *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, June 13-15.

Narendra, K.S., Lin, Y.H., and Valavani, L.S., (1980). "Stable Adaptive Controller Design- Part II: Proof of Stability," *IEEE Transactions on Automatic Control*, Vol. 25, pp. 440-448.

Narendra, K.S., Mukhopadhyay, S., (1997). "Adaptive Control Using Neural Networks and Approximate Models," *IEEE Transactions on Neural Networks*, Vol.8, No.3, pages 475-485.

Peddrew, Kathryn H., (1981). *A User's Guide to the Langley 16-Foot Transonic Tunnel.* NASA Technical Memorandum 83186.

Principe, J., Hsu, H., and Kuo, J., (1994). "Analysis of Short Term Neural Memory Structures for Nonlinear Prediction," *NIPS*, pp. 1011-1018.

Principe, J., and Kuo, J-M., (1994). "Dynamic Modeling of Chaotic Time Series with Neural Networks," *NIPS*, pp. 311-318.

Principe, J.C., Kuo, J., Celebi, S., (1994). "An Analysis of the Gamma Memory in Dynamic Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 5, No. 2.

Principe, J. C., and Motter, M., (1994). "System Identification with Dynamic Neural Networks," *Proceedings of the 1994 World Congress on Neural Networks.*

Principe, J., and Wang, L., (1995). "Nonlinear Time Series Modeling with Self-Organizing Feature Maps." *IEEE Workshop on Neural Networks for Signal Processing*, pp. 11-20.

Robbins, H., and Monroe, S., (1951). "A Stochastic Approximation Method," *Annals of Mathematical Statistics*, Vol. 22, pp.400-407.

Ritter, H., Martinetz, M., Schulten, K., (1992). *Neural Computation and Self-Organizing Maps.* Reading, MA: Addison-Wesley.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J., (1986). "Learning Internal Representations by Error Back-Propagation," *Parallel Distributed Processing,* D.E. Rumelhart and J.L. McClelland, editors. Cambridge, MA: The MIT Press.

Soeterboek, Ronald A. M., Pels, A.F., Verbruggen, H.B., A. van Langen, G.C., (1991). "A Predictive Controller for the Mach Number in a Transonic Wind Tunnel," *IEEE Control Systems Magazine*, Vol. 11, Number 11, pp. 63-72.

Si, J., and Lin, S., (1997). "Weight Convergence and Weight Density of the Multi-Dimensional SOFM Algorithm," *Proceedings of the 1997 American Control Conference*, CD edition.

Takens, F., (1980). "Detecting Strange Attractors in Turbulence," *Dynamical Systems and Turbulence*, ed. by D.A. Rand and L.S. Young, Springer Lecture Notes in Mathematics, pp.365-381, Springer-Verlag, New York.

Warwick, K., (1996). "System Identification using neural networks," *Proceedings of the Conference on Identification in Engineering Systems*, M.I.Friswell and J.E. Mottershead, editors. Swansea, UK:University Wales Swansea.

Werbos, P.J., (1990). "Back Propagation Through Time: What It Does and How To Do It," *Proceedings of the IEEE,* 78(10), 1550-1560.

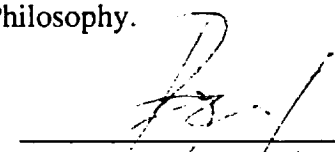
Willshaw, D.J., and C. von der Malsburg, (1976). "How Patterned Neural Connections Can Be Set Up by Self-Organization," *Proceedings of the Royal Society of London, Series B,* 194, 431-445.

Wu, J.-M., Lee, J.-Y., Tu, Y.-C., and Liou, C.-Y., (1991). "Diagnoses for Machine Vibrations Based on Self-Organization Neural Network," *Proc. IECON '91, Int. Conf. on Industrial Electronics Control and Instrumentation*, Vol. II, pp. 1506-1510.

BIOGRAPHICAL SKETCH

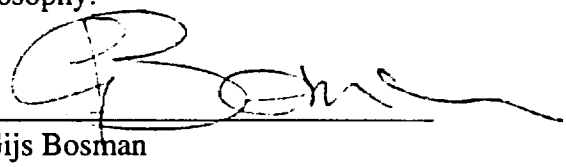
Mark A. Motter was born in [redacted], on [redacted] He served in the United States Navy from 1973 until 1979, and was honorably discharged at the rank of Electronics Technician First Class. He then began his formal engineering education at Old Dominion University in Norfolk, Virginia, receiving his BSEE, magna cum laude, and MSEE in 1983 and 1985, respectively. Since 1985 he has been employed at NASA Langley Research Center, primarily involved in the modeling and control of wind tunnels and associated experimental equipment. He is a member of the IEEE and a registered Professional Engineer.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



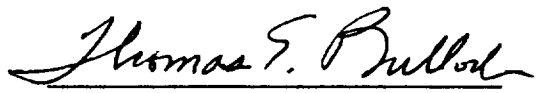
Jose C. Principe, Chairman
Professor of Electrical and Computer Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



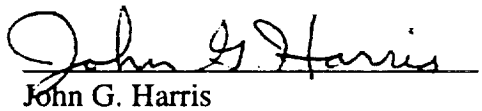
Gijs Bosman
Professor of Electrical and Computer Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Thomas E. Bullock
Professor of Electrical and Computer Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



John G. Harris
Assistant Professor of Electrical and Computer Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Loc Vu-Quoc
Associate Professor of Aerospace Engineering, Mechanics, and Engineering Science

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May, 1998

Winfred M. Phillips
Dean, College of Engineering

Karen A. Holbrook
Dean, Graduate School