Robust/Optimal Temperature Profile Control of a High Speed Aerospace Vehicle Using Neural Networks

Vivek Yadav\(^1\), Radhakant Padhi\(^2\) and S. N. Balakrishnan\(^3\)

**Abstract**

An approximate dynamic programming (ADP) based suboptimal neurocontroller to obtain desired temperature for a high-speed aerospace vehicle is synthesized in this study. A one-dimensional distributed parameter model of a fin is developed from basic thermal physics principles. ‘Snap-shot’ solutions of the dynamics are generated with a simple dynamic inversion based feedback controller. Empirical basis functions are designed using the ‘Proper Orthogonal Decomposition’ technique (POD) and the snap-shot solutions. A *low-order* nonlinear lumped parameter system to characterize the infinite-dimensional system is obtained by carrying out a Galerkin projection. An ADP based neurocontroller with a dual heuristic programming (DHP) formulation is obtained with a single-network-adaptive-critic (SNAC) controller for this approximate nonlinear model. Actual control in the original domain is calculated with the same POD basis functions through a reverse mapping. Further contribution of this paper includes development of an online robust neurocontroller to account for unmodeled dynamics and parametric uncertainties inherent in such a complex dynamic system. A neural network weight update rule that guarantees boundedness of the weights and relaxes the need for persistence of excitation (PE) condition is presented. Simulation studies show that in a fairly extensive but compact domain, any desired temperature profile can be achieved starting from any initial temperature profile. Therefore, the ADP and neural network based controllers appear to have the potential to become controller synthesis tools for nonlinear distributed parameter systems.

1. Introduction

In a strictly mathematical sense, almost all real-world engineering problems are distributed in nature and can be described by a set of partial differential equations (PDEs). Even though for many practical problems (e.g. dynamics of car, airplane etc.) a lumped parameter representation is often adequate, there are wide class of problems (e.g. heat transfer, fluid flow, flexible structures etc.) for which one must take the spatial distribution into account. These systems are also known as distributed parameter systems (DPS). In this paper, an ADP based neurocontrol of the temperature profile across the fin of a high speed...
aerospace vehicle, modeled as a nonlinear distributed parameter system, is considered. An interesting historical perspective of the control of distributed parameter systems can be found in [Lasiecka]. There exist theoretical methods for the control of distributed parameter systems [Curtain] in an infinite dimensional operator theory framework. One engineering approach to control of distributed parameter systems is to develop an approximate model of the system based on finite difference techniques and applying the control design tools directly on that approximated model [Padhi]. Another technique is to have a finite dimensional approximation of the system using a set of orthogonal basis functions via Galerkin projection [Holmes]. Galerkin projection normally results in high order lumped system representations. For this reason, attention is being increasingly focused in recent literature on the technique of proper orthogonal decomposition (POD) [Banks, Burns, Holmes, Ravindran, Singh]. In this technique, a set of problem oriented orthogonal functions are designed to approximately span the solution space of the original system of PDEs. This is done through the so-called “snap-shot” solutions, which are representative ensemble of the system states at arbitrary instants of time. In the process a very low order lumped model is created that is sufficient for practical controller design. For linear systems it has been proved that such an approach leads to an optimal representation in the sense that it captures the maximum energy of the infinite dimensional system with the least number of basis functions [Holmes, Ravindran]. Even though theorems do not exist, this idea has been successfully used in controller design for nonlinear systems, [Burns, Ly, Singh]. An important open question in this area is the construction of proper input functions to collect representative snap-shots. Quite often an open-loop controller is used for this purpose. Recently, there have been some attempts at modifications to this technique [Ravindran, Annaswamy] since a snap-shot solution based model may not ‘see’ some modes that could be excited in a feedback situation. This drawback is eliminated in our study with the use of a feedback linearized controller [Slotine] in generating snap-shots.

Rest of the paper is organized as follows: Neural network based controllers are reviewed in Section 2. In Section 3, a nonlinear model for the fin of a high speed aerospace vehicle that accounts for all of the three types of heat transfer, namely conduction, convection and radiation is developed. In Section 4, the controller design objectives are presented and the related problem formulation is given. In Section 5, a POD based basis function design and its subsequent use in a Galerkin projection scheme are discussed. In Section 6, the DHP based neural network synthesis procedure to design the optimal controller is described. Furthermore, the SNAC algorithm is presented. In section 7, an online neural network controller development to provide robustness against uncertainties is provided. In Section 8 the numerical
results are analyzed. Conclusions are drawn in Section 9. Proofs of the hypothesis used are presented in
the Appendix.

2. Background

There has also been a lot of interest in the use of neural networks for controller design that guarantees
desired performance in the presence of uncertainties and unmodeled dynamics. A multistage neural
network robot controller with guaranteed tracking performance was proposed by [Lewis]. This controller
was designed specifically for a serial link robot arm and was developed by using a filtered error/passivity
approach. Bounded tracking errors and bounded neural network weights were guaranteed. [Narendra]
presented many architectures of adaptive controllers using recurrent networks. A robust adaptive output
feedback controller for SISO systems with bounded disturbance was studied by [Aloliwi]. [Calise]
presented theoretical development and numerical investigation of an adaptive tracking controller using
neural networks. They provided stable weight adjustment rules for an online neural network and
simulation results for an F-18 aircraft model. [Leitner] designed an online adaptive neural network for use
in a nonlinear helicopter flight controller. The network helped the system with good tracking capabilities
in the face of significant modeling errors. An adaptive output feedback control scheme for uncertain
systems using neural networks was proposed by [Hovakimyan]. In [Lewis] the authors discussed an
online neural network that approximates unknown functions and is used in controlling the plant. A robust
adaptive control methodology that uses single hidden layer feedforward neural networks was presented by
[McFarland]. [Huang] developed a Lyapunov equation based theory for robust stability of systems in the
presence of uncertainties. The result is an ‘extra control’ which when added to the basic control effort
kept the system stable. This approach was illustrated through a helicopter problem. We develop a version
of extra control in this study to account for the unmodeled dynamics and parametric uncertainties.

The method of dynamic programming [Bryson,Lewis] is a powerful tool to solve many real life problems.
It produces a comprehensive solution by generating a family of optimal paths, or what is known as the
“field of extremals”. A major drawback of the dynamic programming approach is that it requires a
prohibitive amount of computation and storage and therefore, is impractical to use.

However, an approximate dynamic programming approach to circumvent the computational load with an
adaptive critic neurocontroller synthesis has been proposed in the literature [Balakrishnan, Prokhorov,
Werbos, White]. The adaptive critic methodology approximates and optimizes a control law iteratively
during the off-line training of ‘action’ and ‘critic’ networks, for an entire envelope of states. The action
networks capture the relationship between the state and control variables, where as the critic network
captures the relationship between state and costate variables in the development of optimal control theory.
There are many variations of this technique in the literature [Prokhorov]. Among many successful uses of
this method for nonlinear control design are [Balakrishnan,Ferrari,Venayagamoorthy]. Issues of convergence and stability of adaptive critic methods have been addressed by [Liu, Murphy].

3. Mathematical Model for the Problem

Mathematical model of heat transfer in a cooling fin of a high speed aerospace vehicle traversing through the earth’s atmosphere is developed in this section using concepts from basic thermal physics [Mills]. The development is illustrated in Figure 1.

Figure 1: Pictorial Representation of the Problem

The law of conservation of energy in the infinitesimal volume at a distance \( y \), having length \( \Delta y \) (as depicted in Figure 1), for this problem is described in (3.1).

\[
Q_y + Q_{gen} = Q_{y+\Delta y} + Q_{conv} + Q_{rad} + Q_{chg} \tag{3.1}
\]

where \( Q_y = -kA \frac{\partial T}{\partial y} \) is the entering rate of heat conduction. \( Q_{gen} = S(T,y)A\Delta y \) is the rate of heat generation. \( S(T,y) \) is the rate of heat generation per unit volume (also a function of both time \( t \) and spatial location \( y \)) and acts as the control variable for this problem. Note that the controller has been assumed to be continuous in the spatial domain. \( Q_{y+\Delta y} \) is the exiting rate of heat conduction, \( Q_{conv} = h P \Delta y (T - T_e) \) is the rate of heat convection, \( Q_{rad} = \varepsilon \sigma P \Delta y (T^4 - T_e^4) \) is the rate of heat radiation, and \( Q_{chg} = \rho C A \Delta y \frac{\partial T}{\partial t} \) is the rate of heat change [Miller].

A first-order Taylor series expansion is used to approximate the exiting rate of heat conduction as
\[ Q_{y+\Delta y} = Q_y + \left( \frac{\partial Q_y}{\partial y} \right) \Delta y \] (3.2)

T(t, y) represents the temperature, which varies with both time t and spatial location y. Definitions of the various parameters and the numerical values used in this research are given in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>Thermal conductivity</td>
<td>19 W/(m°C)</td>
</tr>
<tr>
<td>A</td>
<td>Cross sectional area</td>
<td>2’ × 3”</td>
</tr>
<tr>
<td>P</td>
<td>Perimeter</td>
<td>4’6”</td>
</tr>
<tr>
<td>h</td>
<td>Convective heat transfer coefficient</td>
<td>20 W/(m°C)</td>
</tr>
<tr>
<td>T_∞</td>
<td>Temperature of the medium in the immediate surrounding of the surface</td>
<td>100°C</td>
</tr>
<tr>
<td>T_∞</td>
<td>Temperature at a far away place in the direction normal to the surface</td>
<td>-40°C</td>
</tr>
<tr>
<td>ε</td>
<td>Emissivity of the material</td>
<td>0.965</td>
</tr>
<tr>
<td>σ</td>
<td>Stefan-Boltzmann constant</td>
<td>5.669×10^{-8} W/m² K^{-4}</td>
</tr>
<tr>
<td>ρ</td>
<td>Density of the material</td>
<td>7865 kg/m³</td>
</tr>
<tr>
<td>C</td>
<td>Specific heat of the material</td>
<td>0.46 kJ/(kg°C)</td>
</tr>
</tbody>
</table>

Table 1: Parameter Definitions and Numerical Values

Area A and perimeter P are computed assuming the following fin dimensions, 10’×2’×3”. Note that a one-dimensional approximation for the dynamics is used. In the study, this means a uniform temperature in the other two dimensions being arrived at instantaneously.

By using the expressions for individual terms and defining \( \alpha_1 \triangleq \frac{k}{\rho C} \), \( \alpha_2 \triangleq \frac{(Ph)}{(A\rho C)} \), \( \alpha_3 \triangleq \frac{(P\varepsilon\sigma)}{(A\rho C)} \) and \( \beta \triangleq \frac{1}{(\rho C)} \), the PDE representation of conservation of energy in (3.1) becomes

\[
\frac{\partial T}{\partial t} = \alpha_1 \left( \frac{\partial^2 T}{\partial y^2} \right) + \alpha_2 (T - T_\infty) + \alpha_3 (T_{\infty} - T_{\infty}^t) + \beta S(T, y) \tag{3.3}
\]

Boundary conditions for (3.3) are

\[
\frac{\partial T}{\partial y} \bigg|_{y=0} = c(t), \quad \frac{\partial T}{\partial y} \bigg|_{y=L} = 0 \tag{3.4}
\]

where the value of c(t) is dictated by the temperature profile T(t, y) at y = 0. An insulated boundary condition at the tip is assumed.
4. Controller Objectives

Design objectives and preliminary steps in a temperature feedback controller development for a high speed aerospace vehicle are presented in this section.

The main objective of the controller is to make the system reach a desired temperature profile on the fin, $T(y) \rightarrow T_d(y)$ as time $t \rightarrow \infty$, where $T_d(y)$ is the desired temperature distribution along the fin. The controller design is carried out such that $T_d(y)$ acts as a steady state condition after $T(y) \rightarrow T_d(y)$.

Temperature is normalized between 0 and 1 to help with round off errors. The normalized temperature, $X$ is defined as

$$X(y) = \frac{T(y) - T_{w_2}}{T_{\text{max}} - T_{w_2}}$$

where $T_{\text{max}} (=1100 \text{ K})$ and the normalized time, $\tau$ is defined as $\tau = \frac{t}{F}$ where $F (=1000)$. Let $X_d(y)$ be the non-dimensional temperature corresponding to $T_d(y)$. Defining the deviation(error), $x(y) = X(y) - X_d(y)$ and $S_i = \frac{\beta FS}{\Delta T}$ and substituting in (3.3), the error dynamics become

$$\frac{\partial x}{\partial \tau} = A_0 \left( \frac{\partial^2 x}{\partial y^2} \right) + A_1(X_d)x + A_2(X_d)x^2 + A_3(X_d)x^3 + A_4(X_d)x^4$$

$$+ \left[ \alpha F \left( \frac{\partial^2 X_d}{\partial y^2} \right) + \frac{\alpha F}{\Delta T} (\Delta T X_d + T_{w_1} - T_{\alpha}) + \frac{\alpha F}{\Delta T} (\left(\Delta T X_d + T_{w_2}\right)^4 - T_{\alpha}^4) \right] + S_i$$

(4.1)

where

$$A_0 = \alpha_1 F$$

$$A_1(X_d) = F \left( \alpha_2 + 4 \alpha_1 \left( X_d \Delta T + T_{w_1} \right) \right)$$

$$A_2(X_d) = 6 \alpha_1 F \Delta T \left( X_d \Delta T + T_{w_1} \right)^2$$

$$A_3(X_d) = 4 \alpha_1 F \Delta T^2 \left( X_d \Delta T + T_{w_1} \right)$$

$$A_4 = \alpha_1 F \Delta T^3$$

$$\Delta T = T_{\text{max}} - T_{w_2}$$

Note that the coefficients of non-linear terms (4.1) and (4.2) are functions of the desired temperature profile. The controller development now becomes a regulator problem. The steady state control solution $S_i^*$ is obtained by substituting $X_d$ in place of $X$ in (3.3) and imposing the steady-state condition.

$$S^*(y) = -\frac{1}{\beta} \left[ \alpha_1 \Delta T \left( \frac{\partial^2 X_d}{\partial y^2} \right) + \alpha_2 \left( \Delta T X_d + T_{w_1} - T_{\alpha} \right) + \alpha_3 \left( \left(\Delta T X_d + T_{w_2}\right)^4 - T_{\alpha}^4 \right) \right]$$

(4.3)
The $S_i'$ in (4.3) acts as a feedforward controller for this problem. A feedback controller (see Section 4.2) is added to this control to yield the total control.

4.1 Feedback Controller

The actual temperature and control variables are written in terms of the desired final (steady) state temperature profile $X_d(y)$ and the control $S_i'(y)$.

$$
X(t, y) = X_d(y) + x(t, y)
$$
$$
S_i(t, y) = S_i'(y) + u(t, y)
$$

(4.4)

In (4.4), $x(t, y)$ and $u(t, y)$ are the deviations from their respective steady state values. Next, the deviation dynamics are developed by substituting (4.4) in (4.1) and simplifying the resulting equation. Consequently, we get

$$
\frac{\partial x}{\partial t} = A_0 \left( \frac{\partial^2 x}{\partial y^2} \right) + F(x, X_d) + u
$$

(4.5)

where

$$
F(x, X_d) = (A_1(X_d)x + A_2(X_d)x^2 + A_3(X_d)x^3 + A_4(X_d)x^4).
$$

Similarly, the boundary conditions are expressed in terms of the deviations from the steady state values as

$$
\frac{\partial x}{\partial y} \bigg|_{y=0} = c(t) - \frac{\partial X_L}{\partial y} \bigg|_{y=0}, \quad \frac{\partial x}{\partial y} \bigg|_{y=L} = -\frac{\partial X_L}{\partial y} \bigg|_{y=L}
$$

(4.6)

The purpose of the feedback controller $u(t, y)$ is to make the deviations from the steady state conditions, i.e. $x(t, y) \to 0$ as time $t \to \infty$. As the state deviations tend to zero with time, the associated control effort goes to zero. These goals are achieved by minimizing a quadratic cost function in (4.7).

$$
J = \frac{1}{2} \int_0^L \int_0^L \left( q \ x^2 + r \ u^2 \right) dy \ dt
$$

(4.7)

In (4.7) $q \geq 0$ and $r > 0$ are the weights that express the designer’s concern for excessive deviations from the nominal and the control effort respectively. Equations (4.5) – (4.7) define the complete optimal control problem.

5. Low-Order Lumped Parameter Approximation

This section discusses the development of low-order finite-dimensional models for controller synthesis.

5.1 Procedure to Generate Snapshot Solutions
Snapshot solutions are generated by starting the solution process from different initial conditions that satisfy the boundary conditions. In order to generate snapshot solutions, the simulation is carried out for a fixed amount of time and snapshots are selected at equally spaced instants on this trajectory. This process is repeated for different initial conditions. Rather than using chosen input functions, a feedback linearization [Slotine] based control is used to simulate the closed loop behavior of the system.

5.2 Proper Orthogonal Decomposition: A Brief Review

Proper Orthogonal Decomposition (POD) is a technique for determining an optimal set of basis functions, with a set of snapshot solutions obtained by the method described in section 5.1.

Let \( \{ x_i(y); 1 \leq i \leq N, \ y \in \Omega \} \) be a set of \( N \) snapshot solutions (observations) of a physical process over the domain \( \Omega \) at arbitrary instants of time. Let the dimension of \( x_i(y) \) be \( n \times 1 \). The goal of the POD technique is to find a set of basis functions \( \varphi \) such that \( I \) is maximized in (5.1).

\[
I = \frac{1}{N} \sum_{i=1}^{N} \left| \left| \langle x_{i,\text{act}}, \varphi_{\text{act}} \rangle \right| \right|^2 / \langle \varphi_{\text{act}}^*, \varphi_{\text{act}}^* \rangle
\]  

(5.1)

As a standard notation, the \( L^2 \) inner product is defined as \( \langle \varphi_{\text{act}}^*, \psi_{\text{act}}^* \rangle = \int_{\Omega} \varphi_{\text{act}}^* \psi_{\text{act}}^* \, dy \). It has been shown [Sirovich] that when the number of degrees of freedom required to describe \( x_i \) is larger than the number of snapshots \( N \) (always true for infinite dimensional systems), it is sufficient to express the basis functions as linear combinations of the snapshots as

\[
\varphi_{\text{act}} = \sum_{j=1}^{N} w_j x_{j,\text{act}}
\]

(5.2)

Here, the coefficients \( w_j \) are to be determined such that \( \varphi \) maximizes (5.1). This approach then consists of the following steps:

- Construct an eigenvalue problem

\[
C_{N \times N} W_{N \times N} = \lambda W_{N \times N}
\]

where \( C = \left[ c_{ij} \right]_{N \times N} \), \( c_{ij} = \frac{1}{N} \int_{\Omega} x_i(y) x_j(y) \, dy \) 

(5.3)

- Obtain \( N \) eigenvalues and corresponding eigenvectors of the \( C \) matrix. Note that matrix \( C \) is symmetric and hence its eigenvalues are real. Also, it has been shown that all eigenvalues of \( C \) are non-negative ["Ravindran"].

8
• Sort the eigenvalues of C in a descending order \( \chi_1 \geq \chi_2 \geq \cdots \geq \chi_N \geq 0 \). Let the corresponding eigenvectors be \( W^1 = \begin{bmatrix} w_1^1 & \cdots & w_N^1 \end{bmatrix}^T \), \( W^2 = \begin{bmatrix} w_1^2 & \cdots & w_N^2 \end{bmatrix}^T \), \cdots, \( W^N = \begin{bmatrix} w_1^N & \cdots & w_N^N \end{bmatrix}^T \). A property of these eigenvectors is that they are mutually orthogonal.

• Normalize the eigenvectors to satisfy
\[
\langle W^i, W^j \rangle = (W^i)^T W^j = \frac{1}{N \lambda_i}
\]

This will ensure that the POD basis functions are orthonormal.

• Cut-off the eigenspectrum judiciously, so that the truncated system with \( \tilde{N} \leq N \) eigenvalues will satisfy \( \sum_{j=1}^{\tilde{N}} \lambda_j = \sum_{j=1}^{N} \lambda_j \). Usually, it turns out that \( \tilde{N} \ll N \).

• Finally, construct the \( \tilde{N} \) basis functions as
\[
\varphi_i(y) = \sum_{j=1}^{\tilde{N}} w_j^i x_j(y)
\]
\[
\vdots
\]
\[
\varphi_{\tilde{N}}(y) = \sum_{j=1}^{\tilde{N}} w_j^\tilde{N} x_j(y)
\]

An interested reader can refer to [Holmes, Ly, Ravindran] for details about this procedure.

### 5.3 Lumped Parameter Problem

The reduction of the infinite-dimensional PDE-driven problem to a finite set of ordinary differential equations and a related cost function are explained in this subsection. After obtaining the basis functions \( \varphi(y) \), \( x(t,y) \) and \( u(t,y) \) are expanded as follows.
\[
x(t,y) = \sum_{j=1}^{\tilde{N}} \hat{x}_j(t) \varphi_j(y)
\]
\[
u(t,y) = \sum_{j=1}^{\tilde{N}} \hat{u}_j(t) \varphi_j(y)
\]

One can notice that both \( x(t,y) \) and \( u(t,y) \) are characterized by the same basis functions. This implies that a state feedback controller spans a subspace of the state variables and hence, the basis functions for the state are assumed to be capable of spanning the controller as well. By substituting these expansions of state and controller variables in (4.5), we get
\[
\sum_{j=1}^{N} \dot{x}_j \varphi_j = A_i \sum_{j=1}^{N} \dot{x}_j \varphi_j^* + F \left( \sum_{j=1}^{N} x_j \varphi_j \right) + \sum_{j=1}^{N} \dot{u}_j \varphi_j \tag{5.7}
\]

Note that, \( F \left( \sum_{j=1}^{N} x_j \varphi_j \right) \) has a linear term and other non-linear terms, hence \( F \left( \sum_{j=1}^{N} x_j \varphi_j \right) \) can be written as:

\[
F \left( \sum_{j=1}^{N} x_j \varphi_j, X_d \right) = A_i \left( X_d \right) \sum_{j=1}^{N} x_j \varphi_j + f \left( \sum_{j=1}^{N} x_j \varphi_j, X_d \right) \quad \text{where} \quad f \left( \sum_{j=1}^{N} x_j \varphi_j, X_d \right)
\]

is defined as:

\[
A_i \left( X_d \right) \left( \sum_{j=1}^{N} x_j \varphi_j \right) + A_i \left( X_d \right) \left( \sum_{j=1}^{N} x_j \varphi_j \right)^2 + A_i \left( X_d \right) \left( \sum_{j=1}^{N} x_j \varphi_j \right)^3 + A_i \left( X_d \right) \left( \sum_{j=1}^{N} x_j \varphi_j \right)^4
\]

Next, taking the Galerkin projection of (5.7) on the basis function \( \varphi_i(.) \) (i.e. taking the inner product with respect to \( \varphi_i \)), and using the fact that the basis functions are orthonormal, yields (5.8)

\[
\dot{x}_j = A_i \sum_{j=1}^{N} \langle \varphi_j, \varphi_j^* \rangle \dot{x}_j + A_i \left( X_d \right) \dot{x}_j + \langle f \left( x \right), \varphi_i \rangle + \dot{u}_i \tag{5.8}
\]

Repeating this exercise for \( i = 1, \ldots, N \) and arranging the equations in order, leads to a set of ordinary differential equations of the form

\[
\dot{\tilde{X}} = \tilde{A} \tilde{X} + \tilde{f} \left( \tilde{X} \right) + B \tilde{U} \tag{5.9}
\]

where \( \tilde{X} \triangleq \left[ \tilde{x}_1, \ldots, \tilde{x}_N \right]^T \) and \( \tilde{U} \triangleq \left[ \tilde{u}_1, \ldots, \tilde{u}_N \right]^T \). Other symbols are defined as follows:

\[
\tilde{A} \triangleq A_i \left[ a_{ij} \right] + A_i \left( X_d \right) I_N
\]

\[
a_{ij} \triangleq \langle \varphi_j, \varphi_j^* \rangle = \int_{0}^{1} \varphi_j \varphi_j^* \, dy = \left[ \varphi_j \varphi_j^* \right]_{y=0}^{y=1} - \int_{0}^{1} \varphi_j \varphi_j^* \, dy
\]

\[
\tilde{f} \left( \tilde{X} \right) \triangleq \langle f \left( x \right), \varphi_i \rangle = \int_{0}^{1} f \left( x \right) \varphi_i \, dy
\]

\[
B \triangleq I_N
\]

Next, the terms in the cost function (4.7) that contain \( x \) and \( u \) should be written in terms of \( \tilde{x} \) and \( \tilde{u} \).

\[
\int_{0}^{1} q \tilde{x} \, dy = q \langle x, x \rangle = q \left( \sum_{j=1}^{N} \tilde{x}_j \varphi_j \right) \left( \sum_{j=1}^{N} \tilde{x}_j \varphi_j \right) = q \sum_{j=1}^{N} \tilde{x}_j \tilde{x}_j = \tilde{X}^T Q \tilde{X} \tag{5.11}
\]

where \( Q = q I_N \) and

\[
\int_{0}^{1} r \tilde{u} \, dy = r \langle u, u \rangle = r \left( \sum_{j=1}^{N} \tilde{u}_j \varphi_j \right) \left( \sum_{j=1}^{N} \tilde{u}_j \varphi_j \right) = r \sum_{j=1}^{N} \tilde{u}_j \tilde{u}_j = \tilde{U}^T R \tilde{U} \tag{5.12}
\]

where \( R = r I_N \). Thus the cost function in (4.7) in terms of the finite dimensional states becomes
From (5.9) and (5.13) an analogous optimal control problem in the lumped parameter framework can be defined. This problem is solved next using the neural networks in an ADP framework.

6. APPROXIMATE DYNAMIC PROGRAMMING (ADP)

In this section, the general discussion on the optimal control of the distributed parameter systems is presented in an ADP framework. Detailed derivations of these conditions can also be found in [Balakrishnan, 1996] and [Werbos, 1992] and are repeated here for the sake of clarity and completeness. Development in this section will subsequently be used in synthesizing the neural networks for optimal cooling control of the fin.

A. Problem Description and Optimality Conditions

Assume a scalar cost function, to be minimized, of the form:

\[ J = \sum_{k=0}^{N} \Psi_k(\dot{X}_k, \dot{U}_k) \]  

(6.1)

where \( \dot{X}_k \) and \( \dot{U}_k \) represent the \( n \times 1 \) state vector and \( m \times 1 \) control vector respectively at time step \( k \). \( N \) represents the number of discrete time steps. \( \Psi_k(\dot{X}_k, \dot{U}_k) \) is a nonlinear cost function at step \( k \) that represents the concerns of the control system designer. Note that when \( N \) is large, (6.1) represents the cost function for an infinite horizon problem. Following the above representation of the cost function, we denote the cost function from time step \( k \) as

\[ J_k = \sum_{k=0}^{N} \Psi_k(\dot{X}_k, \dot{U}_k) \]  

(6.2)

A recursive relationship for the cost function is obtained for the cost from \( k \) in terms of the cost from \( (k+1) \), \( J_{k+1} \) and \( \Psi_k \) the cost to go from \( k \) to \( (k+1) \) (called the utility function) as

\[ J_k = \Psi_k + J_{k+1} \]  

(6.3)

The \( n \times 1 \) costate vector at time step \( k \) is defined as

\[ \lambda_k = \frac{\partial J_k}{\partial \dot{X}_k} \]  

(6.4)

Then the necessary condition for optimality is

\[ \frac{\partial J_k}{\partial \dot{U}_k} = 0 \]  

(6.5)
However,

\[
\frac{\partial J}{\partial U_i} = \left( \frac{\partial \Psi}{\partial U_i} \right) + \left( \frac{\partial J_{i+1}}{\partial U_i} \right) = \left( \frac{\partial \Psi}{\partial U_i} \right) + \left( \frac{\partial \hat{X}_{i+1}}{\partial U_i} \right)^\top \lambda_{i+1}
\]

(6.6)

Therefore, the optimality condition reduces to

\[
\begin{bmatrix} \frac{\partial \Psi}{\partial U_i} \ + \ \frac{\partial \hat{X}_{i+1}}{\partial U_i} \end{bmatrix}^\top \lambda_{i+1} = 0
\]

(6.7)

The co-state propagation equation can be derived in the following way.

\[
\begin{align*}
\lambda_k &= \frac{\partial J_k}{\partial X_k} = \left( \frac{\partial \Psi_k}{\partial X_k} \right) + \left( \frac{\partial J_{k+1}}{\partial X_k} \right) \\
&= \left[ \left( \frac{\partial \Psi_k}{\partial X_k} \right) + \left( \frac{\partial \hat{X}_{k+1}}{\partial X_k} \right)^\top \lambda_{k+1} \right] + \left[ \left( \frac{\partial \Psi_k}{\partial U_i} \right) + \left( \frac{\partial \hat{X}_{k+1}}{\partial U_i} \right)^\top \lambda_{k+1} \right]
\end{align*}
\]

(6.8)

Equation (6.8) evaluated along the optimal path given by (6.7) simplifies to

\[
\lambda_k = \left( \frac{\partial \Psi_k}{\partial \hat{X}_k} \right) + \left( \frac{\partial \hat{X}_{k+1}}{\partial \hat{X}_k} \right)^\top \lambda_{k+1}
\]

(6.9)

**Optimality Equations for the Fin Problem**

Temperature propagation equation across the fin can be written in the following discrete form as

\[
\hat{X}_{s_i} = F(\hat{X}_i) + B\hat{U}_i
\]

(6.10)

We notice that a discrete equivalent of the cost function in (5.13) can be written as

\[
J = \sum_{i=1}^{(N-1)\infty} \left( \hat{X}_i^\top Q_0 \hat{X}_i + \hat{U}_i^\top R_0 \hat{U}_i \right)
\]

(6.11)

where \( Q_0 = Q \Delta t \) and \( R_0 = R \Delta t \). We also have

\[
\Psi = \hat{X}_i^\top Q_{0d} \hat{X}_i + \hat{U}_i^\top R_{0d} \hat{U}_i
\]

(6.12)

Then, equations for optimal control and costate can be written as

\[
\hat{U}_i = -R_{0d}^{-1} \hat{B}^\top \lambda_{s i+1}
\]

(6.13)

\[
\lambda_{s i} = G(\hat{X}_i, \hat{U}_i, \lambda_{s i+1})
\]

(6.14)
It should be noted that explicit forms of the functions $F$ and $G$ depend on the type of discretization procedure. With the availability of relationships in (6.10), (6.13) and (6.14), we can proceed to synthesize a neurocontroller as discussed in the following sections.

### 6.2 Single Network Adaptive Critic (SNAC)

Typically, ADP based problems are solved by using two networks in a dual heuristic programming formulation (DHP): one network to capture the relationship between the states and the control at stage $k$ and a second network to capture the relationship between the states and the costates at stage $k$. In contrast, the SNAC captures the relationship between the states at $k$ and the costates at $(k+1)$.

Even though the relationship between $\hat{X}_k$ and $\hat{\lambda}_{k+1}$ can be captured in a single network, in this work the network is split internally into $N$ sub-networks, assuming one network (rather one sub-network) for each element of the costate vector. The input to each sub-network, however, is the entire state vector $\hat{X}_k$.

Having a separate network for each element of the costates has been found to speed up the training process since cross coupling of weights for different components of the output vector are absent.

Choosing specific architectures for the sub-networks mostly relies on experience and intuition. Use small networks may not be adequate to capture the nonlinearities whereas large networks may lead to slower training. In this study, five feedforward $\pi_{5,5,1}$ neural networks are used. A $\pi_{5,5,1}$ neural network implies that the network has five neurons in the input layer, five neurons in the hidden layer and one neuron in the output layer. For activation functions, a tangent sigmoid function is used for all the hidden layers and a linear function is used for the output layer.

### 6.3 State Generation for Neural Network Training

Note that the lumped parameter states can be computed from $x(y)$ as shown by (6.15).

$$\hat{x}_i = \langle x(y), \varphi_i(y) \rangle \quad (6.15)$$

Let $\hat{X}_{\text{max}}$ and $\hat{X}_{\text{min}}$ denote the vectors of maximum and minimum values of the elements of $\hat{X}_k$ respectively. Note that for values close to zero, the effect of non-linear term is negligible and systems behaves close to linear dynamics. Our offline training process ensures that effect of non-linear terms come in slowly as the training set is expanded. Implementation is carried out by training in a smaller domain and increasing it to accommodate the entire range. Let $0 \leq C_i \leq 1$. The initial training set is obtained by setting $C_i = 0.05$ and generating training points in $S_i \triangleq [C_i \hat{X}_{\text{min}}, C_i \hat{X}_{\text{max}}]$. Once the network
is trained in this set, \( C_i \) is changed as \( C_i = C_i + 0.05(i-1) \) for \( i = 2, 3, \ldots \) and the network is trained again. This process is repeated until \( C_i = 1 \).

### 6.4 Training of Neural Networks

Neural network training in this paper proceeds along the following steps: (Figure 2).

1. Fix \( C_i \) and generate \( S'_i \) (as described in Section 6.2).
2. For each element \( \hat{X}_k \) of \( S'_i \) follow the steps below.
   - Input \( \hat{X}_k \) to the networks to get \( \lambda_{k+1}^a \). Let us denote it as \( \lambda_{k+1}^a \).
   - Calculate \( \hat{U}_k \), knowing \( \hat{X}_k \) and \( \lambda_{k+1}^a \), from optimal control equation (6.13).
   - Get \( \hat{X}_{k+1} \) from the state equation (6.10), using \( \hat{X}_k \) and \( \hat{U}_k \).
   - Input \( \hat{X}_{k+1} \) to the networks to get \( \lambda_{k+2}^a \).
   - Calculate \( \lambda_{k+1}^a \), form the costate equation (6.9). Let us denote this as \( \lambda_{k+1}^c \).
3. Train the networks, with all \( \hat{X}_k \) as input and all corresponding \( \lambda_{k+1}^c \) as output.
4. If proper convergence is achieved, stop and revert to step 1, with \( S'_{i+1} \). If not, go to step 1 and retrain the networks with a new \( S'_i \).
For faster convergence, a convex combination \[ \theta \lambda_{i+1}^j + (1-\theta) \lambda_{k+1}^j \], \(0 < \theta < 1\) as the target output for training, is used. Moreover, to minimize the chance of getting trapped in a local minimum, the \textit{batch training} philosophy is followed, where the network is trained for all of the elements of \(S_i^c\) simultaneously. The Levenberg-Marquardt method [Hagan] is used for training. For each \(S_i^c\) 2000 input-output data points are chosen. After training the networks with 2000 data points for 25 epochs, the networks are checked for convergence (see Section 6.5) with another 2000 different data points. If the convergence condition is met, the networks are trained again with a different set of 2000 data points in \(S_{i+1}^c\) and so on. Otherwise, the training process is repeated by generating another set of random data in \(S_i^c\).

6.5 Convergence Check

Before changing \(C_i\) to \(C_{i+1}\) and generating new profiles for further training, it should be assured that proper convergence is achieved for \(C_i\). For this purpose a training set is generated in \(S_i^c\) and is used as described below.

1. Fix a tolerance value (In this study, \(tol = 0.1\)).
2. By using the profiles from \(S_i^c\), generate the target outputs, as described in Section 6.2. Let the outputs be \(\lambda_1^i, \lambda_2^i, \ldots, \lambda_N^i\).
3. Generate the actual output from the networks, by simulating the \textit{trained} networks with the profiles from \(S_k^c\). The values of the outputs are \(\lambda_1^u, \lambda_2^u, \ldots, \lambda_N^u\).

Check whether simultaneously \(\|\lambda_j^i - \lambda_j^u\|_2 < tol\), \(\forall j = 1, 2, \ldots, N\). If yes, it can be said that the networks have converged.

6.6 Implementation of the Control Solution

After the network controller is synthesized offline, it is implemented as a feed back controller as shown in Figure 3.
Note that the SNAC controller generates the control that will take the current fin state from any temperature profile (within the compact set used) to the desired temperature profile \( X_d \).

### 7. Robust Control Design Using On-Line Neural Networks

Accurate thermal modeling of a high speed aerospace vehicle is very difficult. It is imperative that any thermal controller is robust to uncertainties due to modeling errors or parameter variations. In this section, an extra control scheme to compensate for unmodeled dynamics is developed.

#### 7.1 Problem formulation and Uncertainty description

Let the true model be given by

\[
\frac{\partial x(t, y)}{\partial t} = f \left( x(t, y), \frac{\partial x(t, y)}{\partial y}, \frac{\partial^2 x(t, y)}{\partial^2 y}, \ldots \right) + \beta u(t, y) + D(x(t, y), y) \tag{7.1}
\]

where \( D(x(t, y), y) \) represents the bounded uncertainty not captured by the nominal model. The goal is to find an extra control that can offset the effects of this uncertainty and help perform close to nominal system behavior.

Note that the uncertainty can be expanded as explained in section 5.3 as follows:

\[
D(x(t, y), y) = \sum_{i=1}^{\hat{s}} \hat{d}_i(t) \varphi_i(y) \tag{7.2}
\]

By using (7.2) and taking inner product of (7.1) with the basis functions, a reduced order model for the true plant is obtained as

\[
\hat{X} = \hat{F}(\hat{X}) + \hat{B}\hat{U} + \hat{D}(\hat{X}) \tag{7.3}
\]

where
\[
\dot{\hat{X}}(t) \triangleq \left\{ f\left(x(t, y), \frac{\partial x(t, y)}{\partial y}, \frac{\partial^2 x(t, y)}{\partial y^2}, \cdots \right), \varphi \right\}
\]
\[
B \triangleq \mathbf{I}_\xi
\]
\[
\hat{D}(\hat{X}) = \langle D(x(t, y), \varphi) \rangle = \hat{a}(t)
\]

### 7.2 Uncertainty Modeling with Online Network and Weight Updates

This section describes how the system uncertainty is modeled through a neural network. The key idea is to capture the unmodeled dynamics using a neural network, the weights of which are updated online. In order to ensure convergence, the weight update scheme is so chosen as to yield bounded weight estimates.

Note that the output of a neural network can be written as \( W'\phi(\hat{X}) \) where \( W \) is a matrix of weights and \( \phi(\hat{X}) \) is a vector of basis functions. The basis functions are chosen in such a way that \( \lvert \phi(\hat{X}) \rvert \leq M \). It is known that within a compact set of state \( \hat{X} \) there exists a matrix of weights and a vector of basis functions that can approximate the uncertainty \( \hat{D}(\hat{X}) \) to any desired accuracy i.e.

\[
\hat{D}(\hat{X}) = W_{\text{ideal}}'\phi(\hat{X}) + \varepsilon \quad \text{(7.4)}
\]

where \( W_{\text{ideal}} \) is the matrix of ideal weights and \( \varepsilon \) is the approximation error of the neural network and for any positive number \( \varepsilon_N \), there exists a neural network such that \( \lvert \varepsilon \rvert \leq \varepsilon_N \).

The chosen basis functions that form the online network are \( \left[ 1 \quad \sin(\hat{X}_i) \quad \sin(2\hat{X}_i) \quad \cdots \quad \sin(20\hat{X}_i) \right] \) for all \( i \). Fourier series is used because of its boundedness, orthogonality and good non-linear function approximation capabilities. Twenty terms of the Fourier series are found to be sufficient for this application. For this neural network structure, the weight update rule is presented next.

Let \( \hat{U}_{\text{opt}} \) denote the control generated by the adaptive critic network controller for the nominal system

\[
\dot{\hat{X}} = \hat{F}(\hat{X}) + B\hat{U} \quad \text{(7.5)}
\]

Let \( \hat{U}_{\text{ex}} \) denote the extra control being applied to compensate for uncertainty in the model. The total control applied in is \( \hat{U} = \hat{U}_{\text{opt}} + \hat{U}_{\text{ex}} \).

By substituting the above expression for \( \hat{U} \) in the actual plant model in (7.3) leads to

\[
\dot{\hat{X}} = \hat{F}(\hat{X}) + B(\hat{U}_{\text{opt}} + \hat{U}_{\text{ex}}) + \hat{D}(\hat{X}) \quad \text{(7.6)}
\]

By choosing \( \hat{U}_{\text{ex}} \) as \(-B'W'\phi(\hat{X})\), the uncertainty is shown to be compensated for. By substituting for \( \hat{U}_{\text{ex}} \) and \( \hat{D}(\hat{X}) \) from (7.4), equation (7.6) becomes

\[
\dot{\hat{X}} = \hat{F}(\hat{X}) + B\hat{U}_{\text{opt}} - W'\phi(\hat{X}) + W_{\text{ideal}}'\phi(\hat{X}) + \varepsilon \quad \text{(7.7)}
\]

Now, an ‘approximate’ system mimicking the nominal system is defined as follows:
\[
\dot{\hat{X}}_a = \hat{F}(\hat{X}) + B\hat{U}_{opt} + K(\hat{X} - \hat{\hat{X}}_a)
\]

where \( K \) is of the form \( K = kI \) (\( k \) is scalar and \( I \) is the identity matrix).

This (observer-type) system (7.8) is introduced to get an approximation of the error between the states of (7.5) and (7.7). The extra term \( K(\hat{\hat{X}}_a - \hat{X}_a) \) is introduced to make \( \) reduce the error between its states and (7.6) and when \( \hat{\hat{X}} \rightarrow \hat{X}_a \), (7.5) and (7.8) become identical. We prove this by choosing the online neural network weight update rule as,

\[
\dot{\hat{W}} = \Gamma^{-1}(\phi(\hat{X})\hat{e} - k\delta|\hat{e}|W)
\]

where \( \hat{e} \triangleq \hat{\hat{X}} - \hat{X}_a \). Note that the weights are guaranteed to be bounded and the bound on error between the approximate plant and the actual plant can be made as small as possible by choosing design parameters \( k \) and \( \delta \) (see Appendix). \( \Gamma \) is the learning rate of the network. The first term in (7.9) is used to realize a good approximation of the uncertainty in the model and the second term helps ensure the boundedness of the weights.

7.3 Application to High Speed Aerospace Vehicle Problem

It can be seen from (7.9) that the coefficients in the model are functions of the desired temperature profile. Since the control scheme was presented for the profile given by \( X_d \), for any other desired profile (say \( X_d^* \)), the equation for deviation of states \( X \) from \( X_d^* \) can be written in a form similar to (7.8) as,

\[
\frac{\partial x}{\partial t} = A_h\left(\frac{\partial x}{\partial y}\right) + F(x, X_d^*) + u
\]

where \( x = X - X_d^* \)

(7.10)

It can be seen that the optimal control scheme developed for \( X_d \) cannot be used directly to achieve \( X_d^* \). In order to achieve any desired profile, the SNAC network must be trained again. In order to use the same network trained to yield \( X_d \) to help achieve \( X_d^* \), add and subtract \( f(x, X_d) \) in (7.10) to get

\[
\frac{\partial x}{\partial t} = A_h\left(\frac{\partial x}{\partial y}\right) + F(x, X_d) + u + (F(x, X_d^*) - F(x, X_d))
\]

(7.11)

This manipulation allows us to treat the last term \( (F(x, X_d^*) - F(x, X_d)) \) as unmodeled dynamics. Now, we can rewrite (7.11) in a form similar to

\[
\frac{\partial x}{\partial t} = A_h\left(\frac{\partial x}{\partial y}\right) + F(x, X_d) + u + D(x)
\]

where \( D(x) = (F(x, X_d^*) - F(x, X_d)) \)

(7.12)

By using the online neural network approach to compensate for this unmodeled dynamics, any desired profile \( X_d^* \) can be achieved.
It should be pointed out that yet another type of important uncertainty in physics modeling can be accommodated in this framework. Viscous forces in high temperature are difficult to model accurately. The heat generated due to viscosity can be considered as an unmodeled uncertainty. The shear stress generated due to viscosity is given by [Anderson],

\[ \tau_v = \mu \frac{\partial V}{\partial z} \]  \hspace{1cm} (7.13)

where \( y \) is along and \( z \) is perpendicular to the motion of the vehicle (also fin surface), \( V \) is the velocity of the air along \( y \) and \( \mu \) is viscosity of air and varies linearly with temperature. The viscosity term is evaluated at the surface of the fin. The heat transfer equation with the effect of viscosity becomes

\[ \frac{\partial x}{\partial t} = A_h \left( \frac{\partial^2 x}{\partial y^2} \right) + F(x, X_u) + \frac{\mu V}{\rho c_r} \left( \frac{\partial V}{\partial y} \right)_{fin} + u \]  \hspace{1cm} (7.14)

The robust control scheme formulated in this paper can handle this type of uncertainty too.

7.4 Implementation of the Robust/Optimal Control

Figure 4 shows the control solution implementation scheme that compensates for model uncertainties and/or change of desired temperature profiles.

8. Numerical Results

Numerical simulations were performed in MATLAB. Five basis functions (\( \tilde{N} = 5 \)) were found to be adequate to describe the high temperature model for an aerospace vehicle. Simulation results are presented in three parts. First part shows the results from the SNAC design where the desired temperature profile was a constant temperature along the spatial direction. Second part presents the results from using the robust control scheme. In this study, this relates to achieving any desired temperature profile from a set of initial profiles. The third section discusses results where the true plant has a viscosity related term
and a resultant heat addition over the nominal model. Values of the parameters used in the simulations are presented in Table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>Weight on deviation from desired state</td>
<td>1</td>
</tr>
<tr>
<td>$r$</td>
<td>Weight on control</td>
<td>1</td>
</tr>
<tr>
<td>$k$</td>
<td>Design parameter for online network</td>
<td>5</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Design parameter for online network</td>
<td>0.1</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>Learning rate of online network</td>
<td>$0.75 I_{100}$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Coefficient of Viscosity</td>
<td>$1.2 \times 10^{-7} - 1.7 \times 10^{-5}$</td>
</tr>
<tr>
<td>$V_{\frac{\partial V_{y_k}}{\partial z_k}}$</td>
<td>Velocity gradient term in heat equation</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2. Parameter Definitions and Values

8.1 SNAC Controller
This section presents the results obtained by using the SNAC controller to achieve a constant temperature of 873 K across the fin. It can be seen from the three-dimensional temperature history presented in Figure 5 that the desired final temperature profile is reached.

Figure 5. Temperature Profile

Figure 6 presents the control effort from the adaptive critic controller on the reduced order side transformed to the original side. Note that this additional control is used (on the reduced-order side) to take the deviations (again on the reduced-order side) to zero. The steady state control is not added to the computed control. The finite-dimensional model is used to compute the additional amount of control needed to drive the error between the desired profile and the current profile to zero; then the basis
functions are used to transform this control effort to a true control and applied to the original model. As to be expected, this history decays to zero as in Figure 6.

![Graph](image)

Figure 6. Control Output of the Network

### 8.2 Online Neural Network to Reach Any Desired Final Profile

The method proposed in section 7.3 is tested in a case to obtain an exponential profile. If the SNAC controller is used alone to drive the error between the desired state and the final state to zero then the desired temperature is not achieved because the SNAC was designed to obtain a single specific desired profile although it can achieve it from any starting temperature profile. However, with the use of the online neural network for compensation of unmodeled dynamics any desired profile could be achieved. The parameters used for the simulations of this network are $k = 5$ and $\delta = .01$.

Figure 7 presents the temperature history in the case of desired exponential profile. Note that the temperature error history (from the desired profile) is presented in Figure 8 and it goes to zero with time. Figure 9 shows the desired temperature profile along the fin and the actual temperature at steady state. Note that they are almost identical. Control history, a sum of SNAC output and extra control, is presented in Figure 10.
Figure 7. Temperature Profile Variation with Time

Figure 8. Error between Actual Temperature and the Desired Temperature
8.3 Online Neural Network (Robustness to Viscosity Effects)

This section presents the simulation results where the controller design is shown to be robust to unmodeled viscous forces. In the first case, the SNAC is designed as in the first section to produce a constant temperature across the fin using a model without the viscous terms. The online neural network is used to compensate for the viscous effects. It can be seen from Figure 11 that the desired final temperature is achieved when there is online compensation but it is quite different without it. Figure 12
shows that the final temperature achieved with the online network and without it. The end results are quite different.

Figure 11. Temperature variation with Time with and without uncertainty compensation

Figure 12. Final temperatures with and without uncertainty compensation

9. Conclusions

In this study, approximate dynamic programming based formulations were used to synthesize suboptimal neurocontrollers for high speed aerospace vehicles traversing through the atmosphere.. An adaptive critic based SNAC controller was shown to be able used to drive any given initial temperature profile to a desired profile. In order to compensate for the effect of uncertainty and to use the same adaptive critic network for achieving any desired profile, an online neural network was used. The weight update rule proposed in this paper ensures boundedness of the weight estimates and hence relaxes the persistence of
excitation condition. It was shown in simulation studies that the proposed robust control scheme can achieve any desired end profile and can compensate for viscous effects that are difficult to model (unmodeled dynamics). The technique developed in this paper is implementable. Simulation results demonstrate that the proposed technique has excellent promise and could be very useful for a variety of applications since the formulation uses very few assumptions in its development.

Acknowledgement

This research was supported by NSF grant 0324428. We gratefully acknowledge the heat transfer related discussions with Dr. D.C. Look, Emeritus professor, University of Missouri-Rolla. Thanks are also due to the anonymous reviewers whose detailed and specific comments have made this paper better in many ways.

References


In this section it is shown the weight update scheme (7.9) drives the error between actual and approximate plant to zero and also guarantees that the weights are bounded. First, an expression for the error dynamics is obtained. Next, a Lyapunov function based analysis is used to obtain an upper bound on magnitude of norms of error and show that the weight matrix of the network is bounded. As the weights are bounded, the need to satisfy the persistence of excitation condition is relaxed. Variables and parameters used in the proof are listed in Table A1. All the norms used in the proof are 2-norms.

### Appendix

In this section it is shown the weight update scheme (7.9) drives the error between actual and approximate plant to zero and also guarantees that the weights are bounded. First, an expression for the error dynamics is obtained. Next, a Lyapunov function based analysis is used to obtain an upper bound on magnitude of norms of error and show that the weight matrix of the network is bounded. As the weights are bounded, the need to satisfy the persistence of excitation condition is relaxed. Variables and parameters used in the proof are listed in Table A1. All the norms used in the proof are 2-norms.

<table>
<thead>
<tr>
<th>Variable/Parameter</th>
<th>Description</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{e}$</td>
<td>Error between actual plant and approximate plant</td>
<td>$\hat{N} \times 1$</td>
</tr>
<tr>
<td>$K = k\hat{\gamma}$</td>
<td>A design parameter</td>
<td>$K = \hat{N} \times \hat{N} k = 1 \times 1$</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>Online network’s approximation error</td>
<td>$\hat{N} \times 1$</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td>Dimensions</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>$W_{\text{ideal}}$</td>
<td>Ideal matrix of weights</td>
<td>$M \times \tilde{N}$</td>
</tr>
<tr>
<td>$W$</td>
<td>Actual matrix of weights</td>
<td>$M \times \tilde{N}$</td>
</tr>
<tr>
<td>$\hat{W}$</td>
<td>Difference between ideal and actual weights</td>
<td>$M \times \tilde{N}$</td>
</tr>
<tr>
<td>$\phi(\hat{X})$</td>
<td>Vector of basis functions</td>
<td>$M \times 1$</td>
</tr>
<tr>
<td>$D(\mathbf{x}(t, y), y)$</td>
<td>Actual uncertainty in the PDE system</td>
<td>scalar</td>
</tr>
<tr>
<td>$\hat{D}(\hat{X})$</td>
<td>Actual uncertainty in the ODE system</td>
<td>$M \times 1$</td>
</tr>
<tr>
<td>$B_D$</td>
<td>Upper bound on $</td>
<td>\hat{D}(\hat{X})</td>
</tr>
<tr>
<td>$\epsilon_{\tilde{N}}$</td>
<td>Upper bound on $</td>
<td>\epsilon</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>Learning rate of online network</td>
<td>$M \times M$</td>
</tr>
<tr>
<td>$\delta$</td>
<td>A design parameter to control error bound</td>
<td>scalar</td>
</tr>
<tr>
<td>$V$</td>
<td>Lyapunov function candidate</td>
<td>scalar</td>
</tr>
<tr>
<td>$E$</td>
<td>Error between actual uncertainty and network approximation</td>
<td>$\tilde{N} \times 1$</td>
</tr>
</tbody>
</table>

**Table A1. Descriptions of Variables and Parameters**

By defining $\hat{\epsilon} \triangleq \hat{X} - \tilde{X}_a$ and $\hat{W} \triangleq W_{\text{ideal}} - W$, equations for the error and the weight update are

$$\hat{e} = -K\hat{e} + \hat{W}'\phi(\hat{X}) + \epsilon \quad (A.1)$$

$$\hat{W} = \Gamma^{-1}\left(\phi(\hat{X})\hat{e}' - k\delta|\hat{e}|W\right)$$

where $\hat{e} \triangleq \hat{X} - \tilde{X}_a \quad (A.2)$

By defining a Lyapunov function candidate as

$$V = \frac{1}{2}\hat{e}'\hat{e} + \frac{1}{2}\text{Tr}(\hat{W}'\Gamma\hat{W}) \quad (A.3)$$

and taking time derivative of (A.3), we get

$$\dot{V} = \hat{e}'\hat{e} + \text{Tr}(\hat{W}'\Gamma\dot{W}) \quad (A.4)$$

By substituting error equation in (A.4),

$$\dot{V} = \hat{e}'\left(-K\hat{e} + \hat{W}'\phi(\hat{X}) + \epsilon\right) + \text{Tr}(\hat{W}'\Gamma\dot{W})$$

$$= -\hat{e}'K\hat{e} + \hat{e}'\phi(\hat{X}) + \hat{e}'\epsilon + \text{Tr}(\hat{W}'\Gamma\dot{W}) \quad (A.5)$$

Rewrite $\hat{e}'\hat{W}'\phi(\hat{X})$ in (A.5) in a trace form to get

$$\dot{V} = -\hat{e}'K\hat{e} + \text{Tr} \left(\hat{W}'\phi(\hat{X})\hat{e}'\right) + \hat{e}'\epsilon + \text{Tr} \left(\hat{W}'\Gamma\dot{W}\right)$$

$$= -\hat{e}'K\hat{e} + \hat{e}'\epsilon + \text{Tr} \left(\hat{W}'\left(\phi(\hat{X})\hat{e}' + \Gamma\dot{W}\right)\right) \quad (A.6)$$

Since $\hat{W} = W_{\text{ideal}} - W$, $\dot{\hat{W}} = -\hat{W}$, (A.6) can be rewritten as

$$\dot{V} = -\hat{e}'K\hat{e} + \hat{e}'\epsilon + \text{Tr} \left(\hat{W}'\left(\phi(\hat{X})\hat{e}' - \Gamma\dot{W}\right)\right) \quad (A.7)$$
By using the weight update equation, the trace term in is modified to get
\[
\dot{V} = -\dot{\epsilon}' K\dot{\epsilon} + \dot{\epsilon}' \epsilon + Tr \left( \tilde{W}' k \delta |\dot{\epsilon}| W \right)
\]
\[
= -\dot{\epsilon}' K\dot{\epsilon} + \dot{\epsilon}' \epsilon + Tr \left( (W_{\text{ideal}} - W)' k \delta |\dot{\epsilon}| W \right)
\]
\[
= -\dot{\epsilon}' K\dot{\epsilon} + \dot{\epsilon}' \epsilon + k \delta |\dot{\epsilon}| Tr \left( W_{\text{ideal}}' W \right) - k \delta |\dot{\epsilon}| Tr \left( W' W \right)
\]
(A.8)

Note that \( \dot{\epsilon}' K\dot{\epsilon} = k |\dot{\epsilon}|^2 \) and \( Tr \left( W' W \right) = \|W\|_F^2 \). Therefore, (A.8) becomes
\[
\dot{V} = -k |\dot{\epsilon}|^2 + \dot{\epsilon}' \epsilon + k \delta |\dot{\epsilon}| Tr \left( W_{\text{ideal}}' W \right) - k \delta |\dot{\epsilon}|\|W\|_F^2
\]
(A.9)

The third term in has a summation representation given by
\[
Tr \left( W_{\text{ideal}}' W \right) = \sum_{i=1}^{n} \sum_{j=1}^{m} W_{\text{ideal}}^{ij} W^{ij}
\]
(A.10)

It can be proved that
\[
\sum_{i=1}^{n} \sum_{j=1}^{m} W_{\text{ideal}}^{ij} W^{ij} \leq \sqrt{\left( \sum_{i=1}^{n} \sum_{j=1}^{m} (W_{\text{ideal}}^{ij})^2 \right) \left( \sum_{i=1}^{n} \sum_{j=1}^{m} (W^{ij})^2 \right)} = \|W_{\text{ideal}}\|_F \|W\|_F \leq W_{\text{max}} \|W\|_F
\]
(A.11)

Hence,
\[
\dot{V} \leq -k |\dot{\epsilon}|^2 + \dot{\epsilon}' \epsilon + k \delta |\dot{\epsilon}| \табular{\|W\|_F \& \|W\|_F \leq W_{\text{max}} \|W\|_F (A.12)}
\]

Upper Bounds on \( |\dot{\epsilon}| \) and \( \|W\|_F \)

i. Upper bound on \( |\dot{\epsilon}| \).

In order to establish an upper bound on \( |\dot{\epsilon}| \), the square on \( \|W\|_F \) needs to be completed in,
\[
\dot{V} \leq -k |\dot{\epsilon}|^2 - k \delta |\dot{\epsilon}| \left( \frac{|\dot{\epsilon}|}{k \delta} - W_{\text{max}} \|W\|_F + \|W\|_F \right)
\]
(A.13)
\[
\dot{V} \leq -k |\dot{\epsilon}|^2 - k \delta |\dot{\epsilon}| \left( \frac{|\dot{\epsilon}|}{k \delta} + \left( \|W\|_F - \frac{W_{\text{max}}^2}{2} \right)^2 - \frac{W_{\text{max}}^2}{4} \right)
\]
(A.14)
\[
\dot{V} \leq -k |\dot{\epsilon}| \left( |\dot{\epsilon}| - \frac{|\dot{\epsilon}|}{k} - \frac{W_{\text{max}}^2}{4} \right) - k \delta |\dot{\epsilon}| \left( \|W\|_F - \frac{W_{\text{max}}^2}{2} \right)^2
\]
(A.15)

Equation (A.15) implies that
\[
|\dot{\epsilon}| \geq \frac{|\dot{\epsilon}|}{k} + \delta \frac{W_{\text{max}}^2}{4}, \quad \dot{V} \leq 0
\]
(A.16)

Therefore, \( |\dot{\epsilon}| \) is upper bounded although the bound is conservative. However, this bound can be made as small as desired with a proper choice of the design parameters \( k \) and \( \delta \).
ii. Upper bound on $\|W\|_F$.

To establish an upper bound on $\|W\|_F$, the square term with $|\hat{e}|$ is completed.

$$\dot{V} \leq -k|\hat{e}|^2 - k\delta |\hat{e}|\left(\|W\|_F^2 - W_{\max}\|W\|_F - \frac{|e|}{k\delta}\right)$$

(A.17)

The Lyapunov function is decreasing if the expression inside $[\bullet]$ in (A.17) is positive. That is, if

$$\|W\|_F^2 \geq \frac{W_{\max} + \sqrt{W_{\max}^2 + 4\frac{|e|}{k\delta}}}{2}$$

(A.18)

Therefore, if

$$\|W\|_F^2 \geq \frac{W_{\max} + \sqrt{W_{\max}^2 + 4\frac{|e|}{k\delta}}}{2}, \dot{V} \leq 0$$

(A.19)

then from (A.16) and (A.18), it can be concluded that $|\hat{e}|$ and $\|\hat{W}\|_L$ are uniformly upper bounded.