M. A. Sartori and P. J. Antsaklis, "Neural Network Implementations for Control Scheduling," Technical Report #91-04-02, Dept. of Electrical Engineering, University of Notre Dame, April 1991.

NEURAL NETWORK IMPLEMENTATIONS FOR CONTROL SCHEDULING

Technical Report #91-04-02
Department of Electrical Engineering
University of Notre Dame
April 1991

Michael A. Sartori and Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

M. A. Sartori and P. J. Antsaklis, "Neural Network Implementations for Control Scheduling," Technical Report #91-04-02, Dept. of Electrical Engineering, University of Notre Dame, April 1991.

NEURAL NETWORK IMPLEMENTATIONS FOR CONTROL SCHEDULING

Michael A. Sartori and Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

Two different neural network implementations for control scheduling are presented: one with gaussian nonlinearities and the other with sigmoid nonlinearities. For the implementations discussed here, the neural network is supplied information about existing operating conditions and then responds by supplying control law parameter values to the controller. For each design, the neural network has two layers of weights, and the values of the weights and biases are based on given operating points for the scheduler. By designing the neural network's generalization behavior, specifications for the interpolation between the given operating points are satisfied. The gaussian neural network performs best when the operating points are equidistant and has some drawbacks when used to implement multi-parameter schedulers. The sigmoid neural network performs equally well for both equidistant and non-equidistant operating points and overcomes the gaussian neural network's difficulties of the multi-parameter case.

As discussed in [Antsaklis92, Sartori91], a neural network may be used as a control law scheduler. In this capacity, the neural network can be viewed as a high level decision maker operating outside the conventional control loop to provide a higher degree of autonomy to the system. Given a set of operating points and the associated set of parameters values for the control law, the neural network is designed to satisfy given specifications for the interpolation between the provided operating points. Here, these requirements for the interpolation between operating points are treated as specifications for the generalization behavior of the neural network, a topic of importance in current neural network research. Two types of neural networks are presented here for satisfying the specifications for the control law scheduler: one using gaussian nonlinearities and a second using gaussian-type nonlinearities formed via the difference of sigmoid nonlinearities. The gaussian neural network performs best when the operating points are equidistant and has some

disadvantages when used to implement multi-parameter schedulers. The sigmoid neural network performs equally well for both equidistant and non-equidistant operating points and overcomes the gaussian neural network's difficulties with the multi-parameter case.

Of the numerous curves that satisfy the specifications for the interpolation between the operating points, the two neural network design procedures presented in this paper represent two possible classes. Clearly, there exist other methods besides the ones presented here to (mathematically) describe the curve. For instance, the specific interpolation curve may be modelled with polynomials or splines. The advantage of using the neural network approaches described here instead of one of these schemes is that, in addition to straight forward design schemes, the actual construction of the neural network in hardware would utilize the inherent parallelism of the neural network and hence result in a fast processing time.

For the neural network scheduler implementation described in this paper, the neural network is first given information on the system's operating conditions or its environment and then supplies control law switching information to the controller. As depicted in Figure 1, the neural network's inputs are the inputs and outputs of the plant together with the reference signal. The output of the neural network is the control law adjustment signal sent to the controller. The neural network's inputs are not restricted to these signals; other signals such as the plant's states, derivatives, environmental conditions, or delayed values of any of these can be used as inputs to the neural network. Basically, any signal that is designed into the operation of the scheduler is used as an input to the neural network. Furthermore, the plant and controller can operate in either continuous or discrete time. If the neural network is implemented in analog hardware, both the plant and the controller can operate in continuous time. However, if the neural network is to be implemented in software, both the plant and the controller need to be discrete or discretized versions of continuous ones. In either case, the designing of the neural network as discussed in this paper remains unchanged.

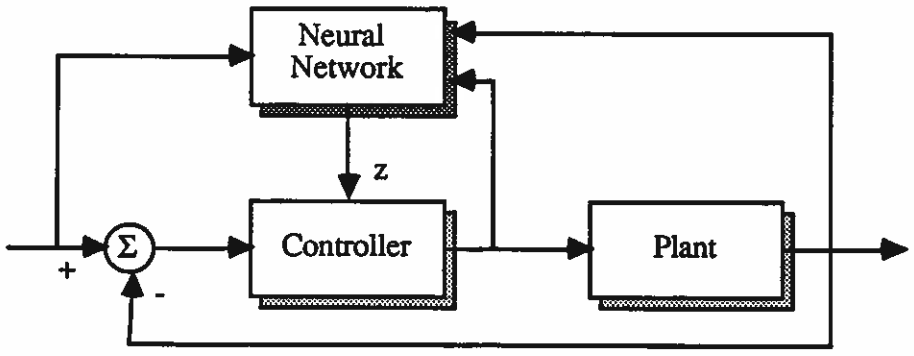

Figure 1 Neural network used as a scheduler.

In Section 1, the type of scheduler implementation considered in this paper is defined, and the implementation using a gaussian neural network is presented for both single-parameter and multi-parameter operating points and discussed in terms of equidistant and non-equidistant operating points. In Section 2, the implementation of a scheduler with a sigmoid neural network is considered and also described in terms of schedulers with single-parameter and multi-parameter operating points. Finally, several examples demonstrating the two neural network scheduling implementations presented in this paper are supplied in Section 3. In particular, three scheduling problems are solved using both of the neural network implementations, and the results are compared.

1 IMPLEMENTATION USING A GAUSSIAN NEURAL NETWORK

The gaussian neural network implementation of a scheduler is divided into two cases: single-parameter operating points and multi-parameter operating points. The single-parameter case is further divided into equidistant operating points and non-equidistant operating points.

1.1 Single-Parameter Gaussian Neural Network Scheduler

In this section, a neural network is constructed to implement a scheduler with operating points defined for a single parameter. For each operating point, the scheduler provides one or more parameter values to be used in controlling the system. For the given values of the scheduler, let $v(j)$ denote the j^{th} operating point for $1 \le j \le p$, and let $\mathbf{v} \in \mathbb{R}^{p\times 1}$ denote the vector of all operating points arranged in ascending order, where $v(j) > v(j-1)$ for $2 \ge j \ge p$. Let $d_i(j)$ denote the i^{th} desired output (i.e., the given control law parameter value) of the scheduler for the j^{th} operating point for $1 \le i \le n$ and $1 \le j \le p$, and let $\mathbf{d}(j) \in \mathbb{R}^{n\times 1}$ and $\mathbf{D} \in \mathbb{R}^{p\times n}$ denote the vector for $1 \le j \le p$ and the matrix of desired outputs, respectively. Thus, the p pairs $\{v(j), \mathbf{d}(j)\}$ are the given operating points. For the neural network, let $u \in \mathbb{R}^{1\times 1}$ denote the neural network's input, and let the neural network have multiple outputs denoted by $\mathbf{z} \in \mathbb{R}^{n\times 1}$. For a specific input u, let $z_i(u)$ for $1 \le i \le n$ and $\mathbf{z}(u)$ denote the output of the neural network.

In constructing the neural network to implement the designed scheduler, three specifications are made:

(i) If $u = v(j)$, then $\mathbf{z}(u) = \mathbf{d}(j)$.

(ii) If $u \in [v(j) - \varepsilon_{j-}, v(j) + \varepsilon_{j+}]$ and $u \neq v(j)$, then $z_i(u) \in [d_i(j) - \gamma_{ij-}, d_i(j) + \gamma_{ij+}]$.

(iii) If $u \in [v(j), v(j+1)]$, then $z_i(u) \in [d_i(j), d_i(j+1)]$.

As an example, Figure 2 illustrates one way in which these specifications can be viewed. The dots correspond to the operating points and the given controller parameter, and according to specification (i), the interpolation curve must pass through these points. The boxes surrounding the dots correspond to the boxes described in specification (ii), and the boxes between the dots

correspond to the bounding of the interpolation curve per specification (iii). The interpolation curve must pass correctly through both sets of boxes. Clearly, there exist numerous curves that satisfy these three specifications. With the design scheme described here for the parameters of the neural network, a particular class of curves is achieved that satisfy these specifications, and of the three specifications, (i) and (ii) can be satisfied precisely using the proposed procedure, and (iii) can be approximately satisfied.

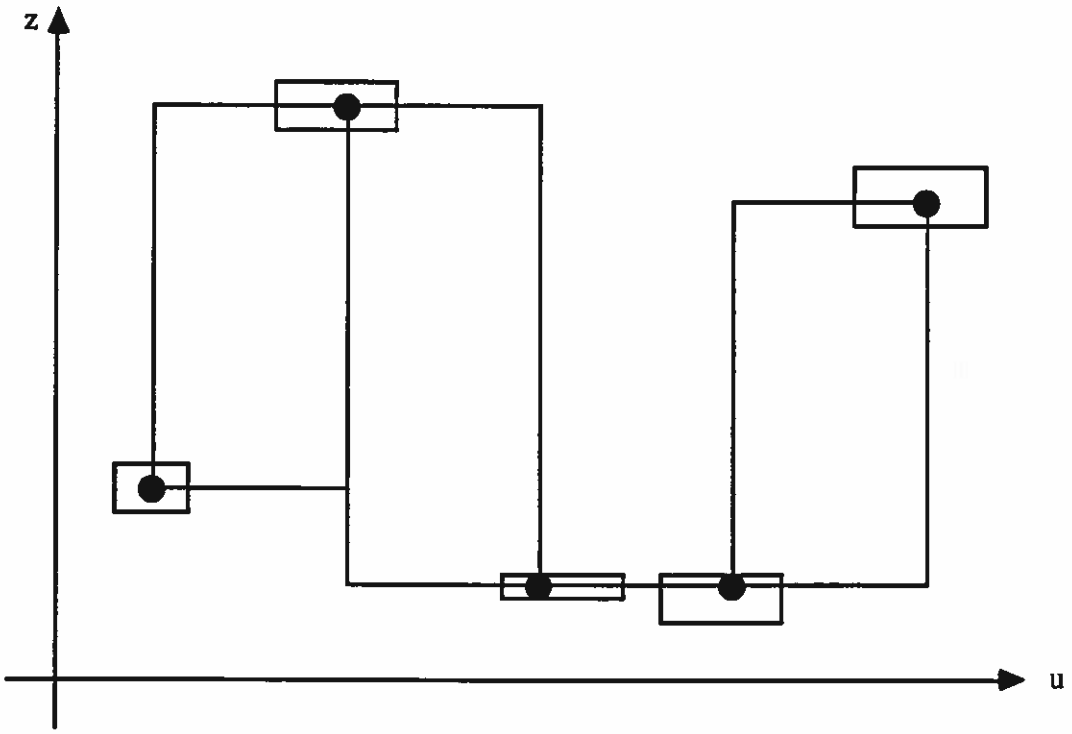

Figure 2 Illustration of interpolation curve specifications.

With $z \in \mathbb{R}^{nx1}$, an individual output of the neural network is described by:

$$z_i = \sum_{k=1}^{h} w_{ik}\, g(x_k) \tag{1}$$

where $1 \leq i \leq n$, w_{ik} is a weight of the i^{th} linear neuron in the output layer, and $g(x_k)$ is the output of the k^{th} neuron in the hidden layer. Let $W \in \mathbb{R}^{hxn}$ denote the matrix of weights for the output layer. Since the scheduler has a single parameter describing its operating points, the neural network requires only a single input $u \in \mathbb{R}^{1x1}$. The output of the k^{th} gaussian neuron in the neural network's hidden layer is described by:

$$g(x_k) = \exp(-x_k^{\alpha}) \tag{2}$$

and

$$x_k = s_k(v - c_k)^2 \tag{3}$$

where $1 \leq k \leq h$, $g:\mathbb{R} \rightarrow \mathbb{R}$ is the nonlinearity of the hidden layer neurons and the gaussian function with the variable α describing its curvature, s_k is the weight (or "width" of the gaussian function), and c_k is the bias (or "center" of the gaussian function). Let $\mathbf{s} \in \mathbb{R}^{hx1}$ denote the vector of widths, and $\mathbf{c} \in \mathbb{R}^{hx1}$ denote the vector of centers. With $\alpha = 1$, the traditional bell-shaped curve is achieved, but by allowing α to vary, a class of curves is possible, which allows for greater flexibility in the design of the interpolation curve. Thus, for a particular neural network implementation of a scheduler, the parameters **W**, **s**, **c**, and α need to be selected. The choosing of the gaussian neural network parameters is divided into two cases: when the operating points are equidistant and when they are non-equidistant.

1.1.1 Design with Equidistant Operating Points

For the first case, the operating points v(j) for $1 \leq j \leq p$ are considered to be equidistant. For the p operating points, assume the distance between each operating point is T. The number of hidden layer gaussian neurons is set equal to the number of operating points, that is $h = p$. This choice is not unreasonable since for many scheduler applications the number of operating points is not unusually large. The center c_k for the k^{th} neuron is set equal to the operating point value for the k^{th} operating point:

$$c_k = v(k) \tag{4}$$

for $1 \leq k \leq h$. Since c_k corresponds to the center of the gaussian nonlinearity, setting c_k equal to the operating point parameter corresponds to placing the center of each gaussian neuron at each operating point and allows for a localized effect at the output of these neurons with the appropriate choice for each width s_k.

To aid in satisfying specifications (ii) and (iii), the widths **s** are chosen such that (1) can be approximated by

$$z_i(u) \cong \sum_{k=j}^{j+1} w_{ik}\, g(x_k) \tag{5}$$

when $u \in [v(j), v(j+1)]$. This implies that the tails of the gaussian nonlinearities $g(x_k(u))$ for $k \neq j$ and $k \neq j+1$ are small compared to those for $k = j$ and $k = j+1$ when $u \in [v(j), v(j+1)]$. The approximation of (5) also implies that the gaussian neuron with its center c_k closest to the input u has a larger response compared to the other gaussian neurons signifying that the input is closest to the k^{th} operating point :

$$g(x_k(u)) \geq g(x_f(u)) \tag{6}$$

for $1 \leq f \leq h$ and for $u \in [v(k) - \Delta_{k-1}, v(k) + \Delta_k]$ where Δ_k is defined such that $g(x_k(v(k) + \Delta_k)) = g(x_{k+1}(v(k) + \Delta_k))$. With (6), the localized effect of the hidden layer neurons is preserved. Also, the interpolation curve passing through a box specified by specification (ii) assumes the general

shape of the output of the k^{th} gaussian neuron if $\varepsilon_{k-} \leq \Delta_{k-1}$ and $\varepsilon_{k+} \leq \Delta_k$. To further aid in satisfying specification (iii), the widths s are chosen such that

$$\sum_{k=1}^{h} g(x_k(u)) \cong 1 \tag{7}$$

for $u \in [v(1), v(p)]$. In conjunction with (5) and (6), (7) implies that for $u \in [v(k), v(k+1)]$ the sum of the outputs of $g(x_k(u))$ and $g(x_{k+1}(u))$ is constant, and the k^{th} gaussian neuron contributes more to the sum when u is close to v(k) and less when it is closer to v(k+1). Since all the operating points are spaced apart by an equal distance T, the width s_k for each neuron is selected equivalent such that each neuron has an equal response halfway between any two neurons. By setting $g(x_k)$ in (2) equal to 0.5,

$$s_k = \frac{4g^{-1}(0.5)}{T^2} \tag{8}$$

where $1 \leq k \leq h$ and $g^{-1}(r) := (-\ln(r))^{1/\alpha}$. Unfortunately, due to the nonlinear nature of the gaussian function of (2), using (8) to determine s does not insure that (5), (6), and (7) are satisfied exactly, but rather are satisfied approximately; with (8), a good approximation to a constant unit signal is achieved as described by (7), and through experimentation, a possibly more appealing signal may be found by slightly adjusting the widths.

Due to the use of the gaussian nonlinearity and the choice of function's center and width for each of the hidden layer neurons, the output of the neural network is close to 0 if the input is outside the region of the operating points. In other words, if $u << v(1)$ or $u >> v(p)$, then $z \cong 0$. This is a drawback if a value is needed from the scheduler for all possible inputs. However, this behavior could be used in a fault detection scheme to identify when the inputs have exceeded the design domain.

The choice of the exponent α in (2) clearly affects the shape of the interpolation curve. In Figure 3, $g(x_k) = \exp(-x_k^{\alpha})$ is shown for various values of α with $c_k = 0$ and $s_k = 1$. As can be seen, the gaussian curve begins to better approximate a unit pulse function as α is increased. By appropriately choosing α, the interpolation curve can be designed to pass though the box defined in specification (ii). The exponent α also affects how well (7) is satisfied and the choice of the widths s in (8). It is suggested here that in general $\alpha \geq 1$. Furthermore, unless stated otherwise, it is assumed that each neuron's nonlinearity in the hidden layer has the same value for the exponent α. Although, by choosing different values of α for different neurons, various interpolation curves are possible.

With the widths, centers, and the nonlinearity specified for the hidden layer, the weights for the linear output layer are chosen next. Let $G \in \mathbb{R}^{pxh}$ denote the output of the hidden layer neurons for each of the p operating points. Thus, the output layer weights are found by solving the following linear system of equations:

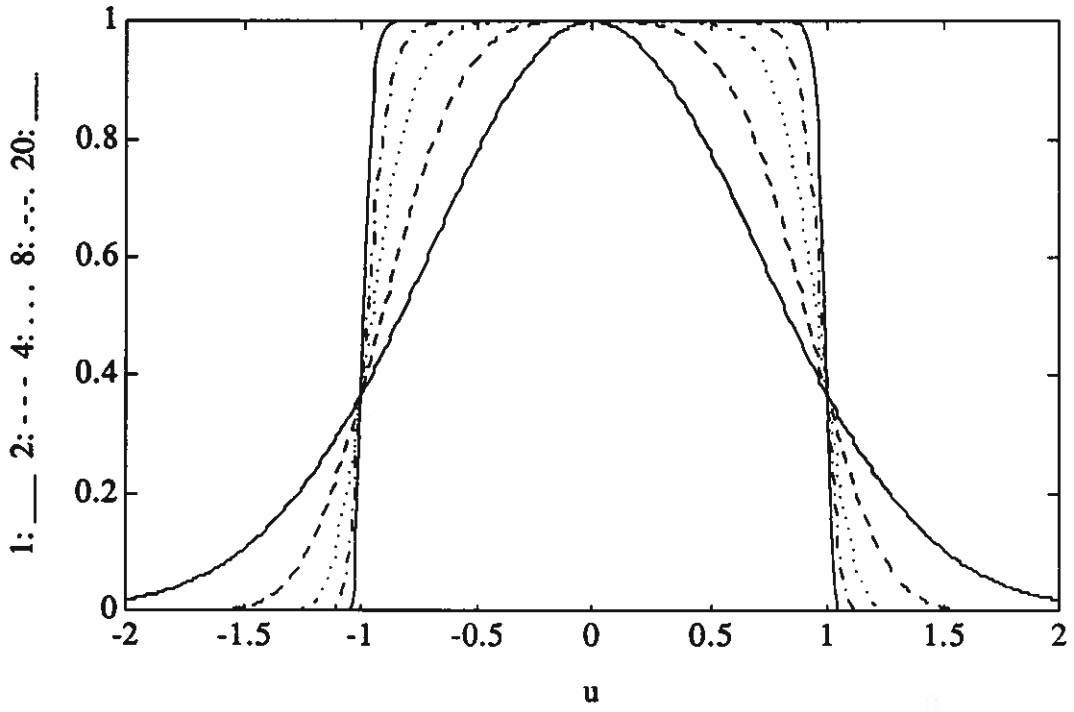

Figure 3 Family of gaussian-type curves for varied α.

$$\mathbf{GW} = \mathbf{D}. \tag{9}$$

Since **G** is square and nonsingular (h = p), (9) has a unique solution **W**. Thus, if the neural network's input u is exactly one of the operating points, the neural network's output is exactly the desired output of the scheduler, which satisfies specification (i). Furthermore, since s is chosen such that (5) and (6) are true, **G** is approximately the identity matrix, and **W** can be chosen as

$$\mathbf{W} \cong \mathbf{D}. \tag{10}$$

1.1.2 Design with Non-Equidistant Operating Points

For the second case, the operating points v(j) for $1 \leq j \leq p$ are considered to be non-equidistant. Two methods are proposed to cope with this situation: adding more gaussian neurons to the hidden layer or using an optimization procedure to adjust the parameters of the hidden layer neurons. It is assumed that there are p hidden layer neurons with centers **c** chosen as in (4), and that the weights **W** are chosen such that (9) is a unique system insuring that specification (i) is satisfied.

First, by adding more gaussian neurons to the hidden layer, a situation in which the centers of the gaussian neurons are equidistant occurs. This approach is useful if some of the operating points are equidistant, and the others occur at specific intervals between the operating points. First, p hidden layer gaussian neurons are formed with their centers equal to the operating points as

specified in (4). Next, more hidden layer gaussian neurons are added, and their centers are chosen such that the centers of the gaussian neurons are equidistant. For example, if

$$v = [1 \;\; 1.5 \;\; 2 \;\; 3 \;\; 4]'$$

is the vector of operating points, five hidden layer gaussian neurons are formed with $c_k = v(k)$. Two more gaussian neurons are added with centers of 2.5 and 3.5 such that $h = 7$. Since all of the centers of the gaussian neurons are equidistant and $T = 0.5$, the widths s are found via (8) with $\alpha = 2$, and $s_k = 13.3209$ for $1 \le k \le 7$. With the centers and the widths selected, the exponent α can be varied to further satisfy the specifications of the interpolation curve. The weights for the output layer are found by forming a linear system of equations similar to that of (9). To make the system unique, the desired output associated with an added gaussian neuron is assigned either a specified value of a close operating point or an interpolated value from nearby operating points.

Second, instead of adding neurons based on pseudo-operating points, an optimization procedure may be used to find the gaussian neural network's parameters. In this paper, a gradient descent procedure is proposed to cope with the case of non-equidistant operating points and to adjust the parameters of the hidden layer. The hidden layer is constructed of $h = p$ gaussian neurons with exponent α and with centers c equal to the operating points of the scheduler as described by (4). With the cost function

$$F = \frac{1}{2}\sum_{j=1}^{p}\left(1 - \sum_{k=1}^{h} g(x_k(v(j)))\right)^2 + \frac{1}{2}\sum_{j=1}^{q}\left(1 - \sum_{k=1}^{h} g(x_k(u(j)))\right)^2, \tag{11}$$

an approximate unit signal from the hidden layer as described by (7) is desired. The first sum corresponds to the neural network's output for inputs equal to the operating points. The second sum corresponds to the output of the neural network for q other values $u(j) \in [v(1), v(p)]$. The particular points in the second sum are not specified in order to leave this as a design consideration. To apply the gradient descent approach, the widths s first need to be initialized. One possibility is to initialize them to values similar to those described by (8):

$$s_k(0) = \frac{4g^{-1}(0.5)}{(v(k+1) - v(k))^2} \tag{12}$$

or

$$s_k(0) = \frac{4g^{-1}(0.5)}{(v(k) - v(k-1))^2}. \tag{13}$$

Since the operating points are assumed to be not equidistant, the values for $s_k(0)$ given by (12) and by (13) are presumably unequal. So, one, the other, or some combination can be used. To help minimize F in (11), the exponent α for each neuron can be adjusted independently such that α_k is the exponent for the k^{th} neuron in the layer.

With the centers, widths, and exponents initialized, the following gradient descent rules are used:

$$s_k(t+1) = s_k(t) - \eta_s \frac{\delta F}{\delta s_k} \tag{14}$$

$$c_k(t+1) = c_k(t) - \eta_c \frac{\delta F}{\delta c_k} \tag{15}$$

and

$$\alpha_k(t+1) = \alpha_k(t) - \eta_\alpha \frac{\delta F}{\delta \alpha_k} \tag{16}$$

where t is the iteration number and η_s, η_c, and η_α are the step sizes. Rewriting the cost function of (11) as

$$F = \sum_{j=1}^{p+q} F(j) = \frac{1}{2} \sum_{j=1}^{p+q} (1 - \sum_{k=1}^{h} g(x_k(u(j))))^2 \tag{17}$$

where $u(j) = v(j)$ for $1 \le j \le p$ and $u(j) \in [v(1), v(p)]$ for $p+1 \le j \le p+q$, the partial derivatives are

$$\frac{\delta F(j)}{\delta s_k} = \alpha\, g(x_k)\, (1 - \sum_{f=1}^{h} g(x_f))\, x_k^{\alpha-1}\, (u(j) - c_k)^2 \tag{18}$$

$$\frac{\delta F(j)}{\delta c_k} = -\, 2\alpha\, g(x_k)\, (1 - \sum_{f=1}^{h} g(x_f))\, x_k^{\alpha-1}\, (u(j) - c_k) s_k \tag{19}$$

and

$$\frac{\delta F(j)}{\delta \alpha_k} = g(x_k)\, (1 - \sum_{f=1}^{h} g(x_f))\, x_k^{\alpha_k}. \tag{20}$$

The gradient procedure can be terminated when the cost function is decreased sufficiently or when the output of the hidden layer is satisfactorily close to a unit signal. In practice, (14) can be modified to

$$s_k(t+1) = s_k(t) - \eta \frac{\delta F(j)}{\delta s_k} \tag{21}$$

where the gradient in (14) is approximated by the partial derivative of (18), and the input points u(j) are chosen randomly with an equal probability for each one. The same approximation can be performed for (15) and (16). This approach may speed convergence as well as help to preserve the local properties of the gaussian neurons in the hidden layer. Furthermore, not all of the parameters need to be modified. For instance, perhaps only the widths s need to be adjusted via the gradient descent procedure. Once the widths, centers, and exponents are found, the weights w_{ik} of the output layer are then uniquely determined per equation (9).

When the operating points are not equidistant, both of the methods proposed can also be used to determine the parameters of the hidden layer gaussian neurons. Extra neurons can first be added at desirable locations, and the gradient descent rules of (14) to (16) can then be used to find the appropriate parameters of the hidden layer. To find the weights w_{ik} of the output layer, extra

desired outputs are needed as described previously to insure that the linear system of equations described in (9) remains unique.

1.1.3 Comparison to Other Methods

Here, other methods proposed to determine the parameters of the gaussian neural network are discussed, and in particular, the design procedure of [Moody89] and [Moody88] is closely examined. In comparing the method proposed here and Moody's method, the following are the same: $h = p$, the centers are chosen as in (4), and the output weights as in (9). The only differences considered here are the choices of $\alpha = 1$ and different widths s.

The design procedure proposed by Moody is presented next. To choose the centers, Moody proposes to use the standard k-means clustering algorithm since it is assumed that the number of training pairs p is large, which is not the assumption made here for the scheduling problem. However, the case when the centers are chosen equal to the training inputs as in (4) is treated in Moody's examples. The exponent α of the gaussian nonlinearities is specified as 1. The widths s are determined by a "P nearest neighbor" heuristic, which in [Moody88] is described as using the root mean square value of the Euclidean distance to the P nearest neighbor centers to determine the widths. Using the notation here and $P = 2$,

$$s_k = \frac{2}{(c_k - c_{k-1})^2 + (c_{k+1} - c_k)^2} \tag{22}$$

for $1 \le k \le h$. The weights of the output layer are determined via a LMS algorithm. If $h = p$ and the centers are chosen as in (4), the LMS iterative procedure is not needed and the unique system of (9) can be solved instead. In comparing the method proposed here and Moody's method, the following are the same: $h = p$, the centers are chosen as in (4), and the output weights as in (9). The only differences considered here are $\alpha = 1$ and the widths chosen as in (22). Application of Moody's method for selecting the neural network's parameters are recorded in [Leonard90] and [Yao90], which both address the problem of fault detection and identification.

In [Moody89] and [Moody88], the use of gradient descent procedures to determine the neural network's centers, widths, and output layer weights are discussed. The cost function considered uses the sum of the squares of the errors:

$$F = \frac{1}{2} \sum_{j=1}^{p} (d(j) - z(v(j)))^2 \tag{23}$$

which can be compared to the one proposed in (11). In (23), the neural network is assumed to have only one output. The gradients of (23) with respect to the centers, widths, and weights are formed and used to update the centers, widths, and weights, respectively, such that (23) is minimized. However, as pointed out in [Moody89], the use of a gradient descent procedure to

determine the widths of the gaussian nonlinearities may reduce the local effect of the hidden layer neurons.

In [Poggio90], a practical algorithm is provided to determine the centers, widths, and weights of the neural network which involves a gradient descent procedure after initializing the neural network's parameters to specific values. Gradient descent procedures to determine the neural network's parameters are also discussed in [Huang89], which addresses the identification of firing patterns of neuronal signals, and in [Farrell90], which uses the neural network to control the depth of a submarine.

1.2 Multi-Parameter Gaussian Neural Network Scheduler

In this section, a neural network is constructed to implement a scheduler with operating points that vary for more than one parameter. For the given values of the scheduler, let $v_r(j)$ denote the r^{th} parameter of the scheduler for the j^{th} operating point for $1 \le r \le m$ and $1 \le j \le p$. Let $\mathbf{v}(j) \in \mathbb{R}^{mx1}$ denote the operating point vector for $1 \le j \le p$. As described previously, $\mathbf{d}(j) \in \mathbb{R}^{nx1}$ and $\mathbf{D} \in \mathbb{R}^{pxn}$ denote the vector for $1 \le j \le p$ and the matrix of desired outputs, respectively. Thus, the p pairs $\{\mathbf{v}(j), \mathbf{d}(j)\}$ are the given operating points. For the neural network, let $\mathbf{u} \in \mathbb{R}^{mx1}$ denote the neural network's input, and let the neural network have multiple outputs denoted by $\mathbf{z} \in \mathbb{R}^{nx1}$. For a specific input $\mathbf{u}$, let $z_i(\mathbf{u})$ for $1 \le i \le n$ and $\mathbf{z}(\mathbf{u})$ denote the output of the neural network.

The output of the gaussian neural network is given by:

$$z_i = \sum_{k=1}^{h} w_{ik}\, g(x_k) \tag{24}$$

where $1 \le i \le n$. The outputs of the hidden layer gaussian neurons are described by:

$$g(x_k) = \exp(-x_k^{\alpha}) \tag{25}$$

and

$$x_k = \sum_{r=1}^{m} s_{kr}(v_r - c_{kr})^2 \tag{26}$$

where $1 \le k \le h$. Let $\mathbf{S} \in \mathbb{R}^{hxm}$ denote the matrix of widths, and $\mathbf{C} \in \mathbb{R}^{hxm}$ denote the matrix of centers. Compared to the gaussian neurons proposed by [Moody89], the widths s_{kr} in (26) vary for the different r, which allows the gaussian nonlinearity to assume an ellipsoid shape for higher dimensions of operating point parameters and not just a circular shape.

In choosing the values for the centers, widths, and the gaussian variable α, it is once again desirable to satisfy the three design specifications stated in Section 1.1. In addition, the two cases considered previously for the spacing of the operating points are applicable here: when the operating points are equidistant and when they are non-equidistant. For the multi-parameter case, equidistance is considered for each individual direction and not as a norm over all the directions.

For the first case when the operating points are equidistant, the previous results are directly extended. A two layer neural network is formed with h = p hidden layer gaussian neurons and output linear neurons. The center c_{kr} for each gaussian neuron is set equal to the corresponding parameter for each operating point:

$$c_{kr} = v_r(k) \tag{27}$$

for $1 \le r \le m$ and $1 \le k \le h$. With T_r denoting the distance between the two operating points for the r^{th} parameter, the width s_{kr} for each neuron is given by:

$$s_{kr} = \frac{4g^{-1}(0.5)}{T_r} \tag{28}$$

for $1 \le r \le m$ and $1 \le k \le h$. The gaussian variable α is chosen to satisfy specification (ii), and the weights w_{ik} for the output layer are found by solving the unique linear system of equations in (9).

For the second case when the operating points are not equidistant, the results from Section 1.1.2 extend directly. If the operating points are well placed, extra gaussian neurons can be added to the hidden layer such that with the extra gaussian neurons the centers of all the gaussian neurons are equidistant. Also, similar to the approach described previously, an optimization procedure may be used to determine the widths, centers, and exponents α_k. In addition, both adding extra gaussian neurons and using an optimization procedure can be combined to satisfy the design specifications.

As the number of parameters for the operating points increases, one potential disadvantage of adding extra neurons is the possibility for a large number of neurons to result; if extra neurons are added for a multi-parameter scheduler of high dimension, the size of the neural network may become unwieldy. However, for the examples provided in the Section 3, this is not the case. Another drawback with this approach is that "valleys" develop in the interpolation curve due to the multi-dimensional ellipsoid described by (25) and (26). This is illustrated in Example 3.1.4.

2 IMPLEMENTATION USING A SIGMOID NEURAL NETWORK

Instead of using hidden layer neurons with gaussian nonlinearities to implement a scheduler as in Section 1, gaussian-type nodes formed by the difference of sigmoid nonlinearities are used in this section for the hidden layer neurons. Several of the drawbacks of the gaussian neural network implementation of the previous section are eliminated using the sigmoid neural network of this section. For a scheduler with single-parameter operating points, the operating points do not need to be equidistant for satisfactory results. For a designed scheduler with multi-dimensional operating points, the unwanted "valleys" are no longer prevalent, and the potentially large number of hidden layer nodes may be avoided. Furthermore, the specifications for the scheduler's interpolation curve are fully satisfied.

The sigmoid neural network implementation of a scheduler discussed in this section is divided into two parts: the single-parameter scheduler discussed in Section 2.1 and the multi-parameter scheduler described in Section 2.2.

2.1 Single-Parameter Sigmoid Neural Network Scheduler

The same notation as in Section 1.1 is used here to denote the scheduler's operating points and the neural network's inputs and outputs. In addition, the same three specifications as described in Section 1.1 for the interpolation curve need to be satisfied.

An individual output of the sigmoid neural network is described by:

$$z_i(u) = \sum_{k=1}^{h+1} w_{ik}\, g_k(u) \tag{29}$$

where $1 \le i \le n$, w_{ik} is a weight of the i^{th} linear neuron in the output layer, and g_k is considered to be the output of the k^{th} gaussian-type node in the hidden layer. Let $\mathbf{W} \in \mathbb{R}^{(h+1)xn}$ denote the matrix of weights for the output layer. The output of the k^{th} node in the neural network's hidden layer is described by:

$$g_k(u) = \sigma(u, c_{k-1}, s_{k-1}) - \sigma(u, c_k, s_k) \tag{30}$$

and

$$\sigma(u, c, s) = \frac{1}{1 + e^{-s(u - c)}} \tag{31}$$

where $1 \le k \le h+1$, $\sigma:\mathbb{R} \rightarrow \mathbb{R}$ is the sigmoid nonlinearity of the hidden layer neurons, c_k is the bias (or "center" of the sigmoid function), and s_k is the weight (or "slope" of the sigmoid function). Let

$$\sigma(u, c_0, s_0) = 1 \tag{32}$$

and

$$\sigma(u, c_{h+1}, s_{h+1}) = 0. \tag{33}$$

for all u. Let $\mathbf{c} \in \mathbb{R}^{hx1}$ denote the vector of centers, and $\mathbf{s} \in \mathbb{R}^{hx1}$ denote the vector of slopes. With $s_k = 1$, the conventional sigmoid curve is achieved, but by allowing s_k to vary, a class of sigmoid functions is possible, which allows for greater flexibility in the design of the interpolation curve. Thus, for a particular neural network implementation of a scheduler, the parameters $\mathbf{c}$, $\mathbf{s}$, and $\mathbf{W}$ need to be selected.

Due to the choice of $\sigma(u, c_0, s_0)$ and $\sigma(u, c_{h+1}, s_{h+1})$, when the neural network's input is outside the design domain, the neural network's output is equal to the designed scheduler's outputs associated with either the first or last operating points. In other words, if $u < v(1)$, $\mathbf{z} = \mathbf{d}(1)$, or if $u > v(p)$, then $\mathbf{z} = \mathbf{d}(p)$. If instead it is desired that if $u << v(1)$ or $u >> v(p)$, then $\mathbf{z} = \mathbf{0}$, the choice of $\sigma(u, c_0, s_0)$ and $\sigma(u, c_{h+1}, s_{h+1})$ can be modified to accomplish this.

2.1.1 Design with Single-Parameter Operating Points

The number of hidden layer gaussian nodes is set equal to one less the number of operating points, that is h = p - 1. The center c_k for the k^{th} node is set equal to the distance halfway between the k^{th} and the $(k+1)^{th}$ operating points:

$$c_k = \frac{1}{2}[v(k) + v(k+1)]. \tag{34}$$

for $1 \le k \le h$. Since $c_{k+1} \ge c_k$, g_k approximates a gaussian-type nonlinearity centered around v(k), and this allows for a localized effect at the output of the nodes g_k with the appropriate choice for each slope s_k.

To aid in satisfying specifications (ii) and (iii), statements similar to those in (5) - (7) are made. The widths s are chosen such that (29) can be approximated by

$$z_i(u) \cong \sum_{k=j}^{j+1} w_{ik}\, g_k(u) \tag{35}$$

when $u \in [v(j), v(j+1)]$. The approximation of (35) implies that when the node g_k has a larger response compared to the other nodes, the input is closest to the k^{th} operating point:

$$g_k(u) \ge g_f(u) \tag{36}$$

for $1 \le f \le h + 1$ and for $u \in [v(k) - \Delta_{k-1}, v(k) + \Delta_k]$ where Δ_k is defined such that $g_k(v(k) + \Delta_k) = g_{k+1}(v(k) + \Delta_k)$. To further aid in satisfying specifications (ii) and (iii), the slopes s are chosen such that

$$\sum_{k=1}^{h+1} g_k(u) = 1 \tag{37}$$

for all u. In conjunction with (35) and (36), (37) implies that for $u \in [v(k), v(k+1)]$ the sum of the outputs of $g_k(u)$ and $g_{k+1}(u)$ is constant, and the node g_k contributes more to the sum when u is close to v(k) and less when it is closer to v(k+1).

The choice of the slope s_k clearly affects the shape of the interpolation curve and the localization properties of gaussian-type nodes g_k. In Figure 4, $\sigma(u, 0, s)$ is shown for various values of s. As can be seen, by increasing s, the sigmoid function curve begins to better approximate a unit step function. As an example of the formation of a gaussian-type curve, Figure 5 shows the output of $g_k(u) = \sigma(u, -2, 5) - \sigma(u, 1, 20)$ which can be viewed as a gaussian-type curve centered at 0 with asymmetric sides. Furthermore, by appropriately choosing s_k as discussed below in Theorems 2.1 and 2.2, the interpolation curve can be designed to pass though the boxes defined in specification (ii).

With the centers and the slopes specified for the hidden layer, the weights for the linear output layer are chosen next. Let $G \in \mathbb{R}^{px(h+1)}$ denote the output of the hidden layer nodes for each of the p operating points. Thus, the output layer weights are found by solving the following linear system of equations:

$$\mathbf{GW} = \mathbf{D}. \tag{38}$$

Since h = p - 1, (38) describes a unique linear system of equations. Thus, if the neural network's input u is exactly one of the operating points, the neural network's output is exactly the desired output of the scheduler, which satisfies specification (i). Furthermore, due to the choice of s per (35) and (36), if the sigmoid neural network's input u is exactly the p^{th} operating point, the output of the p^{th} hidden layer gaussian-type node is 1 while the outputs of the other hidden layer gaussian-type nodes are 0. Hence, G can be assumed to be the identity matrix, and W can be approximated by

$$\mathbf{W} \cong \mathbf{D}. \tag{39}$$

With the next lemma, the choice of the slope s_k is related directly to specification (ii). Assume the centers c are chosen as in (34). Since s needs to satisfy (35) and (36), it is assumed W is found via (38), or (39). For simplicity, it is assumed that n = 1.

Lemma 2.1:

With $u \in [v(k) - \varepsilon_{k-}, v(k)]$ for $0 \le \varepsilon_{k-} < \Delta_{k-1}$, if $z(u) \in [d(k) - \gamma_{k-}, d(k) + \gamma_{k+}]$, then

$$s_{k-1} \ge \frac{-1}{v(k) - \varepsilon_{k-} - c_{k-1}} \ln\left[\min\left\{\frac{w_k - w_{k-1}}{d(k) - \gamma_{k-} - w_{k-1}}, \frac{w_k - w_{k-1}}{d(k) + \gamma_{k+} - w_{k-1}}\right\} - 1\right]. \tag{40}$$

Proof:

From (35),

$$z(u) = w_{k-1}g_{k-1}(u) + w_k g_k(u).$$

Since the sigmoid output $\sigma(u, c_{k-1}, s_{k-1})$ of node g_{k-1} is the same as the sigmoid output $\sigma(u, c_{k-1}, s_{k-1})$ of node g_k and due to the localized effect properties of these nodes as described by (35) and (36), the neural network output depends only on this sigmoid:

$$z(u) = w_{k-1}(1 - \sigma(u, c_{k-1}, s_{k-1})) + w_k\sigma(u, c_{k-1}, s_{k-1})) = w_{k-1} + \frac{w_k - w_{k-1}}{1 + e^{-s_{k-1}(u - c_{k-1})}}.$$

So,

$$\sigma(u, c_{k-1}, s_{k-1}) = \frac{z(u) - w_{k-1}}{w_k - w_{k-1}}.$$

Using $z(u) = d(k) - \gamma_{k-}$ and $z(u) = d(k) + \gamma_{k+}$, two bounds for $\sigma(u, c_{k-1}, s_{k-1})$ are found, and depending on the relationship between d(k) and d(k-1) (i.e, d(k) > d(k-1) or d(k) < d(k-1)) and due to the sharing of the sigmoids between g_{k-1} and g_k, one of these values will be larger than 1 and the other will be less than 1. Due to the nature of the sigmoid function, $0 < \sigma(u, c_{k-1}, s_{k-1}) < 1$, and one of the bounds is useless. Hence,

$$\sigma(u, c_{k-1}, s_{k-1}) \le \min\left\{\frac{d(k) - \gamma_{k-} - w_{k-1}}{w_k - w_{k-1}}, \frac{d(k) + \gamma_{k+} - w_{k-1}}{w_k - w_{k-1}}\right\}.$$

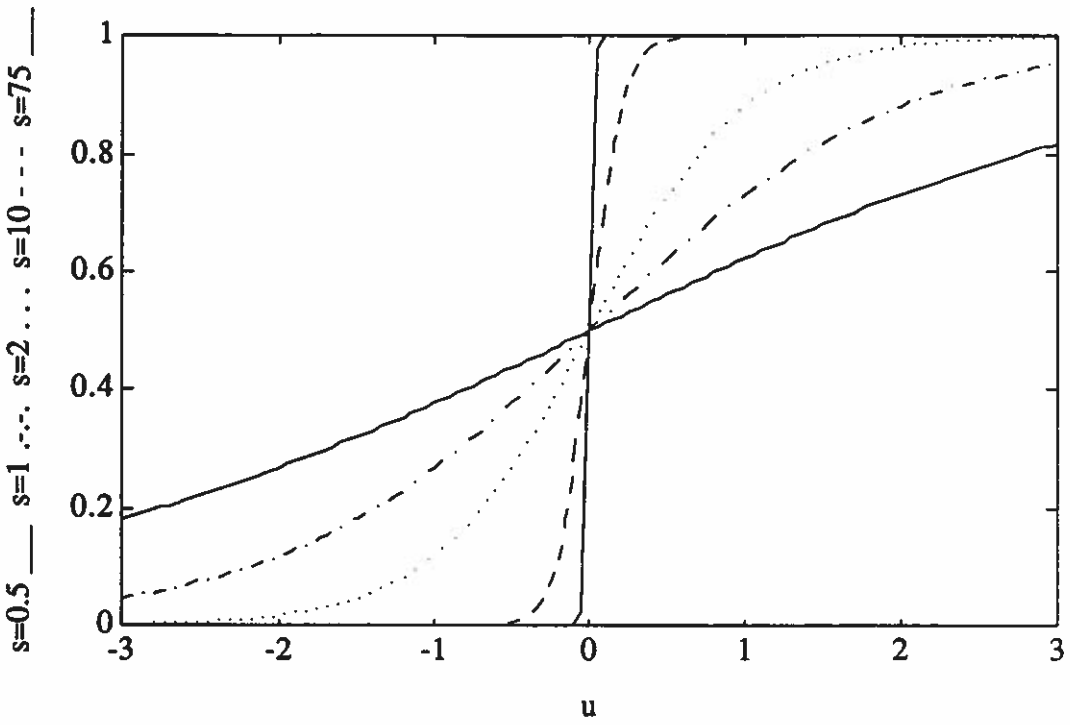

Figure 4 The sigmoid function $\sigma(u, 0, s)$ for various s.

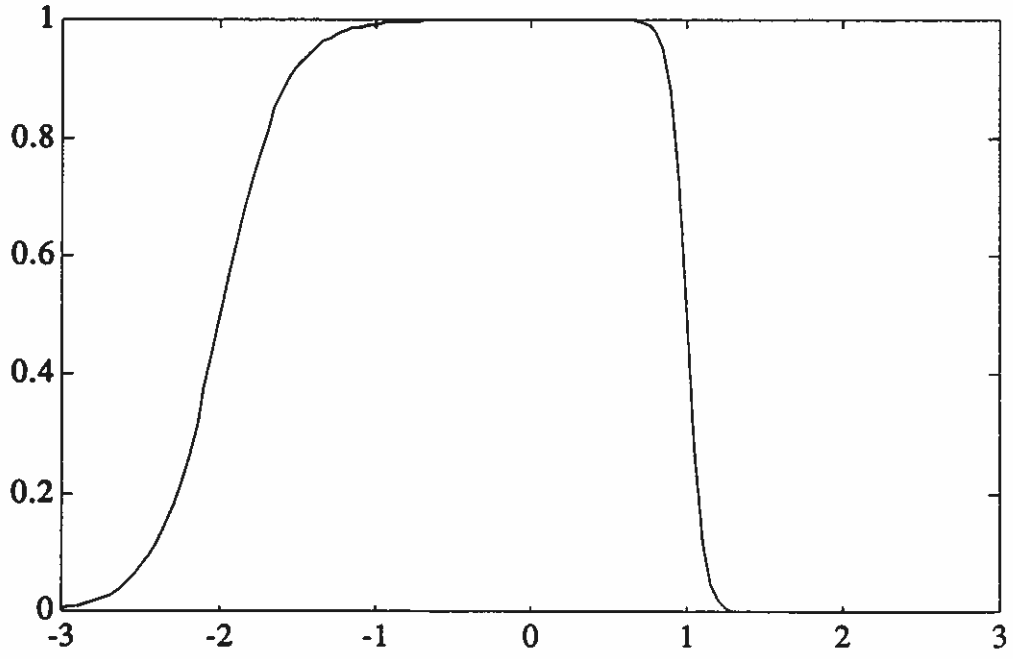

Figure 5 The gaussian-type function $g_k(u) = \sigma(u, -2, 5) - \sigma(u, 1, 20)$.

So,

$$e^{-s_{k-1}(u - c_{k-1})} \leq \min\left\{\frac{w_k - w_{k-1}}{d(k) - \gamma_{k-} - w_{k-1}}, \frac{w_k - w_{k-1}}{d(k) + \gamma_{k+} - w_{k-1}}\right\} - 1,$$

and

$$s_{k-1} \geq \frac{-1}{u - c_{k-1}} \ln\left[\min\left\{\frac{w_k - w_{k-1}}{d(k) - \gamma_{k-} - w_{k-1}}, \frac{w_k - w_{k-1}}{d(k) + \gamma_{k+} - w_{k-1}}\right\} - 1\right].$$

since $u - c_{k-1} > 0$ because $u \geq v(k) - \varepsilon_{k-} > c_{k-1}$. Choosing $u = v(k) - \varepsilon_{k-}$, the endpoint of specified region, (40) results. ♦

The next lemma addresses the case when $u \in [v(k), v(k) + \varepsilon_{k+}]$.

Lemma 2.2:

With $u \in [v(k), v(k) + \varepsilon_{k+}]$ for $0 \leq \varepsilon_{k+} \leq \Delta_k$, if $z(u) \in [d(k) - \gamma_{k-}, d(k) + \gamma_{k+}]$, then

$$s_k \geq \frac{-1}{v(k) + \varepsilon_{k+} - c_k} \ln\left[\max\left\{\frac{w_{k+1} - w_k}{d(k) - \gamma_{k-} - w_k}, \frac{w_{k+1} - w_k}{d(k) + \gamma_{k+} - w_k}\right\} - 1\right]. \qquad (41)$$

Proof:

Similar to the proof of Lemma 2.1. ♦

Thus, to satisfy specification (ii) for all operating points, the slope s_k needs to be chosen such that

$$s_k \geq \frac{-1}{v(k+1) - \varepsilon_{(k+1)-} - c_k} \ln\left[\min\left\{\frac{w_{k+1} - w_k}{d(k+1) - \gamma_{(k+1)-} - w_k}, \frac{w_{k+1} - w_k}{d(k+1) + \gamma_{(k+1)+} - w_k}\right\} - 1\right] \qquad (42)$$

and

$$s_k \geq \frac{-1}{v(k) + \varepsilon_{k+} - c_k} \ln\left[\max\left\{\frac{w_{k+1} - w_k}{d(k) - \gamma_{k-} - w_k}, \frac{w_{k+1} - w_k}{d(k) + \gamma_{k+} - w_k}\right\} - 1\right] \qquad (41)$$

are both met for $1 \leq k \leq h$, and if $n > 1$ for $1 \leq i \leq n$ as well. If $d(k) = d(k+1)$, then specification (iii) requires $z_i(u) = d(k) = d(k+1)$ for $u \in [v(k), v(k+1)]$, and the solving of (42) and (41) for $u = v(k+1) - \varepsilon_{(k+1)-}$ and $u = v(k) + \varepsilon_{k+}$ is unnecessary. To satisfy specification (iii) under this condition, the slope s_k is chosen such that (35), (36), and (37) are satisfied.

Due to the way in which the neural network parameters are chosen, the behavior of the neural network is expressed in equations (29) to (33). However, the neural network can be reconfigured in a more economical fashion, and equations (29) to (33) are equivalent to the following:

$$z_i(u) = \sum_{k=1}^{h-1} (w_{i,k+1} - w_{ik})\sigma(u, c_k, s_k) + w_{i1} - w_{ih}\sigma(u, c_h, s_h) \tag{43}$$

and

$$\sigma(u, c, s) = \frac{1}{1 + e^{-s(u - c)}} \tag{31}$$

where $1 \le i \le n$.

2.2 Multi-Parameter Sigmoid Neural Network Scheduler

The same notation as in Section 1.2 is used here to denote the scheduler's operating points and the neural network's inputs and outputs. In addition, the same three specifications as described in Section 1.1 for the interpolation curve need to be satisfied.

For the multi-parameter scheduler case, the output of the neural network is given by:

$$z_i(u) = \sum_{k_1=1}^{h_1+1} \cdots \sum_{k_m=1}^{h_m+1} w_{ik_1\ldots k_m}\, g_{1k_1}(u_1) \cdots g_{mk_m}(u_m) \tag{44}$$

where $1 \le i \le n$. For $m = 1$, (44) reduces to (29). The output of the rk_r^{th} hidden layer gaussian-type node is described by:

$$g_{rk_r}(u_r) = \sigma(u_r, c_{r(k_r-1)}, s_{r(k_r-1)}) - \sigma(u_r, c_{rk_r}, s_{rk_r}) \tag{45}$$

and

$$\sigma(u, c, s) = \frac{1}{1 + e^{-s(u - c)}} \tag{31}$$

where $1 \le k_r \le h_r + 1$ and $1 \le r \le m$. Let

$$\sigma(u_r, c_{r0}, s_{r0}) = 1 \tag{46}$$

and

$$\sigma(u_r, c_{r(h_r+1)}, s_{r(h_r+1)}) = 0 \tag{47}$$

for $1 \le r \le m$ and for all u_r. Let $c_r \in \mathbb{R}^{h_r x 1}$ denote the vector of centers, and $s_r \in \mathbb{R}^{h_r x 1}$ denote the vector of slopes for $1 \le r \le m$. Since the sizes of c_r and s_r are not restricted to be the same for $1 \le r \le m$, the formation of a matrix of centers and a matrix of slopes may not be possible. In (44), the outputs of the gaussian-type nodes are multiplied together, which is unusual when compared to conventional neural networks. With the multiplication, the specification of the centers and the slopes for the sigmoid neurons corresponds to the specification of hyper-rectangles around the operating points. By appropriately choosing these parameters, the entire input space can be mapped to particular desired outputs of the neural network scheduler, and no "valleys" develop between the gaussian-type nodes, as was the case in Section 1.2.

The selection of the neural network's centers, slopes, and weights for the multi-parameter scheduler is divided into the two cases of equidistant operating points and non-equidistant operating points with the case of equidistant operating points considered first. The term

"equidistant" for the m-dimensional case refers to the operating points being an equal distance apart in each dimension and is not defined as a norm over all the dimensions. Instead of denoting the operating points as v(j) for $1 \le j \le p$, the set of all equidistant operating points is considered to be comprised of elements from vectors containing information on all the values of all the operating points. Let p_r denote the number of values for the r^{th} dimension such that $p = \prod_{r=1}^{m} p_r$. Let $v_r = [v_{r1}, ..., v_{rp_r}]' \in \mathbb{R}^{p_r x 1}$ denote the vector of values of the operating points for the r^{th} dimension such that $v_{r(k_r+1)} \ge v_{rk_r}$ for $1 \le r \le m$ and $1 \le k_r \le p_r$. The counting of the p operating points is not important; rather, their location in the input space is important. The number of hidden layer gaussian-type nodes in the r^{th} dimension is set equal to one less the number of operating points, that is $h_r = p_r - 1$, for $1 \le r \le m$. The center c_{rk_r} is set equal to the distance halfway between the rk_r^{th} and the $r(k_r+1)^{th}$ operating points:

$$c_{rk_r} = \frac{1}{2}[v_{rk_r} + v_{r(k_r+1)}]. \tag{48}$$

for $1 \le r \le m$ and for $1 \le k_r \le h_r$. Since $c_{r(k_r+1)} \ge c_{rk_r}$ for $1 \le r \le m$, g_{rk_r} approximates a gaussian-type nonlinearity centered around v_{rk_r}, and this allows for a localized effect in the shape of a hyper-rectangle at the output of the node $\prod_{r=1}^{m} g_{rk_r}(u_r)$ with the appropriate choice for each slope s_{rk_r}.

In choosing the slopes, the multi-dimensional equivalents of (35) to (37) need to be satisfied. For (35), this implies that for any input u the output is dependent only on the closest gaussian-type nodes. For (36),

$$g_{rk_r}(u_r) \ge g_{rf_r}(u_r) \tag{49}$$

for $1 \le f_r \le h_r + 1$ and for $u_r \in [v_{rk_r} - \Delta_{r(k_r-1)}, v_{rk_r} + \Delta_{rk_r}]$ where Δ_{rk_r} is defined such that $g_{rk_r}(v_{rk_r} + \Delta_{rk_r}) = g_{r(k_r+1)}(v_{rk_r} + \Delta_{rk_r})$ for $1 \le r \le m$ and $1 \le k_r \le h_r$. Furthermore, the interpolation curve passing through a box described by specification (ii) assumes the general shape of the output of $\prod_{r=1}^{m} g_{rk_r}(u_r)$ if $\varepsilon_{rk_r-} \le \Delta_{r(k_r-1)}$ and $\varepsilon_{rk_r+} \le \Delta_{rk_r}$ for $1 \le r \le m$. For (37),

$$\sum_{k_1=1}^{h_1+1} \cdots \sum_{k_m=1}^{h_m+1} \prod_{r=1}^{m} g_{rk_r}(u_r) = 1 \tag{50}$$

for all **u**.

With the centers and slopes chosen to satisfy (49) and (50), which aid in satisfying specification (iii), the weights $w_{ik_1...k_m}$ can be found. Forming a matrix $\mathbf{G} \in \mathbb{R}^{px[(h_1+1)...(h_m+1)]}$ from the gaussian-type node's outputs, (38) is solved for the matrix $\mathbf{W} \in \mathbb{R}^{[(h_1+1)...(h_m+1)]xn}$. Since $h_r = p_r - 1$ for $1 \le r \le m$ and since $p = \prod_{r=1}^{m} p_r$, (38) describes a unique linear system of

equations. Furthermore, the results of Theorems 2.1 and 2.2 can be extended to the multi-dimensional case with the appropriate changes to the indices.

If the operating points are not equidistant, two choices are possible: adding extra pseudo-operating points such that equidistance occurs or locating the boundaries of the hyper-rectangles for each operating point such that the entire input space is covered. The first possibility is more viable when the operating points are close to being equidistant, and a few extra points are needed to achieve the equidistance property. Once this is accomplished, the above procedure for choosing the centers for equidistant operating points is followed for the set of operating points and added pseudo-operating points. When solving (38), extra pseudo-desired outputs are added such that (38) remains unique. These added outputs can be chosen to be the same as nearby ones or extrapolated values from surrounding ones.

For the second possibility, hyper-rectangles are found for the given operating points such that the entire input space is covered. Since the locations of the operating points are no longer equidistant, equation (36) can be replaced by

$$z_i(u) = \sum_{k=1}^{p} w_{ik}\, g_{1k}(u_1) \ldots g_{mk}(u_m) \qquad (51)$$

where $1 \le i \le n$. For low dimensional operating point parameters, the choosing of the hyper-rectangles may be performed manually. However, for higher dimensions, this may become impractical. In [Gonzalez90], this problem is addressed as the "CR_m" problem (or "CR_d" using the notation of [Gonzalez90]). For the CR_m problem, it is desired to to find m-dimensional hyper-rectangles of fixed size and orthogonal to the coordinate axes such that all the p operating points are covered. The problem CR_m is known to be NP-hard, and Gonzalez presents algorithms for approximating a solution to the problem. Once the hyper-rectangles are found, the centers of the sigmoid neurons are chosen as the boundaries between them. The slopes are chosen to satisfy the specifications, and the output layer weights are found via (38).

3 EXAMPLES

The examples are divided into two sections: those pertaining to the gaussian neural network described in Section 1 and those illustrating the sigmoid neural network of Section 2.

3.1 Gaussian Neural Network Examples

Example 3.1.1:

In [Peek90], a parameter learning method is presented and used to define the region of operation for an adaptive control system of a flexible space antenna. In one of the experiments described, an initial pulse disturbance is applied to the plant, and the adaptive controller is required to follow a zero-order reference model. The goal of the parameter learning system is to find values

for the four adaptive controller parameters (σ_1, σ_2, L, and $\tilde{L}$) for varied amplitudes of the initial pulse such that a defined performance index based on the output of the plant is small. In Table 1, the values found for the controller parameters for different pulse amplitudes are repeated. Using this table, the goal here is to construct a neural network scheduler such that a smooth interpolation is achieved between the 9 operating points. Since only the amplitude of the pulse disturbance specifies which values the adaptive controller should use for its 4 parameters and since all the amplitudes are equidistant, the results of Section 1.1.1 for the single-parameter scheduler with equidistant operating points of distance T = 0.5 are applicable. In reference to Figure 1, there is only one input to the neural network, which receives the amplitude of the disturbance, and the neural network's output sends 4 parameter values to the adaptive controller.

Table 1 Initial Disturbances and Parameter Sets.

Amplitude	σ_1	σ_2	L	$\tilde{L}$
2.0	0.093	10.05	49000.0	119703.96
2.5	0.302	7.35	44046.1	145910.16
3.0	0.307	15.79	37325.7	183327.84
3.5	0.876	9.556	44046.1	175092.19
4.0	0.808	10.48	29114.0	203493.90
4.5	1.767	10.48	32025.4	203493.90
5.0	3.924	8.70	48450.7	140073.75
5.5	6.928	7.73	41567.5	138395.55
6.0	11.08	10.05	41567.5	138395.55

For the two-layer gaussian neural network scheduler, the centers of the gaussian neurons in the hidden layer are set equal to the operating points per (4):

$$\mathbf{c} = [2.0 \;\; 2.5 \;\; 3.0 \;\; 3.5 \;\; 4.0 \;\; 4.5 \;\; 5.0 \;\; 5.5 \;\; 6.0]'.$$

Applying (8), the weights of the gaussian neurons in the hidden layer are $s_k = 13.3209$ for $\alpha = 2$ and for $1 \le k \le 9$. With the outputs $g(x_k)$ of the individual gaussian neurons for $1 \le k \le 9$ shown in Figure 6, the localized properties of the gaussian neurons are evident and (6) is satisfied. With $w_{ik} = 1$ for $1 \le i \le 4$ and $1 \le k \le 9$, the output $z = z_i$ satisfies (7) and is shown in Figure 7. For comparison, the resulting graph when the scale of the ordinate axis is increased is shown in Figure 8. Clearly, as desired in (7), an approximate unit signal is achieved. Forming the matrix $\mathbf{G} \in \mathbb{R}^{9x9}$ from the outputs of the hidden layer and forming the matrix $\mathbf{D} \in \mathbb{R}^{9x4}$ of the desired outputs of the neural network scheduler from the entries in Table 1, the weights $\mathbf{W}$ for the linear neurons in the output layer are found by solving (9), and after examining the weights, the approximation of (10) is true. Figures 9 to 12 show the outputs of the gaussian neural network scheduler, which are

the 4 parameter values sent to the adaptive controller. For comparison, the straight line approximations between the operating points are included in the four figures as dotted lines. As can be seen, $z_i(k) = d_i(k)$ for $1 \leq k \leq 9$ and $1 \leq i \leq 4$, and specification (i) is satisfied. In the regions nearby the operating points, the adaptive controller parameter values specified by the neural network scheduler are close to those specified by Table 1 satisfying specification (ii) for very thin and wide boxes. In regions between operating points, a swift yet smooth transition occurs between the value specified by Table 1, and specification (iii) is almost met. Between 4.0 and 4.5 in Figure 10, as well as in other places, specification (iii) requires a straight line, and this is almost satisfied.

For comparison, another set of interpolation curves is produced for an exponent value of $\alpha = 20$, instead of $\alpha = 2$, in Figures 13 to 16. Specification (i) is completely satisfied, and the areas around the operating points form plateaus and satisfy specification (ii) for very small ε_k and large δ_k. Specification (iii) is also approximately satisfied.

As another comparison, the design procedure suggested by [Moody89] and [Moody88] is used. With $h = p$, the centers are chosen as in (4). With $\alpha = 1$, the widths are chosen as $s_k = 4$ for $1 \leq k \leq 9$. The weights $\mathbf{W}$ are found by solving for $\mathbf{W}$ in (9). Figure 17 depicts the individual hidden layer neuron outputs, and (6) is not satisfied. With the curve from Figure 8 as a reference, Figure 18 shows the output $z = z_i$ for $w_{ik} = 1$ for $1 \leq i \leq 4$ and $1 \leq k \leq 9$, and (7) is not satisfied. In Figures 19 to 22, the 4 interpolation curves are displayed along with the straight line approximations. Specification (i) is met due to the solving of (9) for $\mathbf{W}$. However, specification (ii) can only be satisfied for small boxes, and specification (iii) is not met at all.

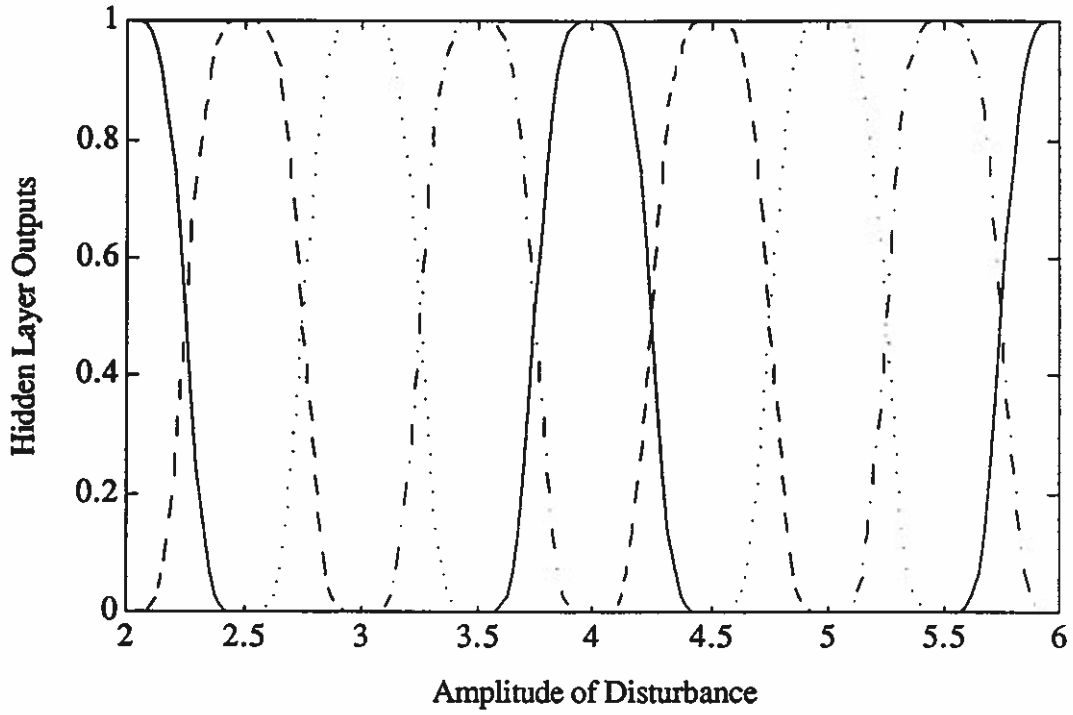

Figure 6 Individual hidden layer outputs with $\alpha = 2$ for Example 3.1.1.

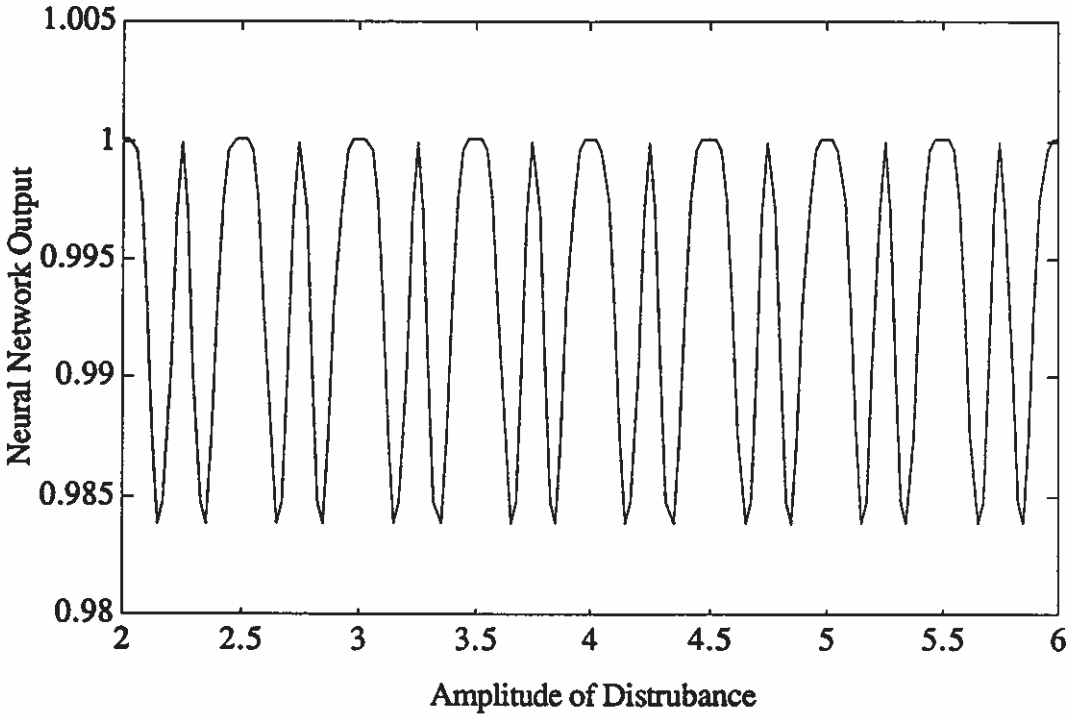

Figure 7 Summation of hidden layer outputs with $\alpha = 2$ for Example 3.1.1.

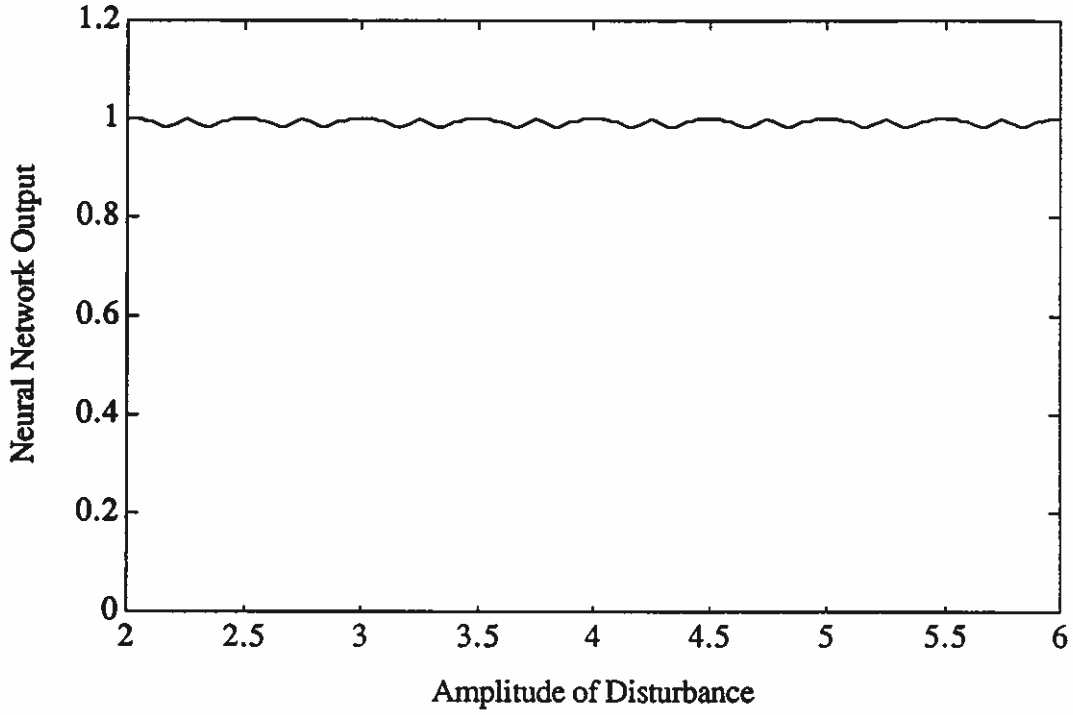

Figure 8 Summation of hidden layer outputs with $\alpha = 2$ for Example 3.1.1.

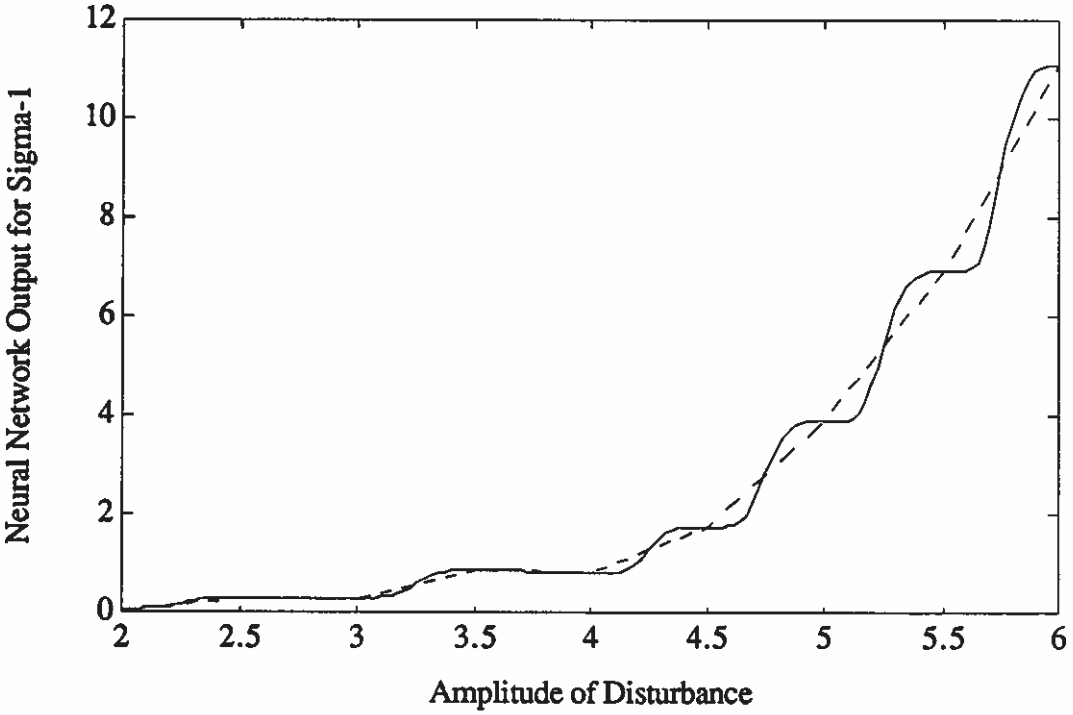

Figure 9 Neural network output for σ_1 with $\alpha = 2$ for Example 3.1.1.

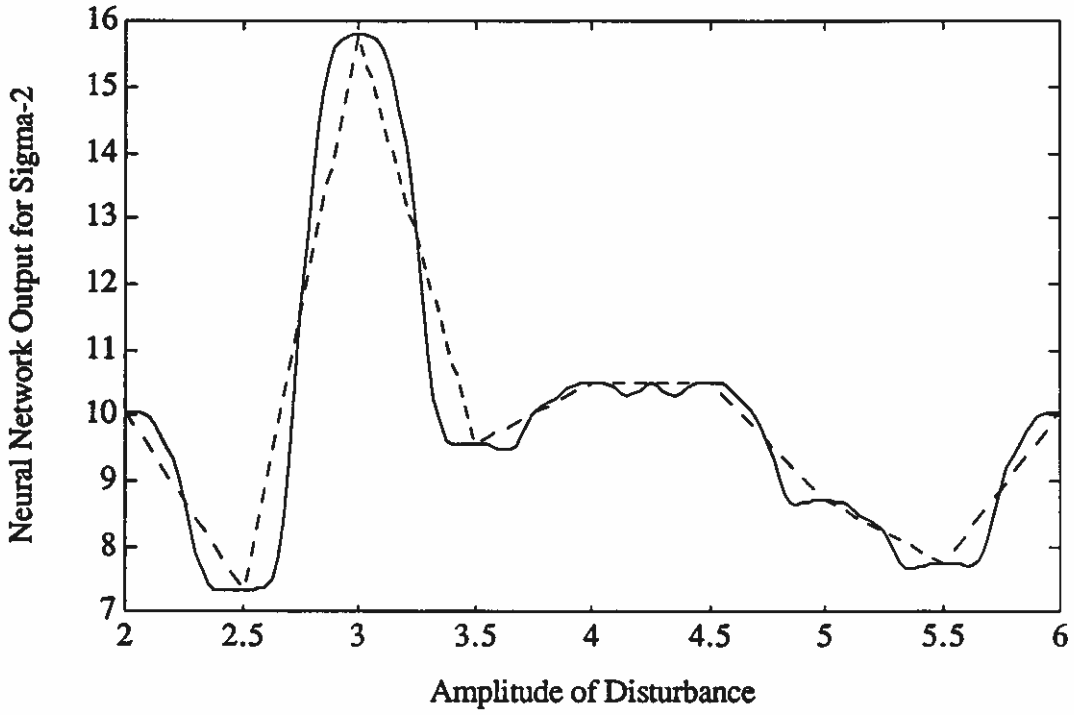

Figure 10 Neural network output for σ_2 with $\alpha = 2$ for Example 3.1.1.

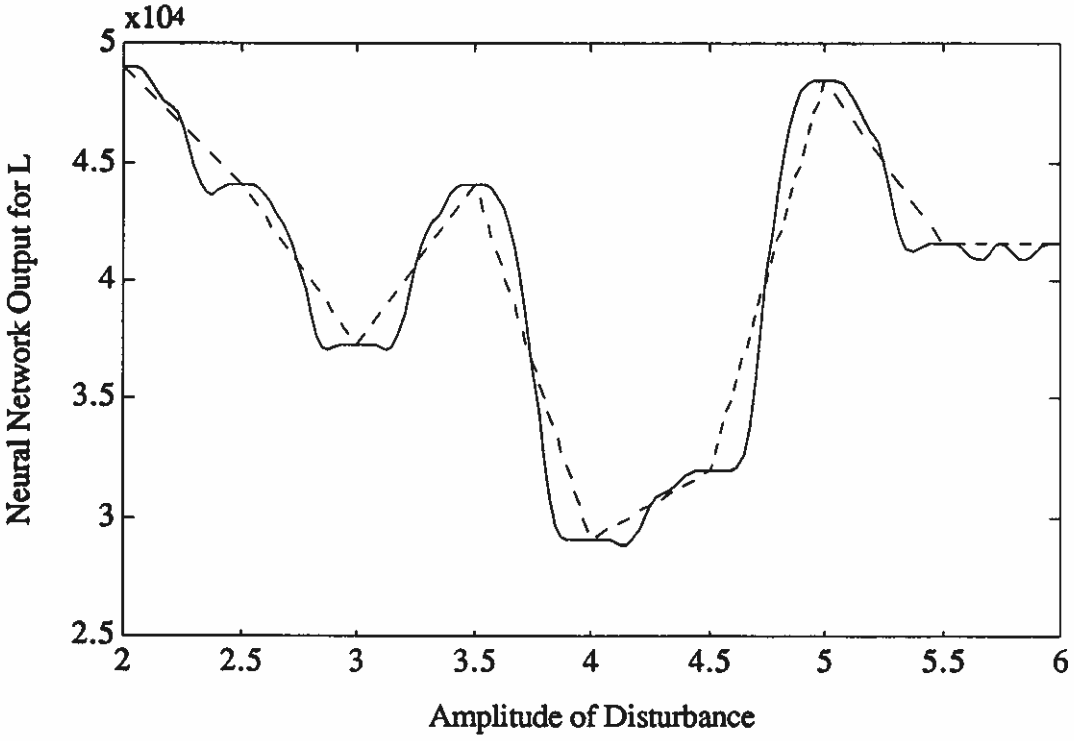

Figure 11 Neural network output for L with $\alpha = 2$ for Example 3.1.1.

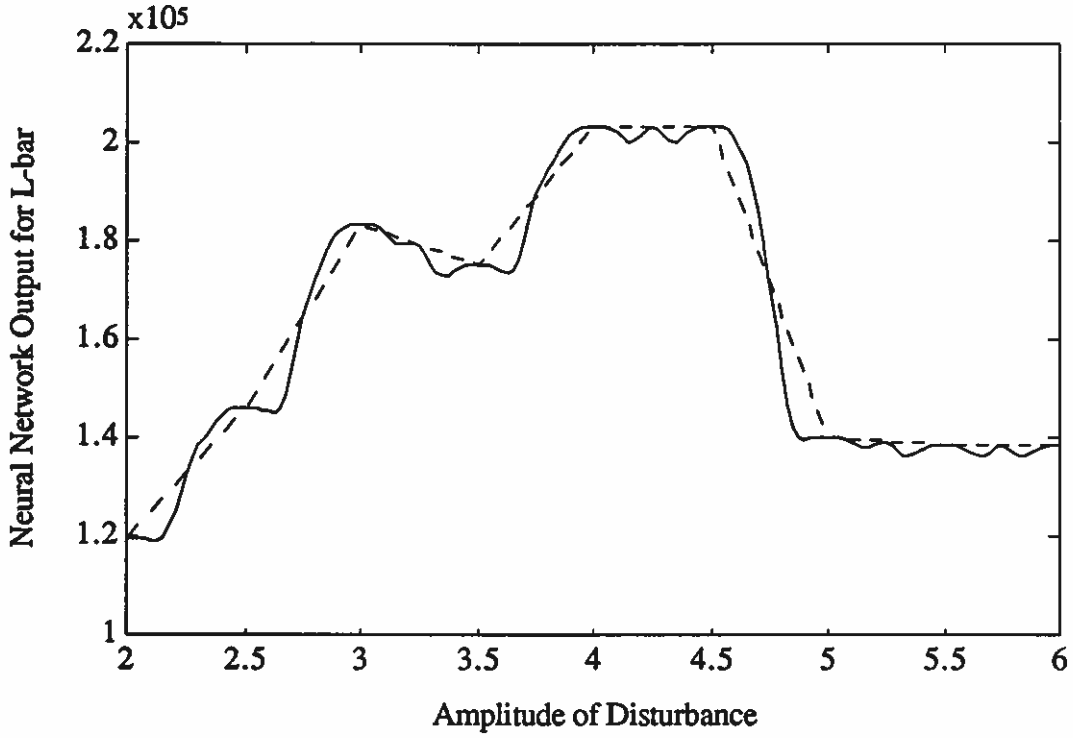

Figure 12 Neural network output for $\bar{L}$ with $\alpha = 2$ for Example 3.1.1.

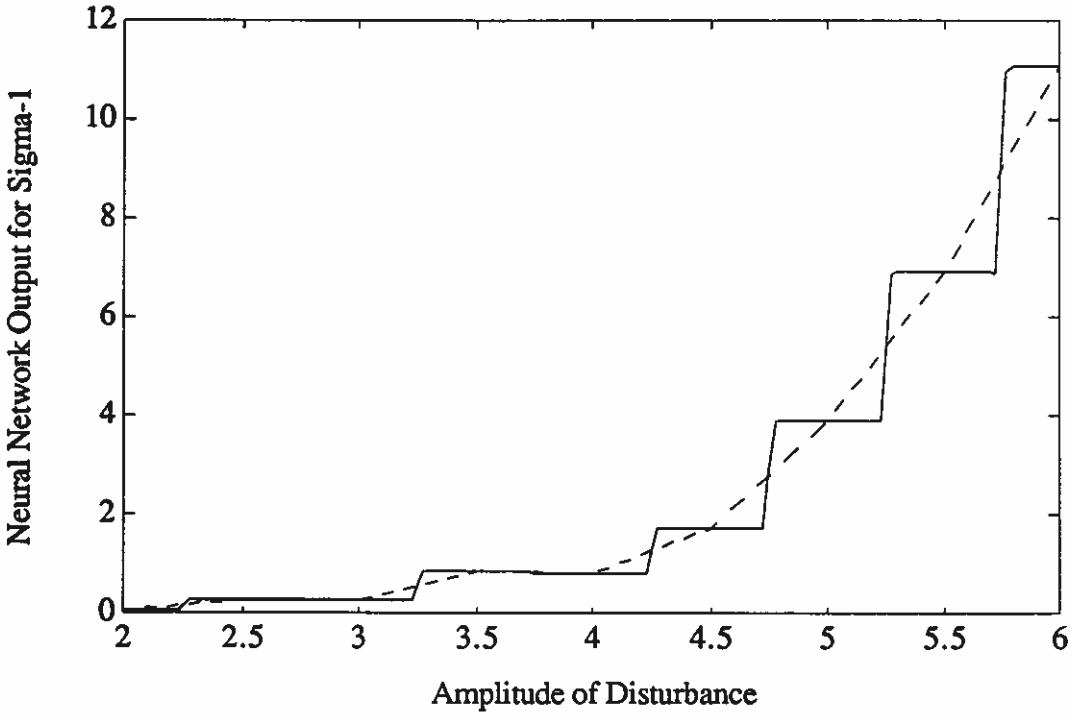

Figure 13 Neural network output for σ_1 with $\alpha = 20$ for Example 3.1.1.

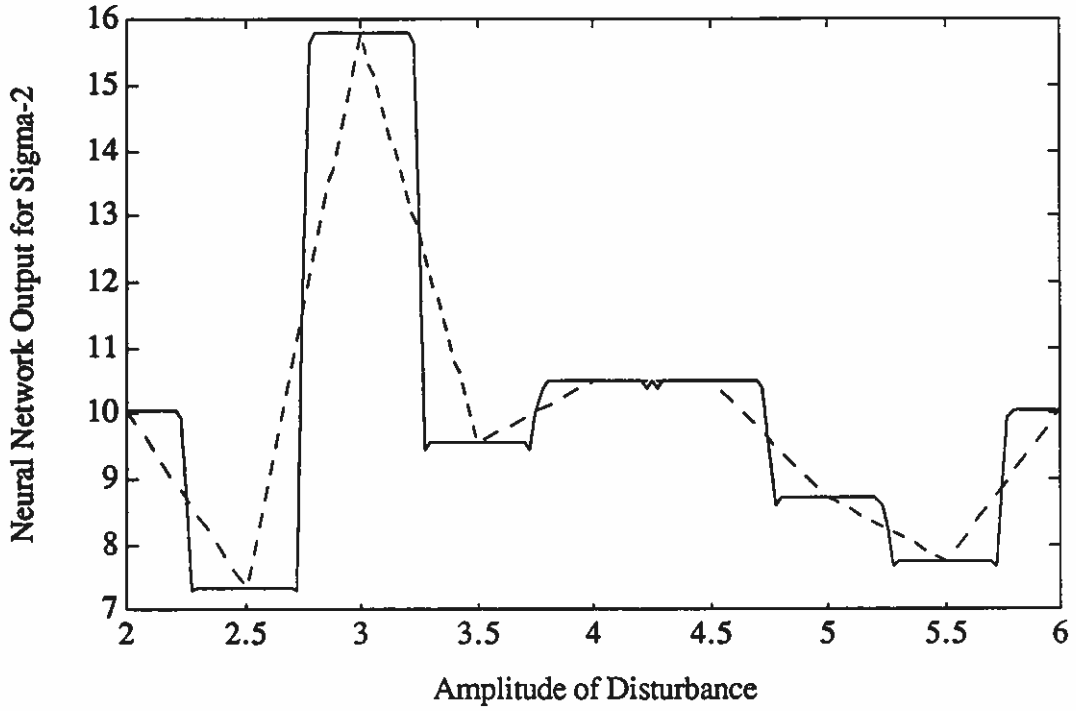

Figure 14 Neural network output for σ_2 with $\alpha = 20$ for Example 3.1.1.

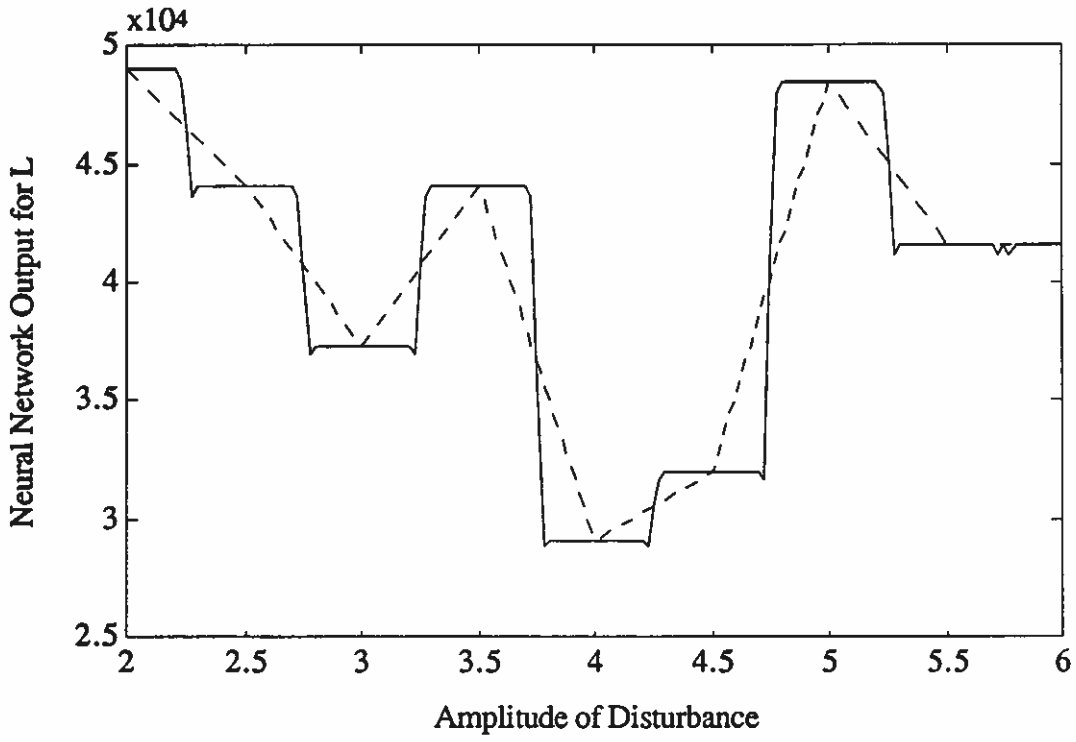

Figure 15 Neural network output for L with $\alpha = 20$ for Example 3.1.1.

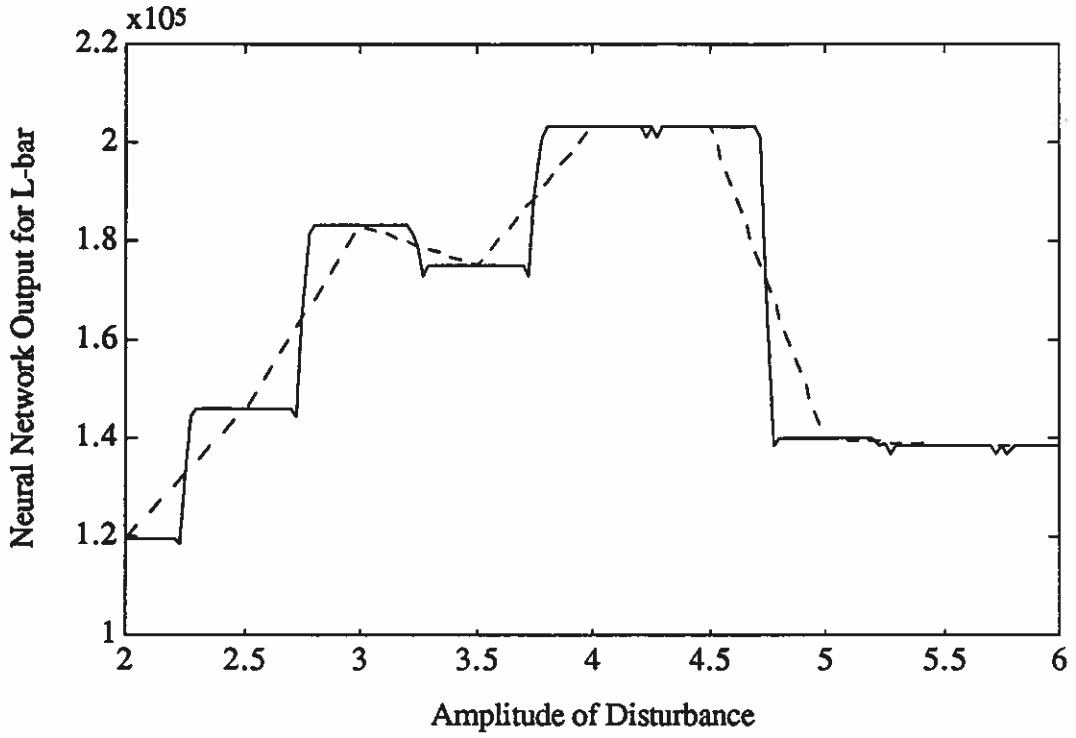

Figure 16 Neural network output for $\bar{L}$ with $\alpha = 20$ for Example 3.1.1.

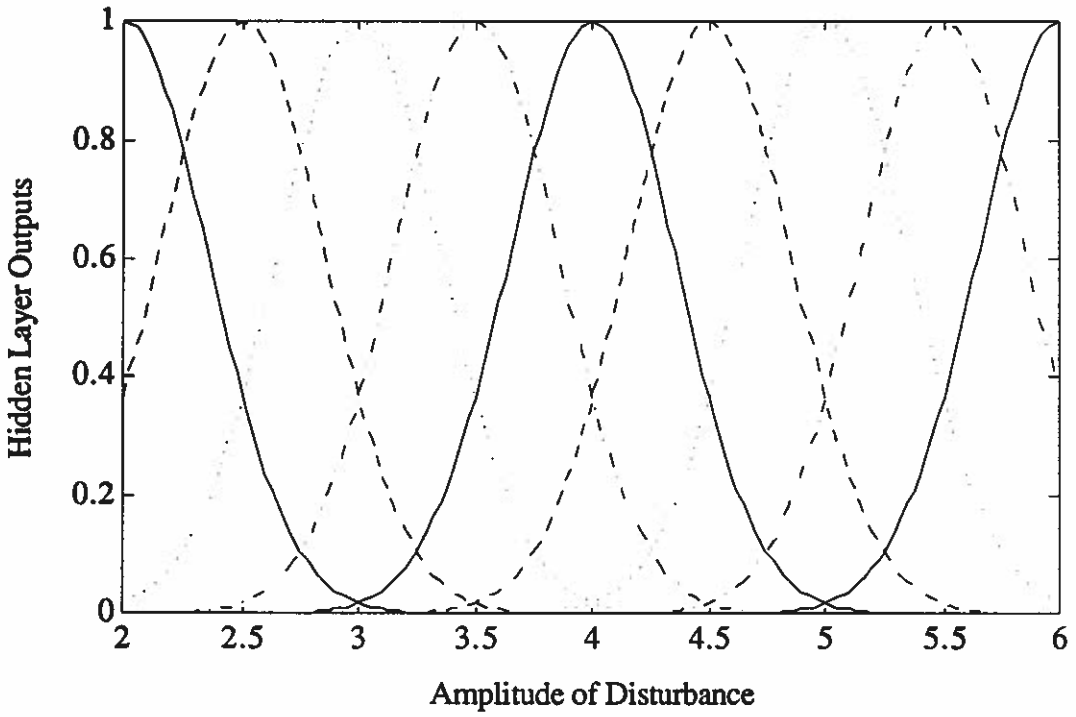

Figure 17 Individual hidden layer outputs with Moody's method for Example 3.1.1.

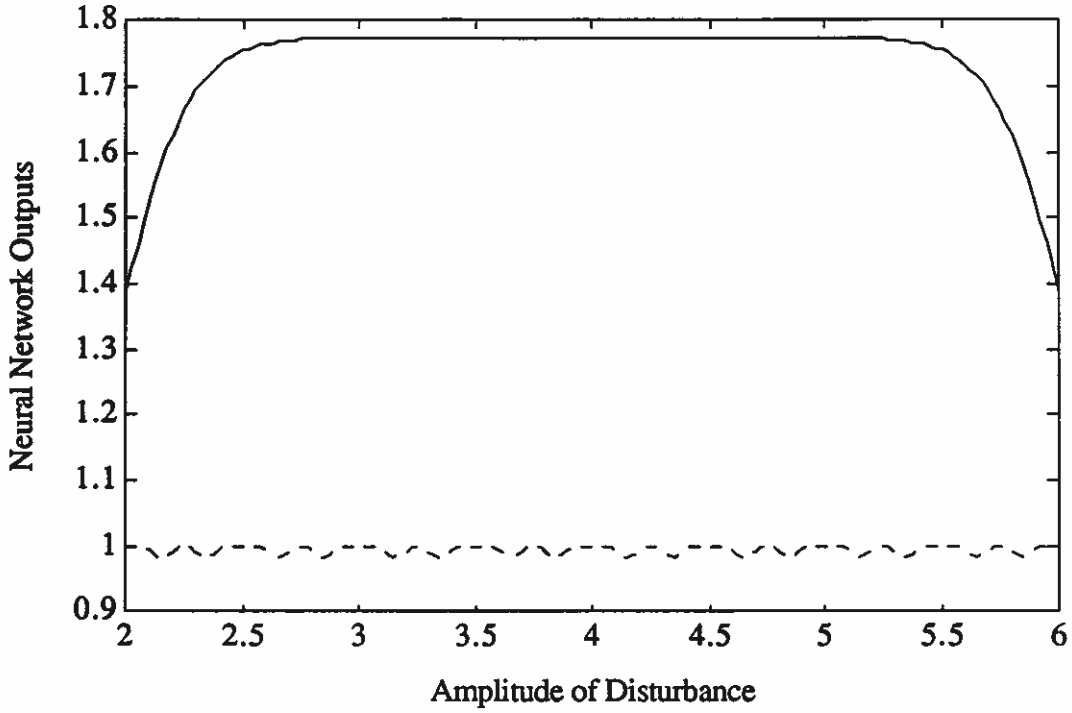

Figure 18 Summation of hidden layer outputs with Moody's method for Example 3.1.1.

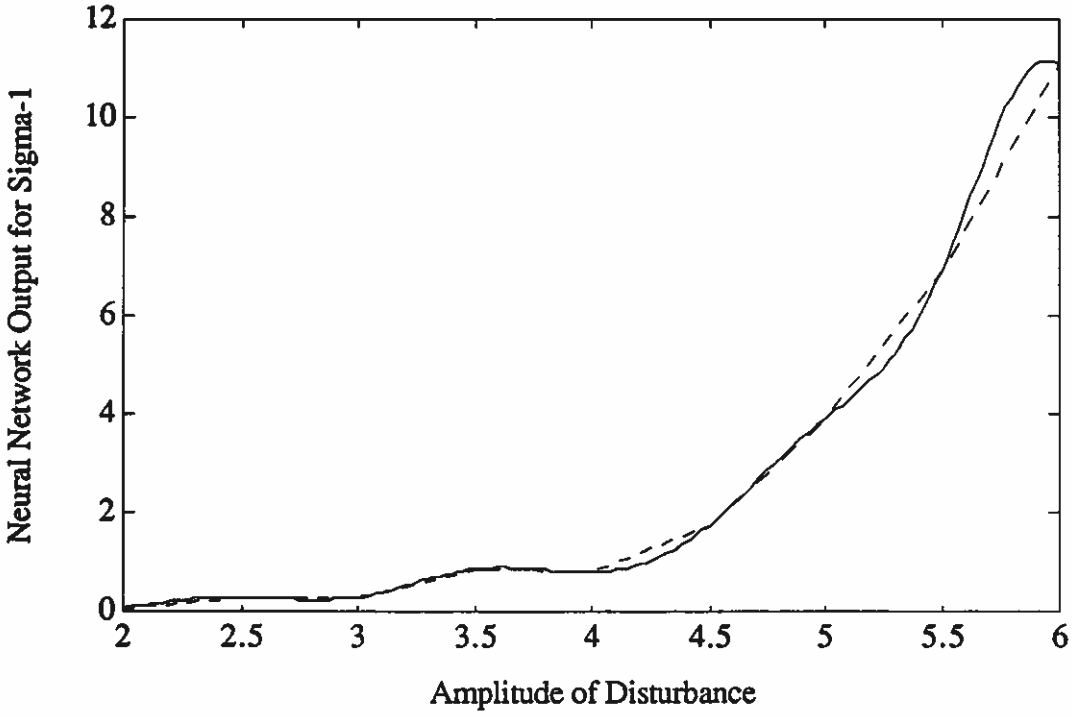

Figure 19 Neural network output for σ_1 with Moody's method for Example 3.1.1.

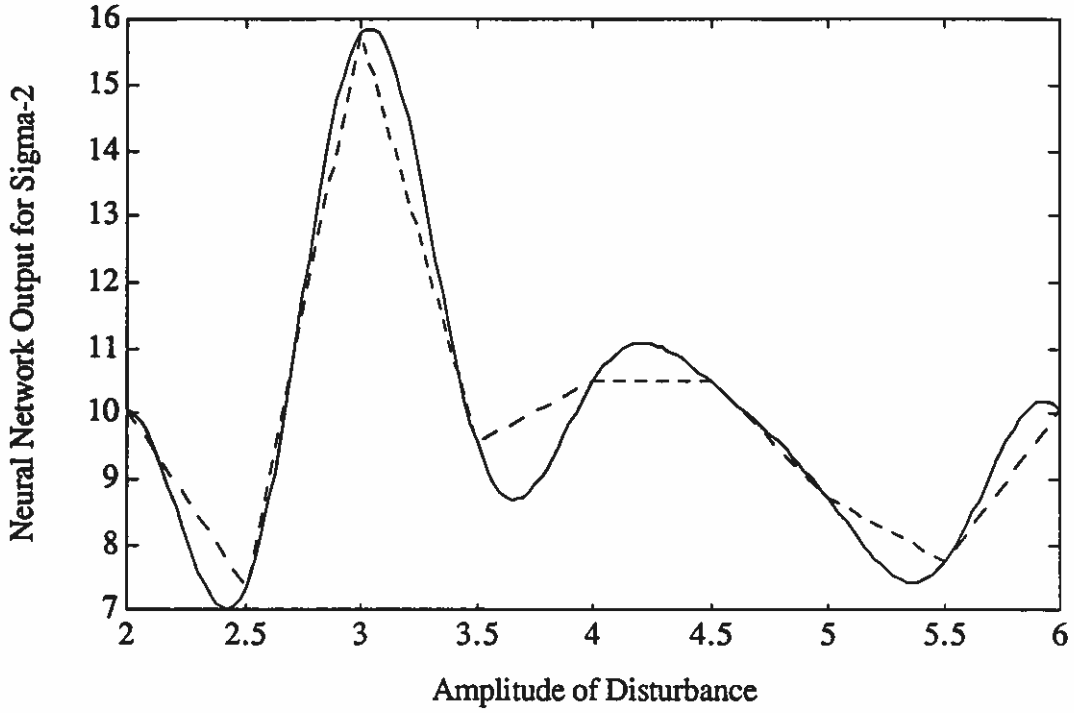

Figure 20 Neural network output for σ_2 with Moody's method for Example 3.1.1.

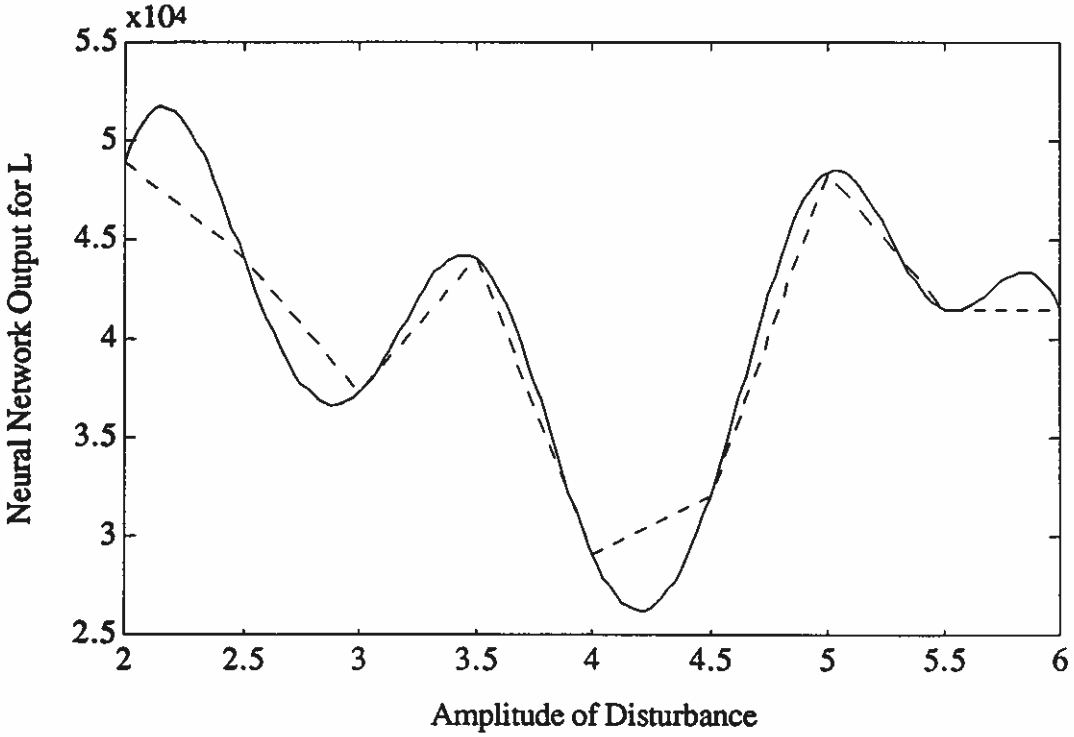

Figure 21 Neural network output for L with Moody's method for Example 3.1.1.

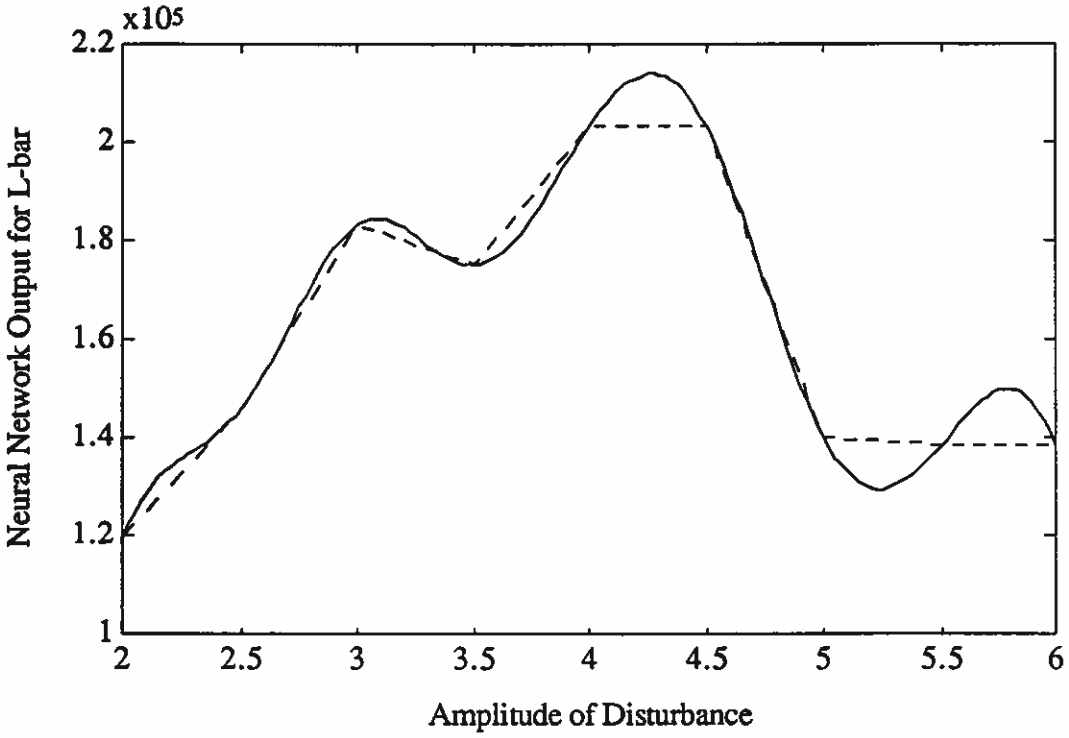

Figure 22 Neural network output for $\bar{L}$ with Moody's method for Example 3.1.1.

Example 3.1.2:

In Table 2, two more disturbances and the appropriate adaptive controller parameter values found by the parameter learning system of [Peek90] are shown. With the addition of these disturbances to those in Table 1, the operating points for the scheduler are no longer equidistant, and the results of Section 1.1.2 are applicable. The new scheduler operating points are

$$\mathbf{v} = [2.0 \;\; 2.25 \;\; 2.5 \;\; 3.0 \;\; 3.5 \;\; 4.0 \;\; 4.5 \;\; 5.0 \;\; 5.5 \;\; 6.0 \;\; 7.0]'.$$

To cope with the disturbance amplitudes not being equidistant, extra gaussian neurons are added to the hidden layer. First, 11 gaussian neurons are formed with centers equal to the amplitudes of the disturbances from Tables 1 and 2. Next, 10 extra gaussian neurons are added with 7 placed such that their centers are between 2.75 and 5.75 in increments of 0.5 and 3 with centers of 6.25, 6.50, and 6.75. By doing this, the resulting 21 neurons have centers which are equidistant:

$$\mathbf{c} = [2.00 \;\; 2.25 \;\; 2.50 \;\; 2.75 \;\; 3.00 \ldots 6.00 \;\; 6.25 \;\; 6.50 \;\; 6.75 \;\; 7.00]'.$$

Applying (8) with T = 0.25 and with $\alpha = 2$, the weights of the gaussian neurons in the hidden layer are $s_k = 53.2835$ for $1 \le k \le 21$. With $w_{ik} = 1$ for $1 \le i \le 4$ and $1 \le k \le 21$, the output $z = z_i$ is shown in Figure 23 to approximate (7). Next, so that the linear system of (9) remains unique, extra desired outputs for the neural network associated with the inputs equal to the new centers are specified. For the inputs between 2.75 and 5.75 in increments of 0.5 and for the input 6.5, the

desired neural network outputs are chosen to be halfway between the appropriate adaptive controller parameters specified in Tables 1 and 2. For the input of 6.25, the desired outputs are chosen to be the same as those for 6.00, and for the input of 6.75, the desired outputs are chosen to be the same as those for 7.00. The matrices $\mathbf{G} \in \mathbb{R}^{21x21}$ and $\mathbf{D} \in \mathbb{R}^{21x4}$ are formed, and the unique linear system of equations in (9) is solved for $\mathbf{W}$. For the neural network scheduler formed, Figures 24 to 27 show its outputs, which are the 4 parameter values sent to the adaptive controller. Once again, specification (i) is satisfied exactly, and specification (iii) is satisfied approximately. Specification (ii) can be satisfied for smaller boxes compared to those boxes possible when the points are equidistant, as in Example 3.1.1.

Table 2 Extra Initial Disturbances and Parameter Sets.

Amplitude	σ_1	σ_2	L	$\tilde{L}$
2.25	0.213	7.04	39200.0	131674.35
7.0	14.41	19.10	41567.5	110716.44

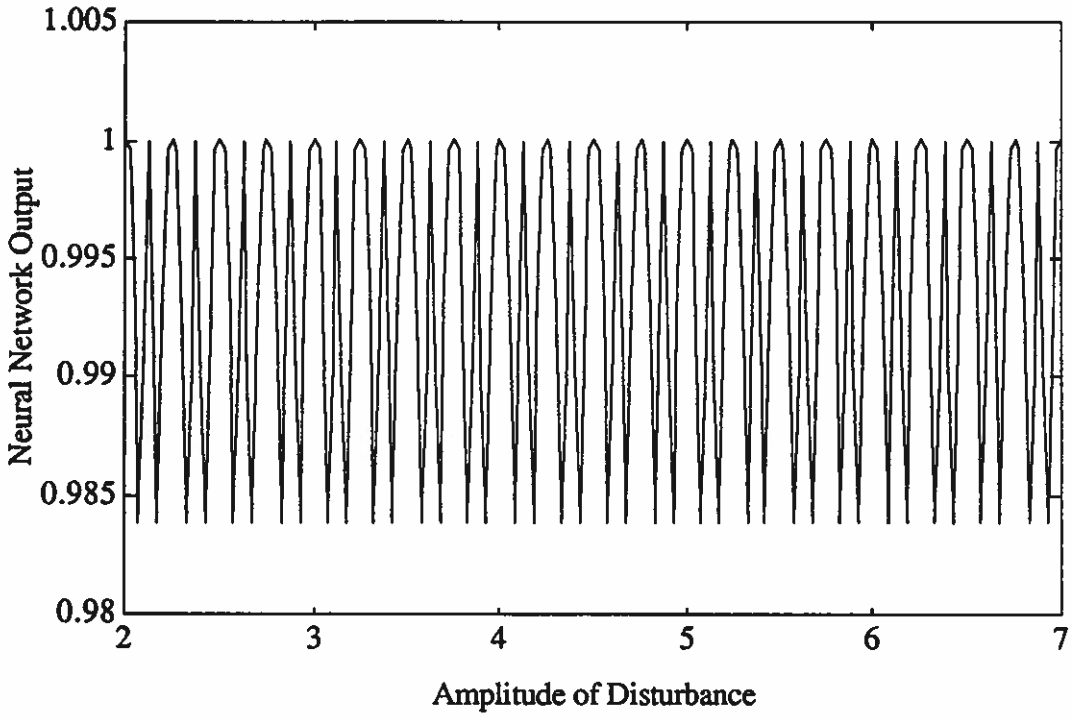

Figure 23 Summation of hidden layer outputs for Example 3.1.2.

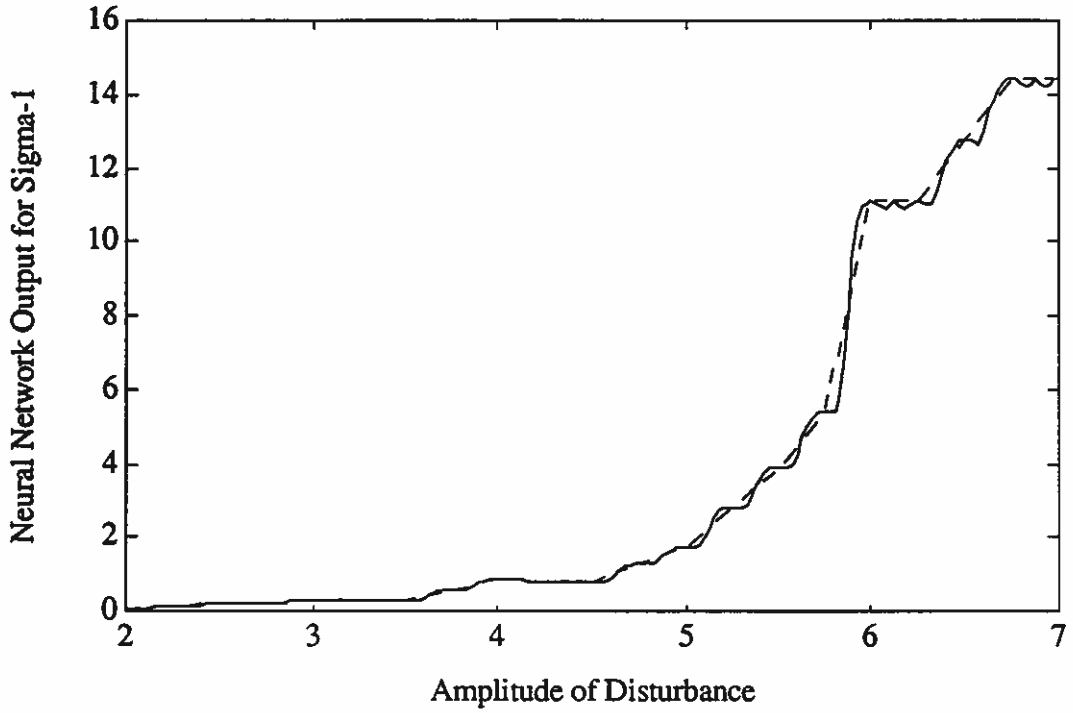

Figure 24 Neural network output for σ_1 for Example 3.1.2.

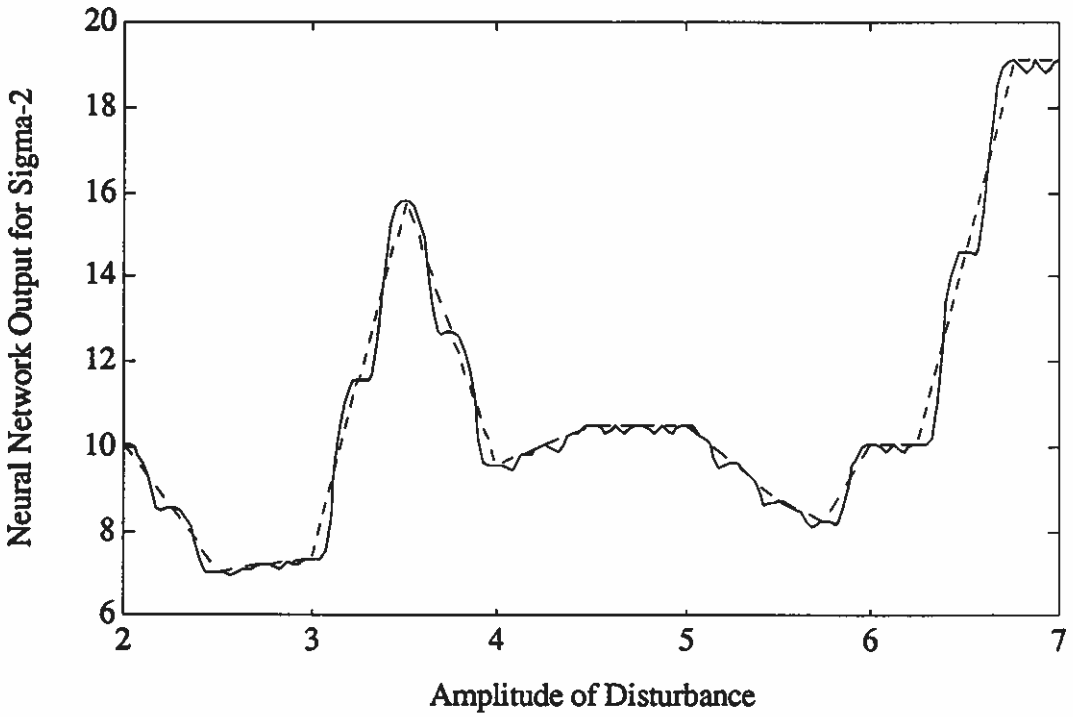

Figure 25 Neural network output for σ_2 for Example 3.1.2.

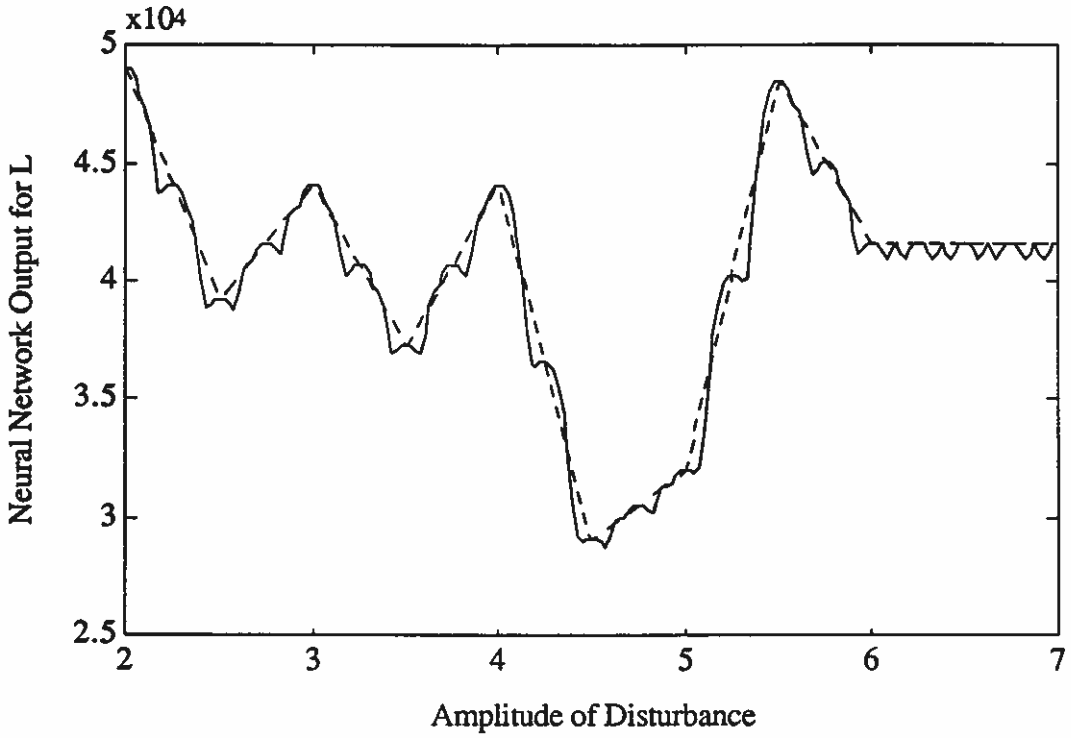

Figure 26 Neural network output for L for Example 3.1.2.

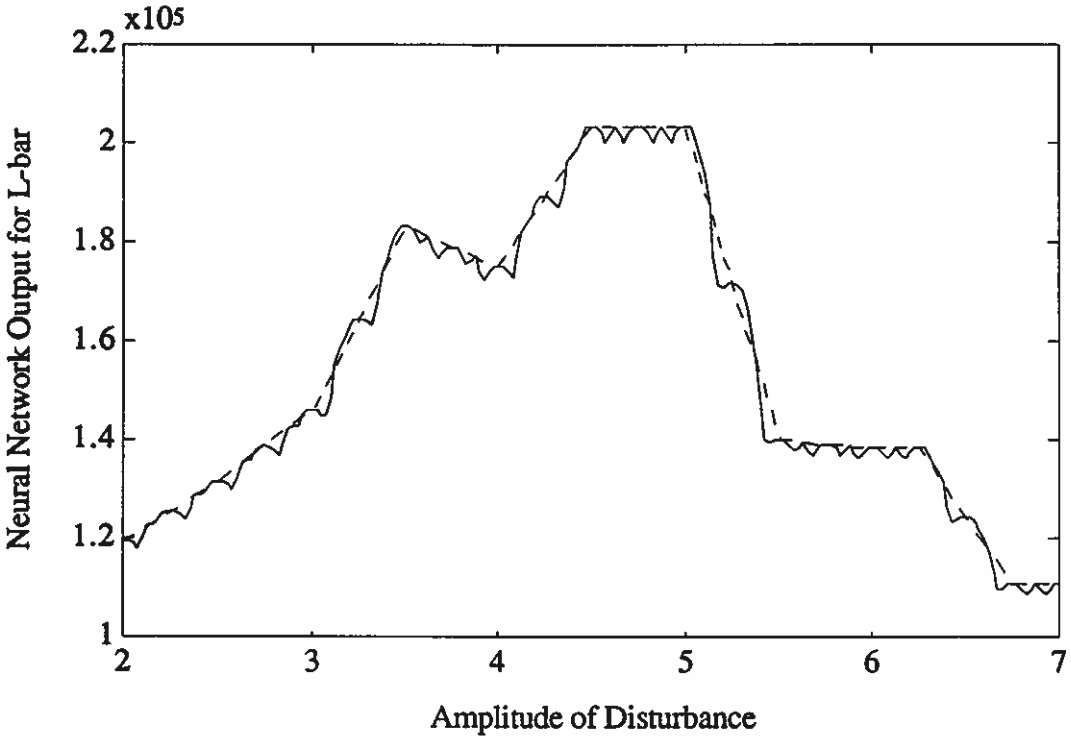

Figure 27 Neural network output for $\bar{L}$ for Example 3.1.2.

Example 3.1.3:

In another of the experiments described in [Peek90], in which a parameter learning method is used to define the region of operation for an adaptive control system of a flexible space antenna, a command step input is applied as the reference signal of the system, and the adaptive controller is required to follow a highly damped second-order reference model. Once again, the goal of the parameter learning system is to find values for the four adaptive controller parameters (σ_1, σ_2, L, and $\bar{L}$) for varied amplitudes of the command step input such that a defined performance index based on the output of the plant is small. In Table 3, the parameters found for the controller for different step amplitudes are repeated. Using this table, the goal here is to construct a neural network scheduler such that a smooth interpolation is achieved between the 8 operating points and such that the three specifications are satisfied. Since only the amplitude of the step input specifies which values the adaptive controller requires for its 4 parameters and since all the amplitudes are not equidistant, the results of Section 1.1.2 for the single-parameter scheduler with non-equidistant operating points are applicable. In reference to Figure 1, there is only one input to the neural network, which receives the amplitude of the step input, and there are four outputs from the neural network's output sends 4 parameters to the adaptive controller.

With the vector of scheduler operating points, 8 gaussian neurons are formed with centers equal to these step input amplitudes. Since all of the operating points are equidistant except for 1.00 and 1.10, an additional gaussian neuron is added with a center of 1.05 such that the vector of centers is

$$\mathbf{c} = [1.00 \;\; 1.05 \;\; 1.10 \;\; 1.15 \;\; 1.20 \;\; 1.25 \;\; 1.30 \;\; 1.35 \;\; 1.40]'.$$

Applying (8) with T = 0.05 and with $\alpha = 2$, the weights of the gaussian neurons in the hidden layer are $s_k = 1332.0874$ for $1 \le k \le 9$. With $w_{ik} = 1$ for $1 \le i \le 4$ and $1 \le k \le 9$, the output $z = z_i$ is shown in Figure 28 to be approximately a unit signal and satisfies (7). Next, so that the linear system of (9) remains unique, pseudo-desired outputs for the neural network associated with the input of 1.05 are chosen to be halfway between the appropriate adaptive controller parameters specified in Table 3 for the inputs of 1.00 and 1.10. The matrices $\mathbf{G} \in \mathbb{R}^{9x9}$ and $\mathbf{D} \in \mathbb{R}^{9x4}$ are formed, and the unique system in (9) is solved for W. For the neural network scheduler formed, Figures 29 to 31 show the smooth interpolation curves formed, which are the parameters σ_1, σ_2, and L sent to the adaptive controller. The plot for $\bar{L}$ is not shown since the designed scheduler in Table 3 specifies that this parameter is 0 for most of the operating points. As in the previous interpolation curves, the straight line approximations are included for comparison and denoted as a dashed lines. For the derived interpolation curves, specification (i) is satisfied exactly, and specification (iii) is satisfied approximately. Specification (ii) can be satisfied for thin wide boxes.

Table 3 Step Input Commands and Parameter Sets.

Amplitude	σ_1	σ_2	L	$\bar{L}$
1.00	-0.098	-15.0	0.409	0.0
1.10	-0.091	-3.0	0.383	0.0
1.15	-0.084	-3.0	0.350	0.0
1.20	-0.078	13.0	0.322	0.0
1.25	-0.072	-15.0	0.297	0.0
1.30	-0.066	-15.0	0.274	0.0
1.35	-0.062	13.0	0.254	0.001
1.40	-0.057	-3.0	0.236	0.0

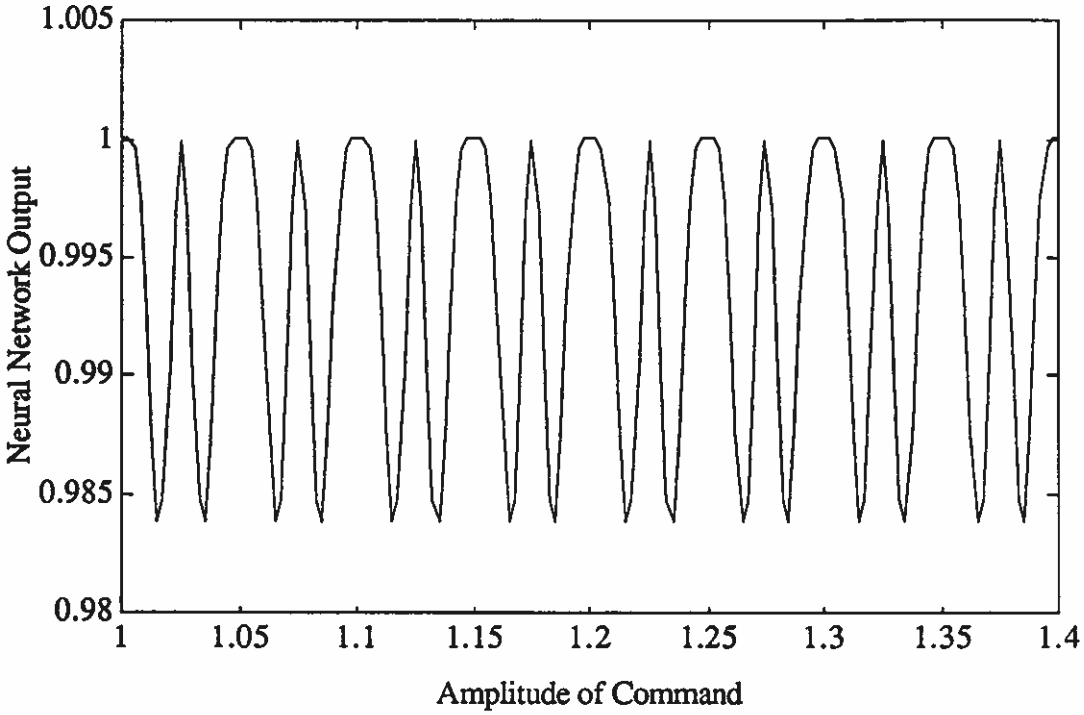

Figure 28 Summation of hidden layer outputs for Example 3.1.3.

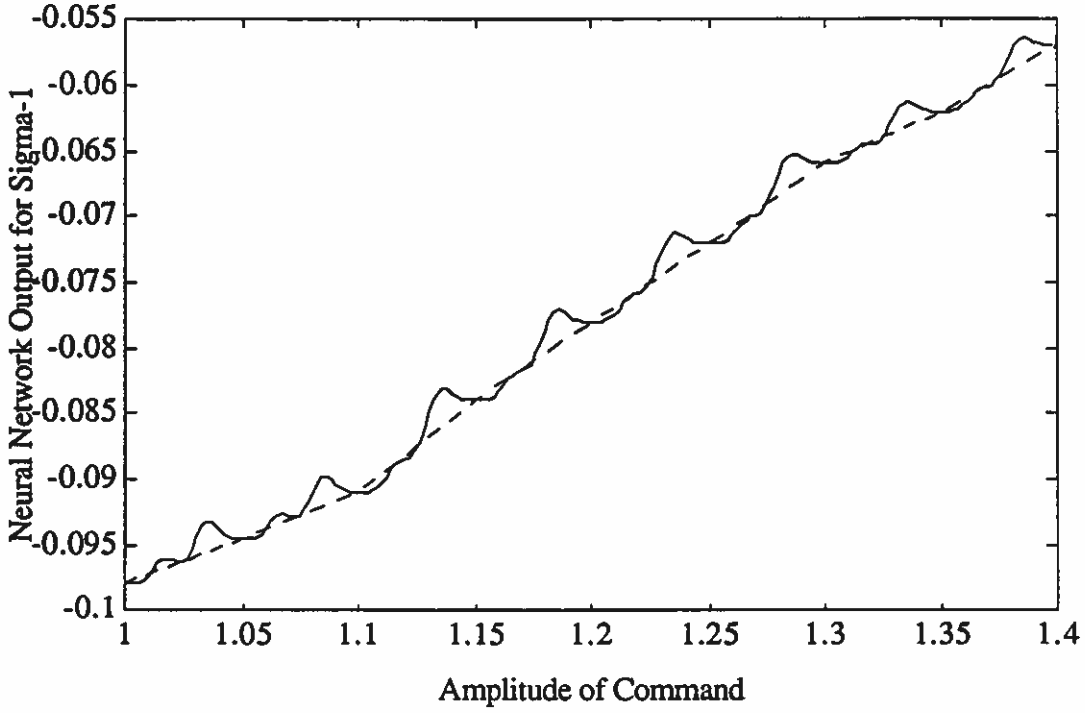

Figure 29 Neural network output for σ_1 for Example 3.1.3.

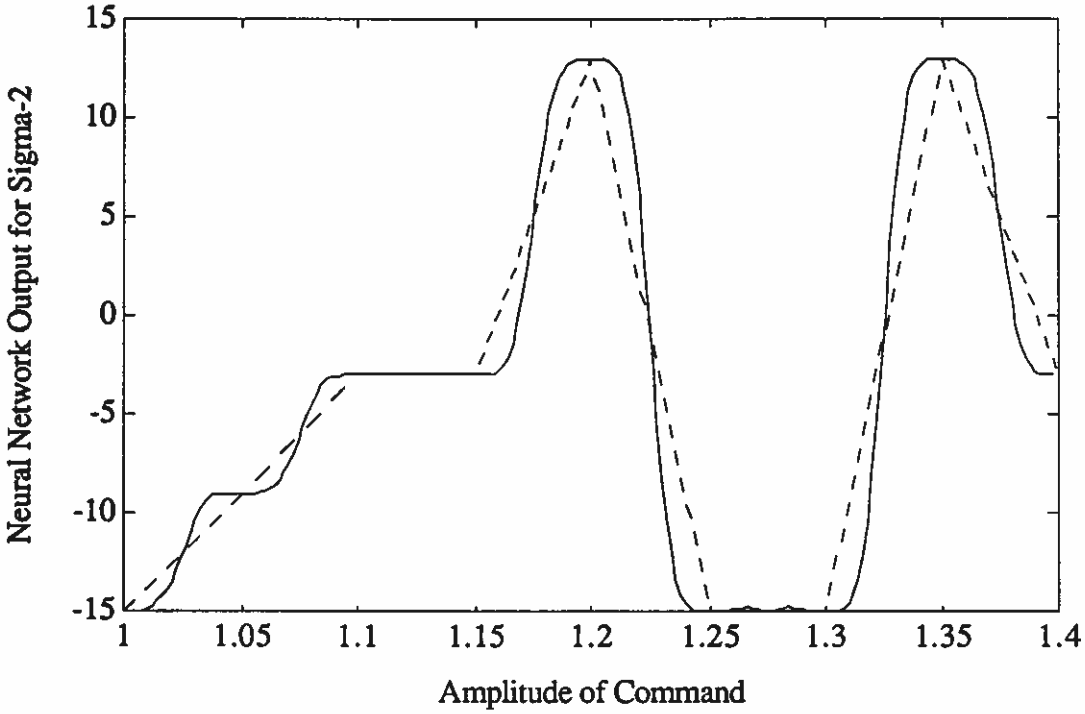

Figure 30 Neural network output for σ_2 for Example 3.1.3.

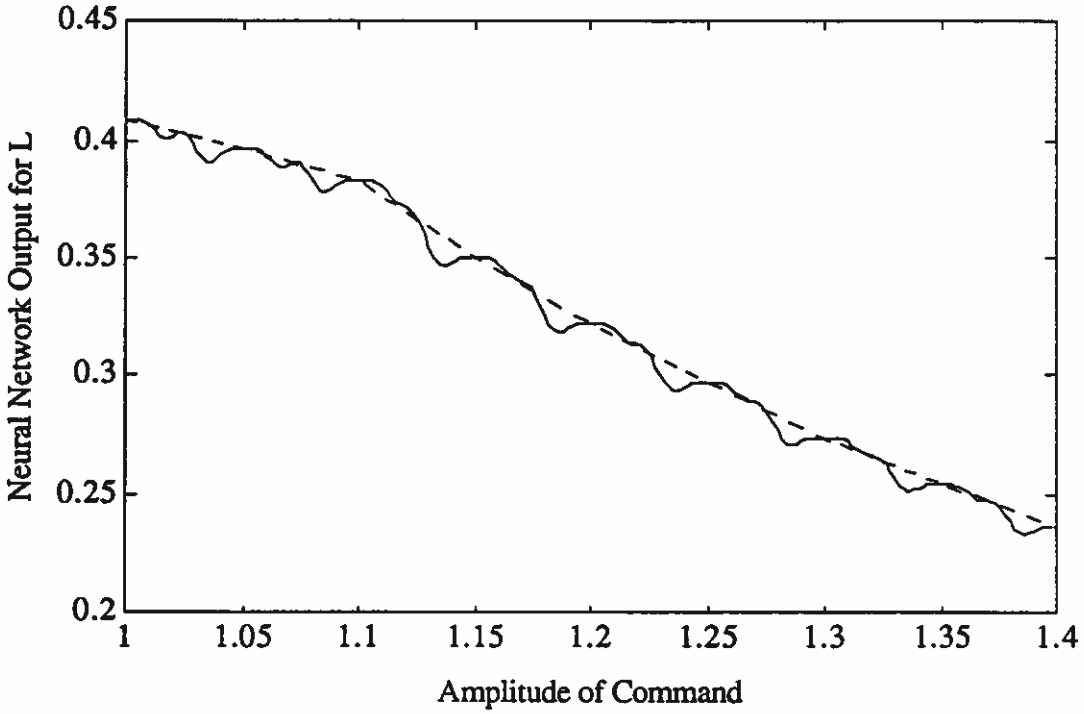

Figure 31 Neural network output for L for Example 3.1.3.

Example 3.1.4:

In [Hackney77], linear models of a F100 engine are developed for various flight points based on altitude and mach number. In Table 4, 6 of these flight points are listed with fictitious controller parameters, and in Figure 32, these six are diagrammed. It is desired to develop a neural network scheduler for interpolation between these flight points. The scheduler desired is a multi-parameter one, and the results of Section 1.2 are applicable. Since the operating points are not equidistant, the method of adding extra gaussian neurons to the hidden layer is chosen, and 6 extra gaussian neurons are added to achieve equidistance. With r = 1 corresponding to the altitude and with r = 2 corresponding to the mach number, the matrix of centers is

$$C' = \begin{bmatrix} 10 & 10 & 10 & 10 & 20 & 20 & 20 & 20 & 30 & 30 & 30 & 30 \\ 0.50 & 0.75 & 1.00 & 1.25 & 0.50 & 0.75 & 1.00 & 1.25 & 0.50 & 0.75 & 1.00 & 1.25 \end{bmatrix}$$

and the vector of outputs for equation (9) is

$$\mathbf{d} = [1 \;\; 2 \;\; 3 \;\; 4 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 5 \;\; 6 \;\; 6]'.$$

Applying (28) with $\alpha = 20$, $T_1 = .25$, and $T_2 = 10$, $s_{k1} = 62.8378$ and $s_{k2} = 0.0393$ for $1 \leq k \leq 12$. Solving (9) for w, the interpolation curve is shown in Figure 33. The lowest corner corresponds to the point (8, 0.4), the most left corner to (32, 0.4), and the most right corner to (8, 1.35). Specifications (i) and (ii) are clearly satisfied, but due to the elliptical nature of the multi-

dimensional gaussian function, Specification (iii) is not met and unwanted "valleys" are formed in the interpolation curve.

For comparison, the design procedure suggested by [Moody89] and [Moody88] is used. There are 12 neurons used with centers as specified above. The exponent α is changed to 1, and the widths are chosen per (22): $s_{k1} = 1/T_1^2 = 16$ and $s_{k2} = 1/T_2^2 = 0.01$ for $1 \le k \le 12$. Note that two widths are used instead of one as suggested by Moody. Solving for **w** in (9), the resulting interpolation curve is shown in Figure 34. The lowest corner corresponds to the point (8, 0.4), the most left corner to (32, 0.4), and the most right corner to (8, 1.35). After examining the actual values for the curve, specification (i) is met. Specification (ii), however, can not be satisfied for small thin boxes around the operating points, and in comparison to the curve in Figure 33, specification (iii) is better satisfied by the curve of Figure 34.

Table 4 Selected Flight Points for F100 Engine.

Amplitude (Thousands Feet)	Mach Number	Controller Parameters
10	0.75	2
10	1.00	3
10	1.25	4
20	0.50	1
30	0.75	5
30	1.00	6

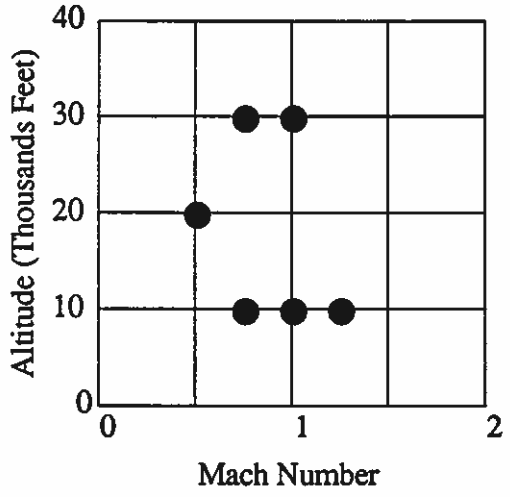

Figure 32 Selected F100 flight points for Example 3.1.4.

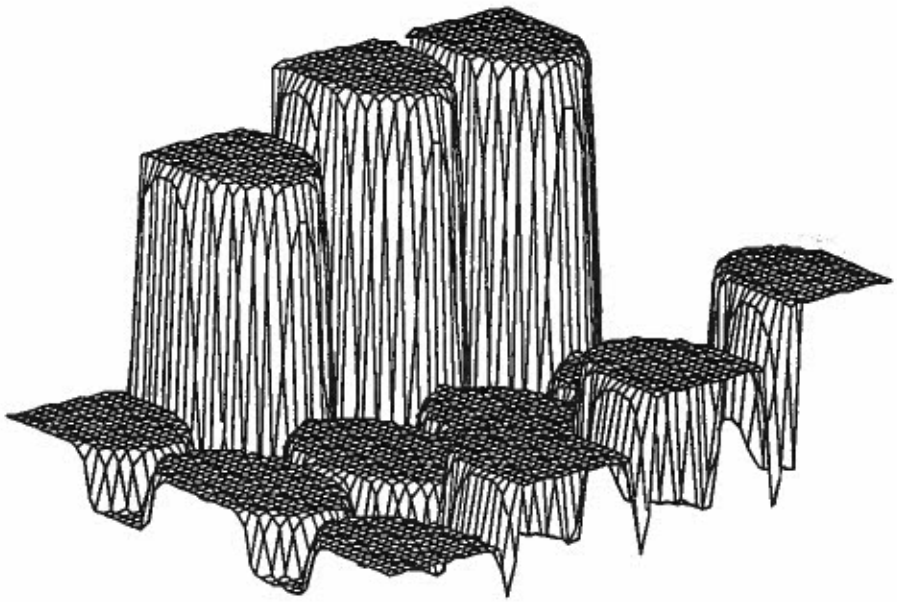

Figure 33 Neural network output for Example 3.1.4

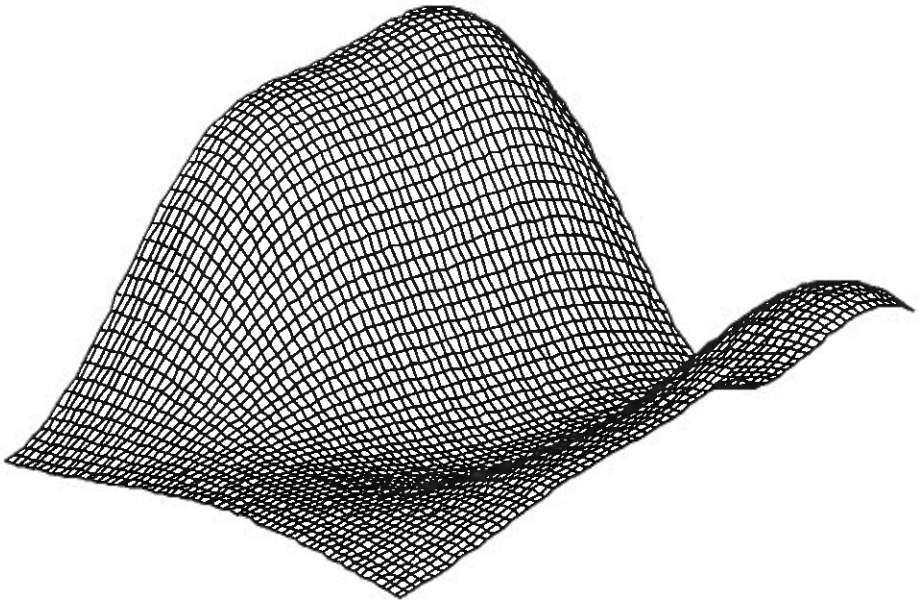

Figure 34 Neural network output with Moody's method for Example 3.1.4.

3.2 Sigmoid Neural Network Examples

Example 3.2.1:

In Examples 3.1.1 and 3.1.2, gaussian neural networks were used to implement interpolation curves between adaptive controller parameters, which were derived in [Peek90]. In Table 5, which is a combination of Tables 1 and 2, the parameters for the controller for different pulse amplitudes are repeated. Using this table, the goal here is to construct a sigmoid neural network scheduler such that a smooth interpolation is achieved between the 11 operating points. Since only the amplitude of the pulse disturbance specifies which values the adaptive controller should use for its 4 parameters, the results of Section 2.1 for the single-parameter scheduler are applicable.

Table 5 Initial Disturbances and Parameter Sets.

Amplitude	σ_1	σ_2	L	$\bar{L}$
2.0	0.093	10.05	49000.0	119703.96
2.25	0.213	7.04	39200.0	131674.35
2.5	0.302	7.35	44046.1	145910.16
3.0	0.307	15.79	37325.7	183327.84
3.5	0.876	9.556	44046.1	175092.19
4.0	0.808	10.48	29114.0	203493.90
4.5	1.767	10.48	32025.4	203493.90
5.0	3.924	8.70	48450.7	140073.75
5.5	6.928	7.73	41567.5	138395.55
6.0	11.08	10.05	41567.5	138395.55
7.0	14.41	19.10	41567.5	110716.44

For the two-layer sigmoid neural network, the centers of the sigmoid neurons in the hidden layer are set halfway between the operating points according to (34):

$$\mathbf{c} = [2.125\ 2.375\ 2.75\ 3.27\ 3.75\ 4.25\ 4.75\ 5.25\ 5.75\ 6.5]'.$$

The slopes for the hidden layer neurons are chosen as $s_k = 75$ for $1 \le k \le 10$. In Figure 35, the 10 sigmoid functions are plotted. As shown in Figure 36, the localized properties of the gaussian nodes g_k are evident and (35) and (36) are satisfied. Notice the asymmetrical gaussian-type node formed for the operating point v(10) = 6.0. With $w_{ik} = 1$ for $1 \le i \le 4$ and $1 \le k \le 10$, the output z $= z_i$ satisfies (37) and is shown in Figure 37. Forming the matrix $G \in \mathbb{R}^{10x10}$ from the outputs of the hidden layer and forming the matrix $D \in \mathbb{R}^{10x4}$ of the desired outputs of the neural network scheduler from the entries in Table 5, the weights W for the linear neurons in the output layer are found by solving (38), and examining the values, (39) is true. Figures 38 to 41 show the outputs of the neural network scheduler, which are the 4 parameter values sent to the adaptive controller. For comparison, the straight line approximations between the operating points are included in the

four figures. As can be seen, $z_i(k) = d_i(k)$ for $1 \leq k \leq 11$ and $1 \leq i \leq 4$ satisfy specification (i). In the regions nearby the operating points, the adaptive controller parameter values specified by the neural network scheduler are close to those specified by Table 5 satisfying specification (ii) for very thin and wide boxes. In regions between operating points, a swift yet smooth transition occurs between the value of Table 5, and specification (iii) is met. Between 4.0 and 4.5 in Figure 39, as well as in other places, specification (iii) requires a straight line, and this is clearly satisfied. Comparing these interpolation curves to those developed in Examples 3.1.1 and 3.1.2, the ones derived here better satisfy the design specifications.

For comparison, the slopes are changed from $s_k = 75$ to $s_k = 10$ for $1 \leq k \leq 10$. The outputs of the gaussian-type nodes g_k for $1 \leq k \leq 11$ are plotted in Figure 42. The requirements of (35) are not satisfied, yet those of (36) and (37) are. The resulting set of interpolation curves are shown in Figures 43 to 46. Specification (i) is satisfied since (38) was solved uniquely. The areas around the operating points do not form plateaus and only satisfy specification (ii) for very small ε_k and very large δ_k. Specification (iii) is violated in numerous places, for example in Figure 44 between 2.25 and 2.5. In addition, specification (iii) requires a straight line between 4.0 and 4.5 in Figure 46, as well as other places, and this is not met.

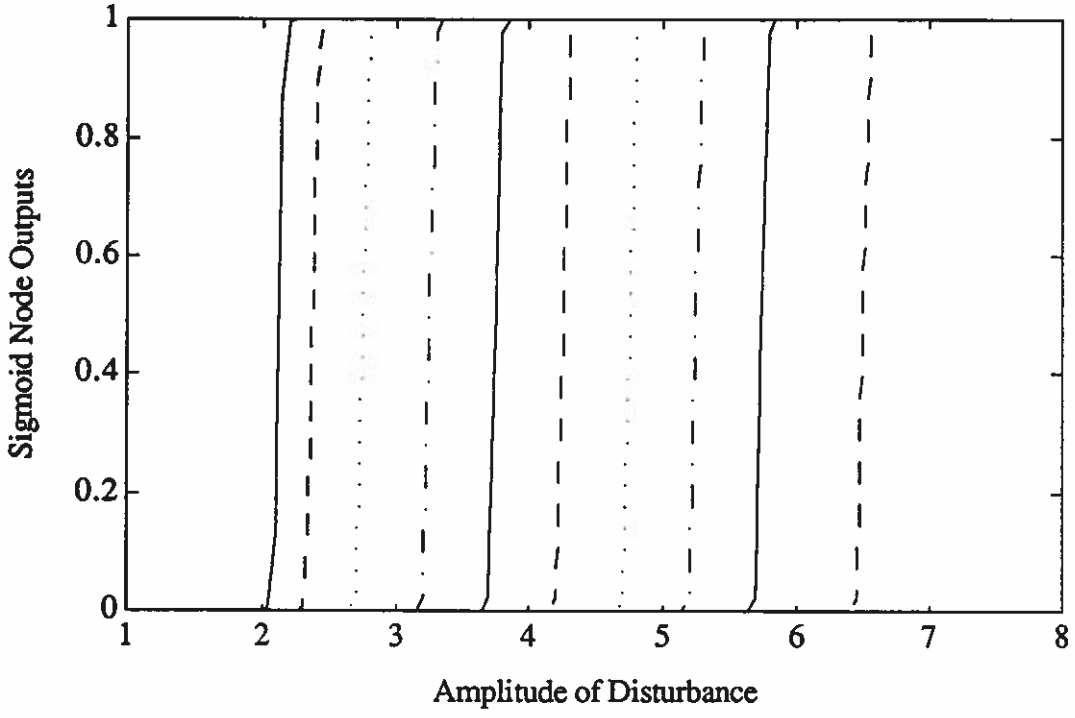

Figure 35 Sigmoid neuron outputs with $s_k = 75$ for Example 3.2.1.

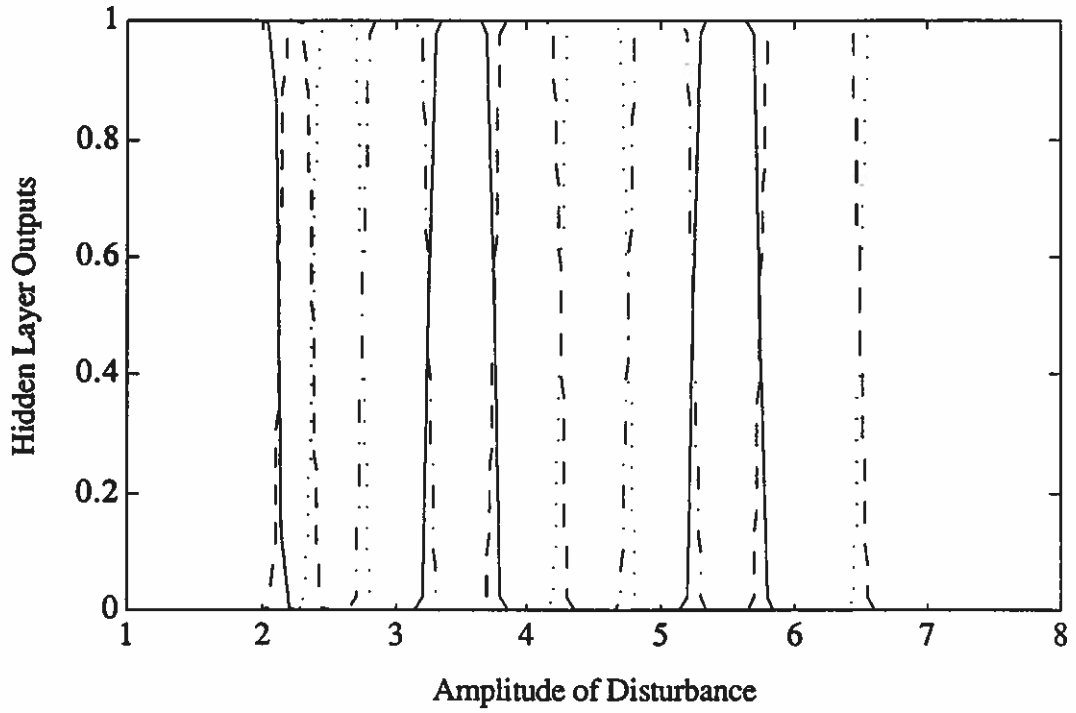

Figure 36 Gaussian-type node outputs with $s_k = 75$ for Example 3.2.1.

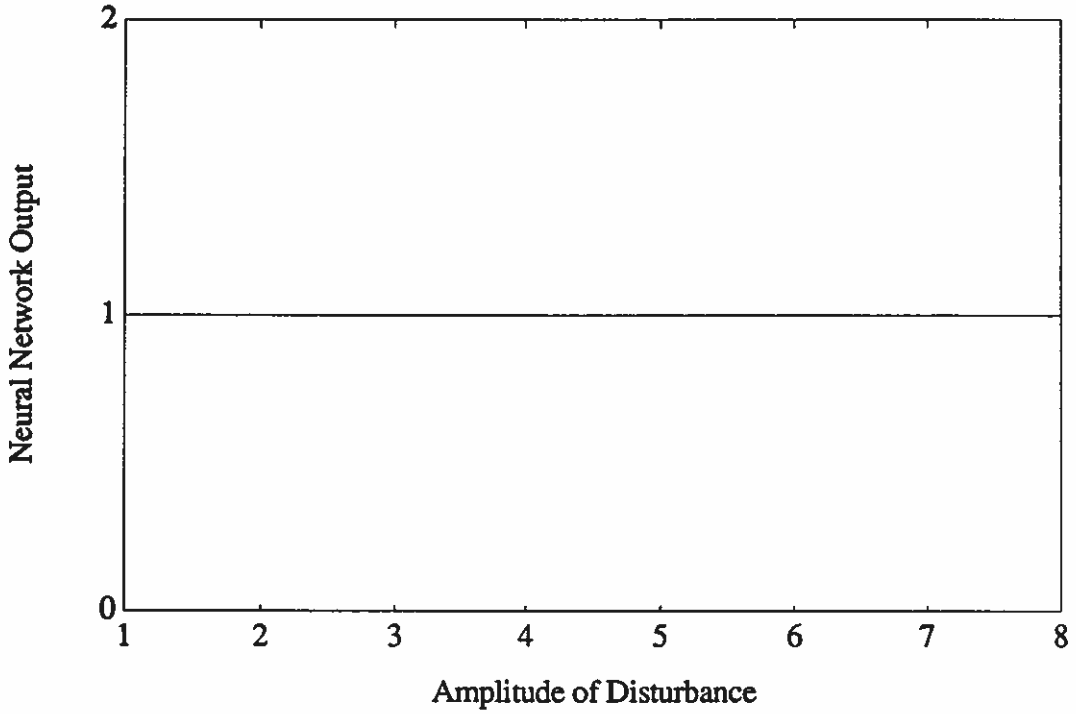

Figure 37 Summation of hidden layer gaussian-type nodes with $s_k = 75$ for Example 3.2.1

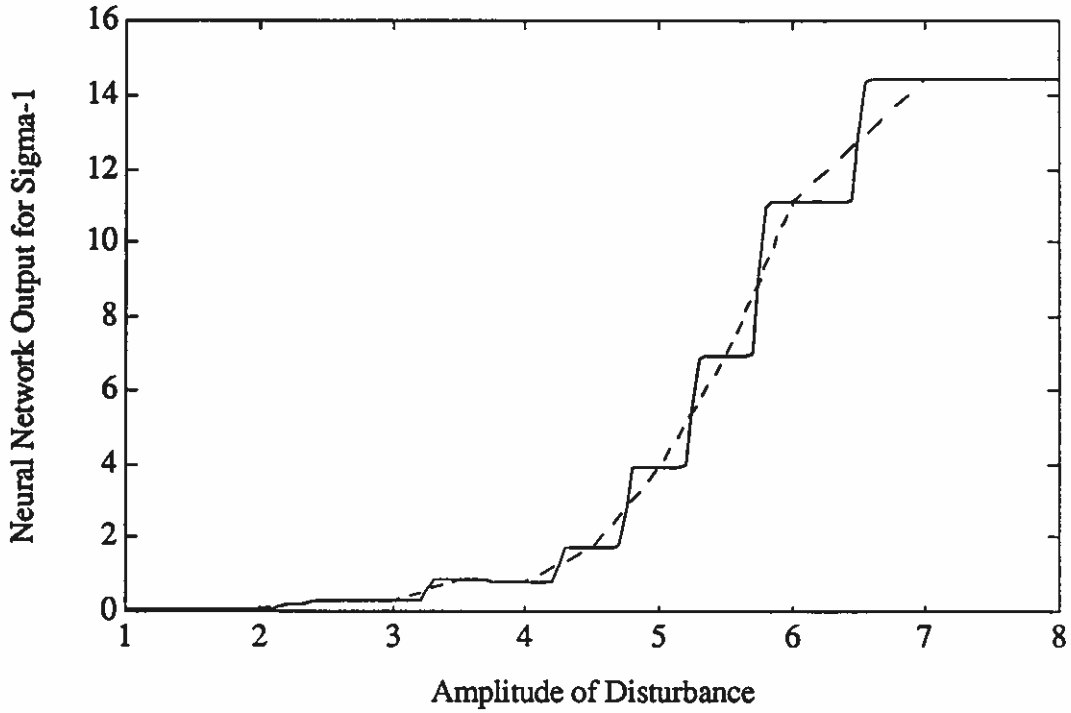

Figure 38 Neural network output for σ_1 with $s_k = 75$ for Example 3.2.1.

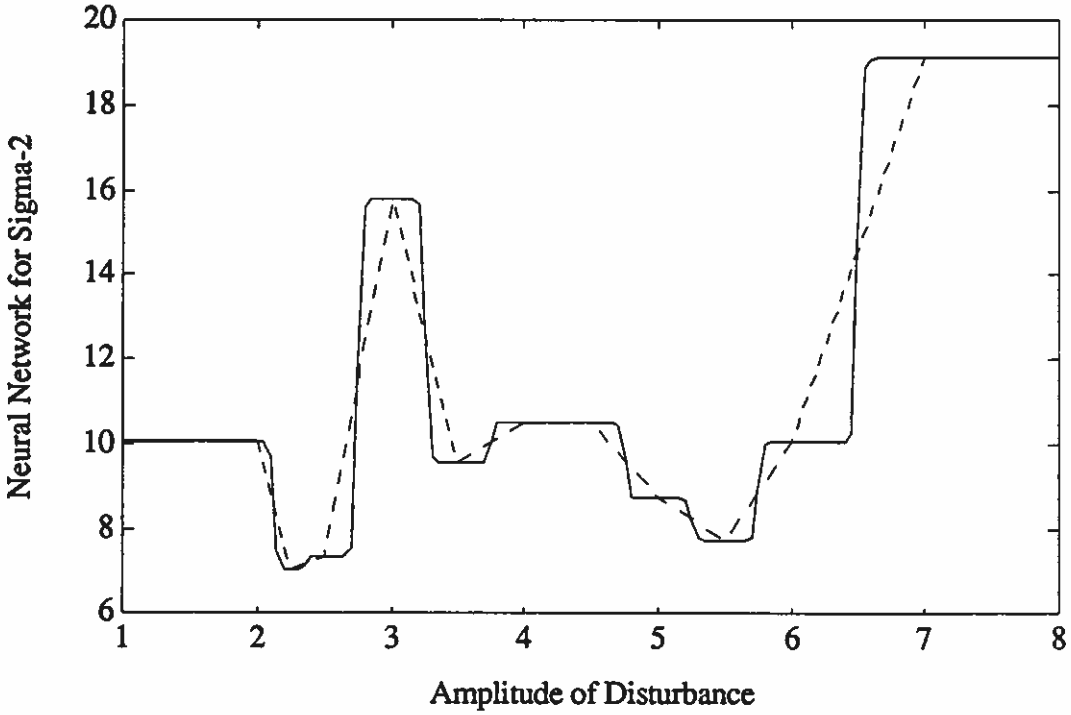

Figure 39 Neural network output for σ_2 with $s_k = 75$ for Example 3.2.1.

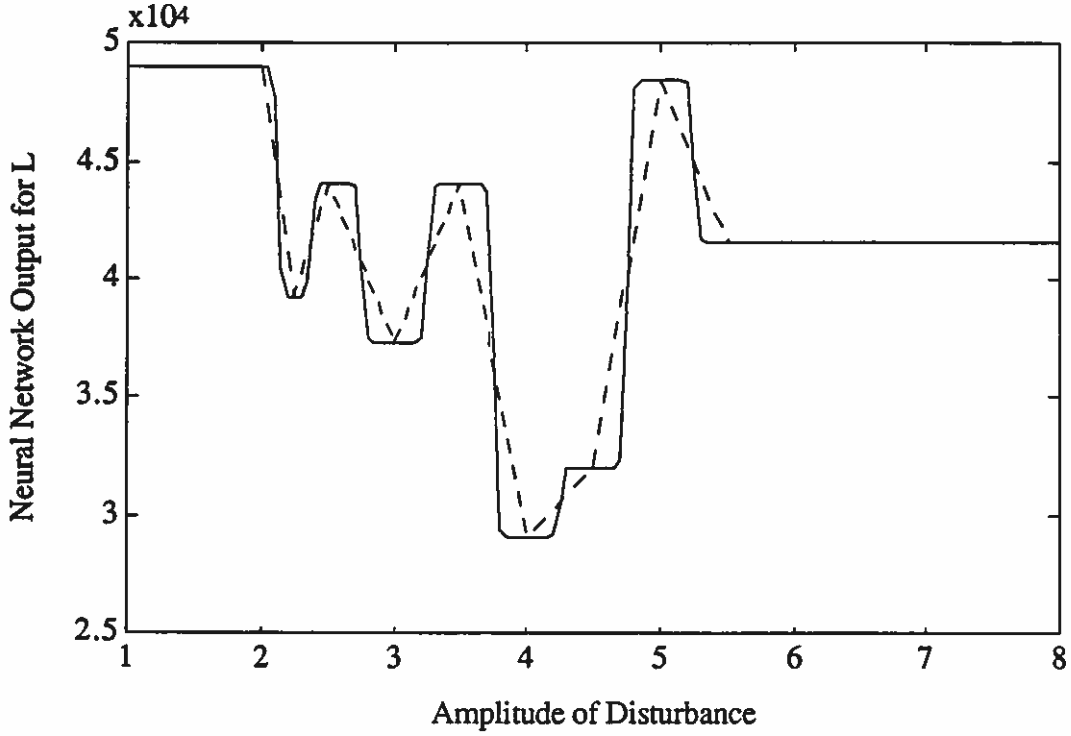

Figure 40 Neural network output for L with $s_k = 75$ for Example 3.2.1.

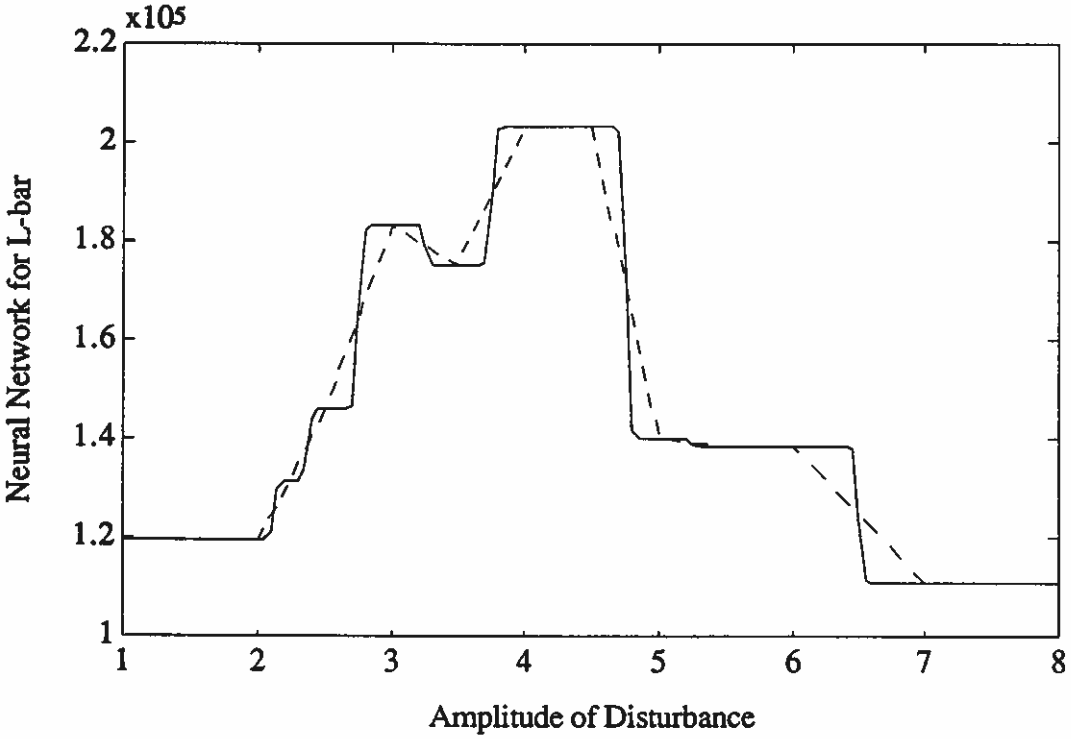

Figure 41 Neural network output for $\bar{L}$ with $s_k = 75$ for Example 3.2.1.

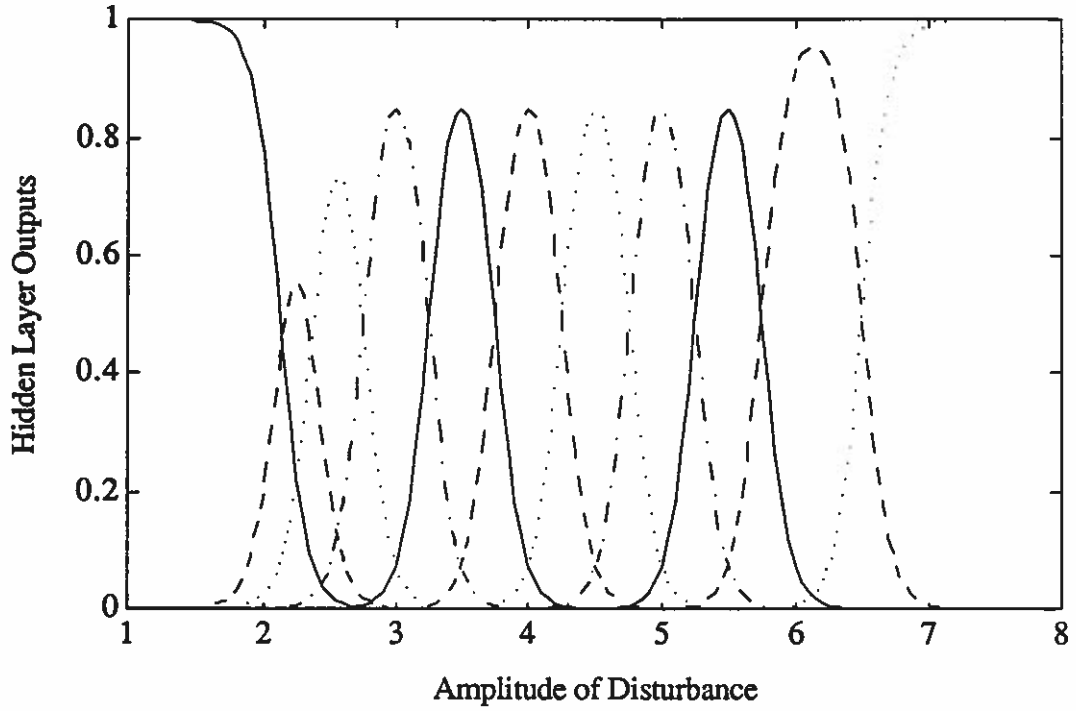

Figure 42 Gaussian-type node outputs with s_k =10 for Example 3.2.1.

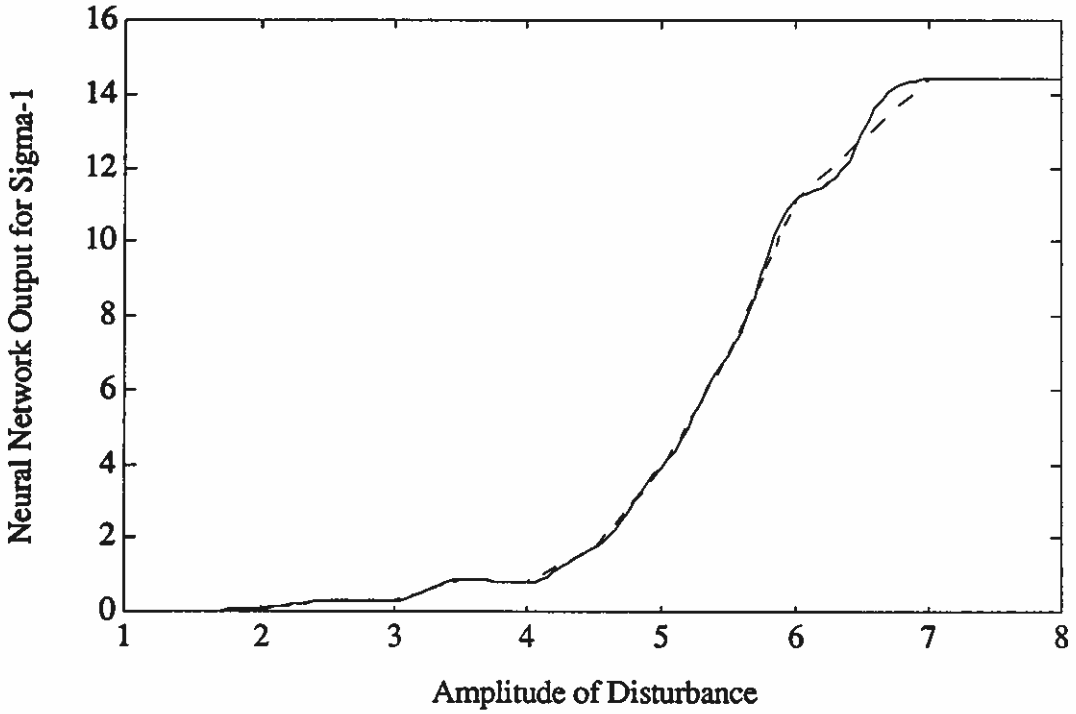

Figure 43 Neural network output for σ_1 with s_k = 10 for Example 3.2.1.

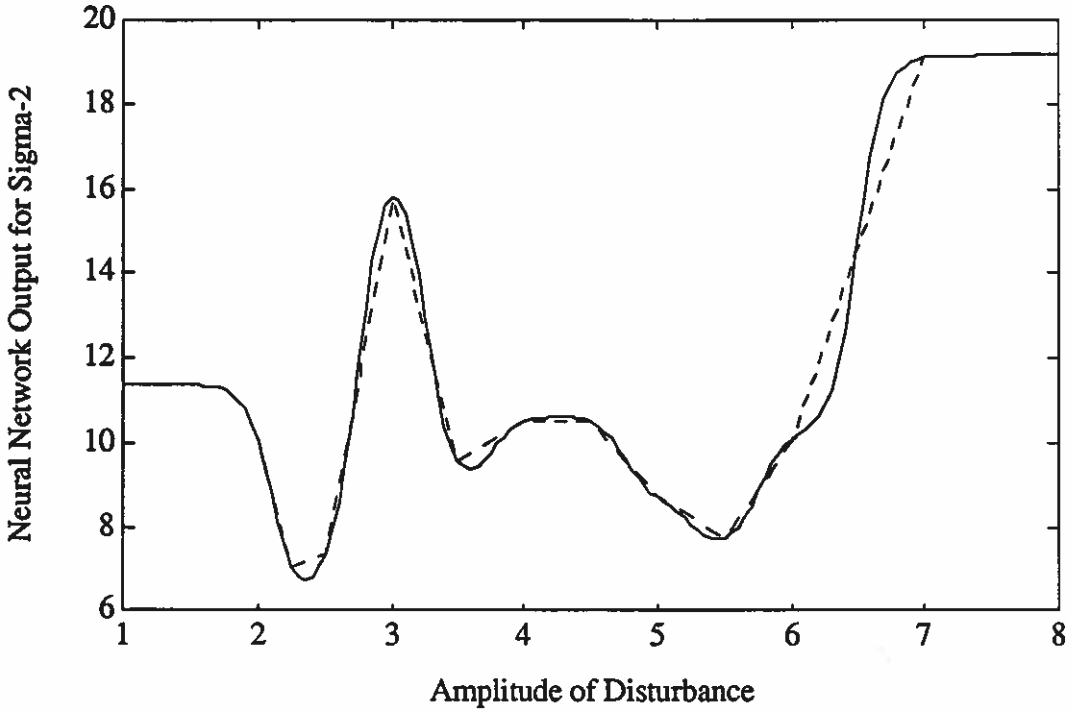

Figure 44 Neural network output for σ_2 with $s_k = 10$ for Example 3.2.1.

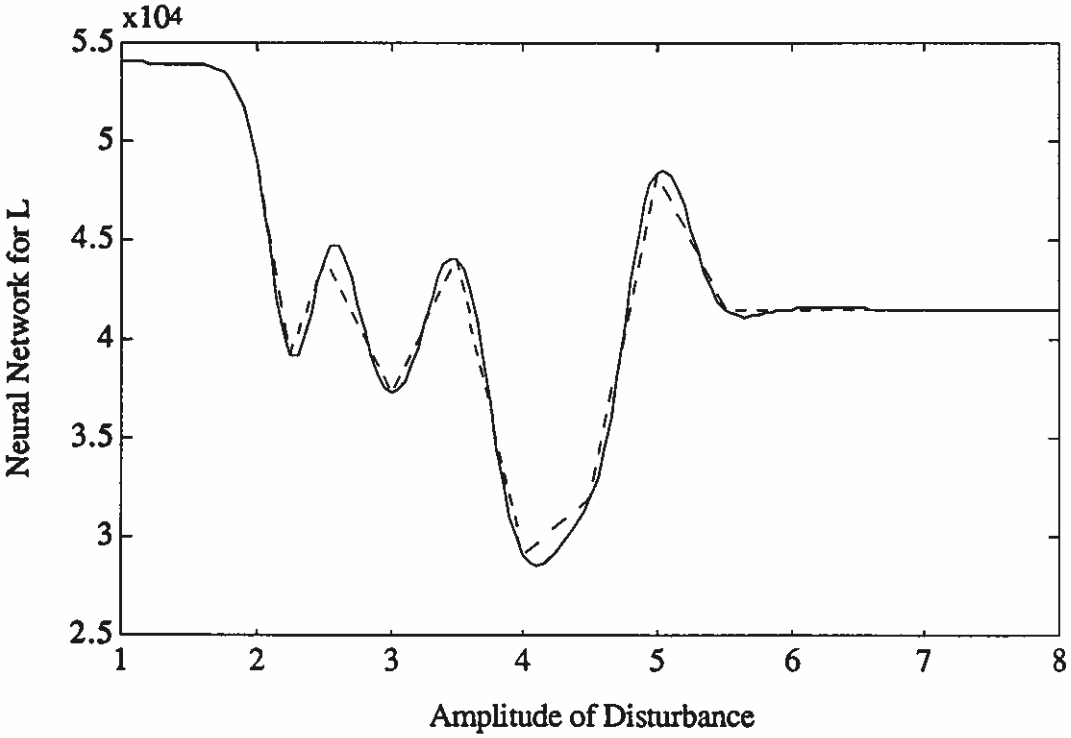

Figure 45 Neural network output for L with $s_k = 10$ for Example 3.2.1.

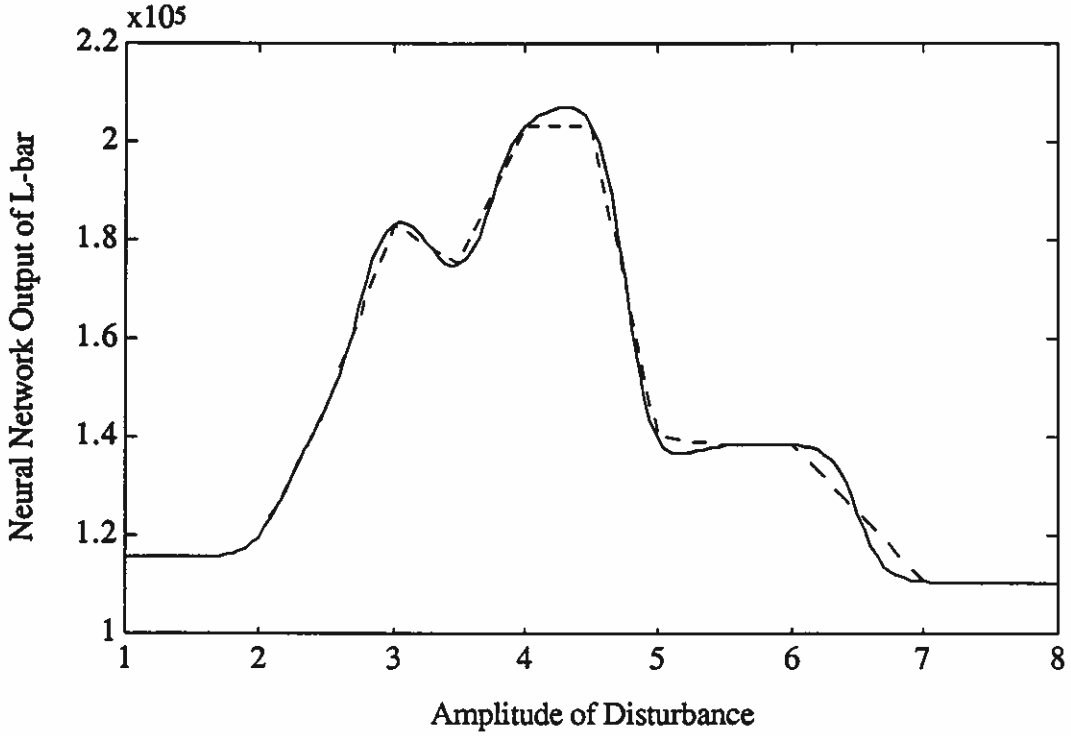

Figure 46 Neural network output for $\bar{L}$ with $s_k = 10$ for Example 3.2.1.

Example 3.2.2:

In another of the experiments described in [Peek90], a command step input is applied as the reference signal of the system, and the adaptive controller is required to follow a highly damped second-order reference model. In Table 3, the parameters found for the controller for different step amplitudes are repeated. In Example 3.1.3, interpolation curves between these points were formed using a gaussian neural network. The goal here is to construct a sigmoid neural network such that a smooth interpolation is achieved between the 8 operating points. Since only the amplitude of the step input specifies which values the adaptive controller requires for its 4 parameters, the results of Section 2.1 for the single-parameter scheduler are applicable.

With the vector of scheduler operating points, 7 sigmoid neurons are formed with centers as in (34):

$$\mathbf{c} = [1.05 \quad 1.125 \quad 1.175 \quad 1.225 \quad 1.275 \quad 1.325 \quad 1.375]'.$$

With the slopes set to $s_k = 400$ for $1 \le k \le 7$, the matrices $\mathbf{G} \in \mathbb{R}^{7x7}$ and $\mathbf{D} \in \mathbb{R}^{7x4}$ are formed, and the unique system in (38) is solved for $\mathbf{W}$, which can be approximated by (39). The outputs of the gaussian-type nodes are plotted in Figure 47. Notice the asymmetrical gaussian-type node formed for the operating point v(2) = 1.10. For the neural network scheduler formed, Figures 48 to 50 show its outputs, which are values for the adaptive controller parameters σ_1, σ_2, and L. The

plot for $\bar{L}$ is not shown since the designed scheduler in Table 3 specifies that this parameter is 0 for most of the operating points. Specifications (i) and (iii) are satisfied and specification (ii) is satisfied for very thin and wide boxes. These plots can be compared to the interpolation curves developed using the gaussian neural network of Example 3.1.3, and the curves derived here can be seen to better satisfy the design specifications.

Example 3.2.3:

In this example, the application of equations (41) and (42) are illustrated. In Table 6, parameters for specification (ii) for the output σ_2 from Table 5 are presented. Solving (41) and (42) for **W = D**, the lower bounds for s_k are in Table 7. For the slope $s_k = 75$ chosen in Example 3.2.1, most of specification (ii) can be satisfied. The entry -5.1986 implies that any positive s_k satisfies the bound of (41). The entries "x" imply that specification (ii) is not applicable since d(6) = d(7). As another illustration, the parameters for specification (ii) for the output σ_2 from Table 3 are presented in Table 8. Solving (41) and (42) for **W = D**, the lower bounds for s_k are in Table 9, and the choice of $s_k = 400$ for $1 \le k \le 7$ in Example 3.2.2 satisfies most of the bounds.

Table 6 Specification (ii) for σ_2 of Table 5.

k	v(k)	ε_{k+}	ε_{k-}	d(k)	γ_{k+}	γ_{k-}
1	2.0	0.1	0.1	10.05	0.5	0.5
2	2.25	0.1	0.05	7.04	0.01	0.3
3	2.5	0.1	0.01	7.35	0.2	0.2
4	3.0	0.15	0.05	15.79	0.05	0.1
5	3.5	0.05	0.05	9.556	0.2	0.2
6	4.0	0.2	0.2	10.48	0.001	0.001
7	4.5	0.2	0.2	10.48	0.001	0.001
8	5.0	0.1	0.1	8.7	0.1	0.1
9	5.5	0.1	0.1	7.73	0.05	0.04
10	6.0	0.45	0.2	10.05	0.01	0.01
11	7.0	0.45	0.45	19.1	0.01	0.01

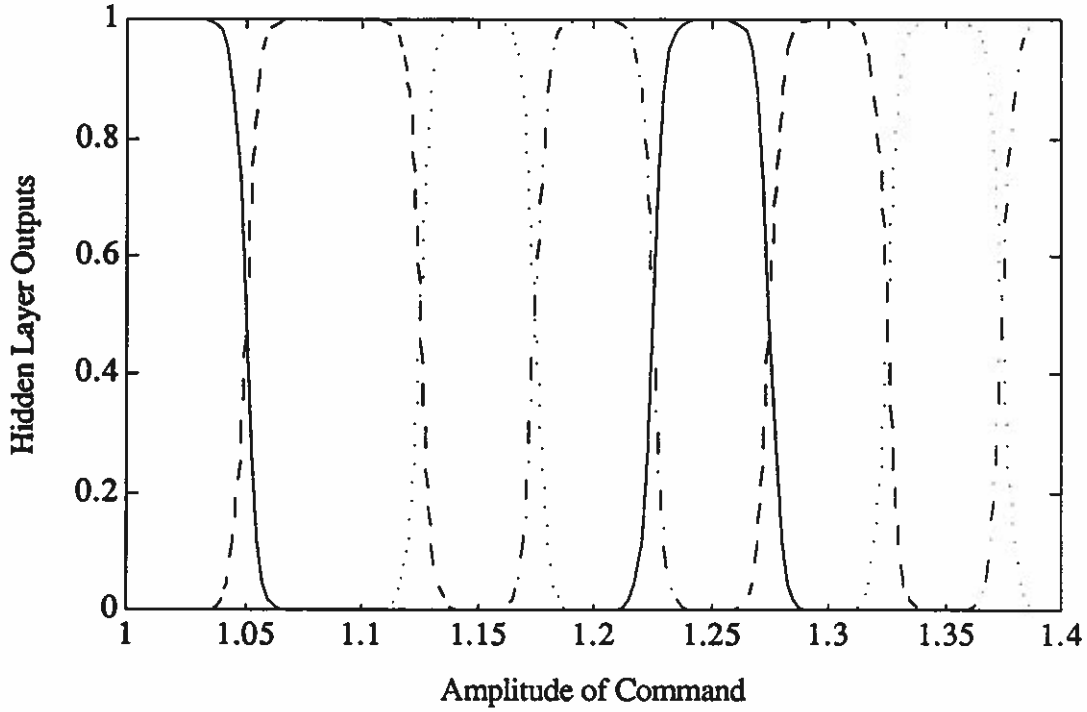

Figure 47 Gaussian-type node outputs with s_k =400 for Example 3.2.2.

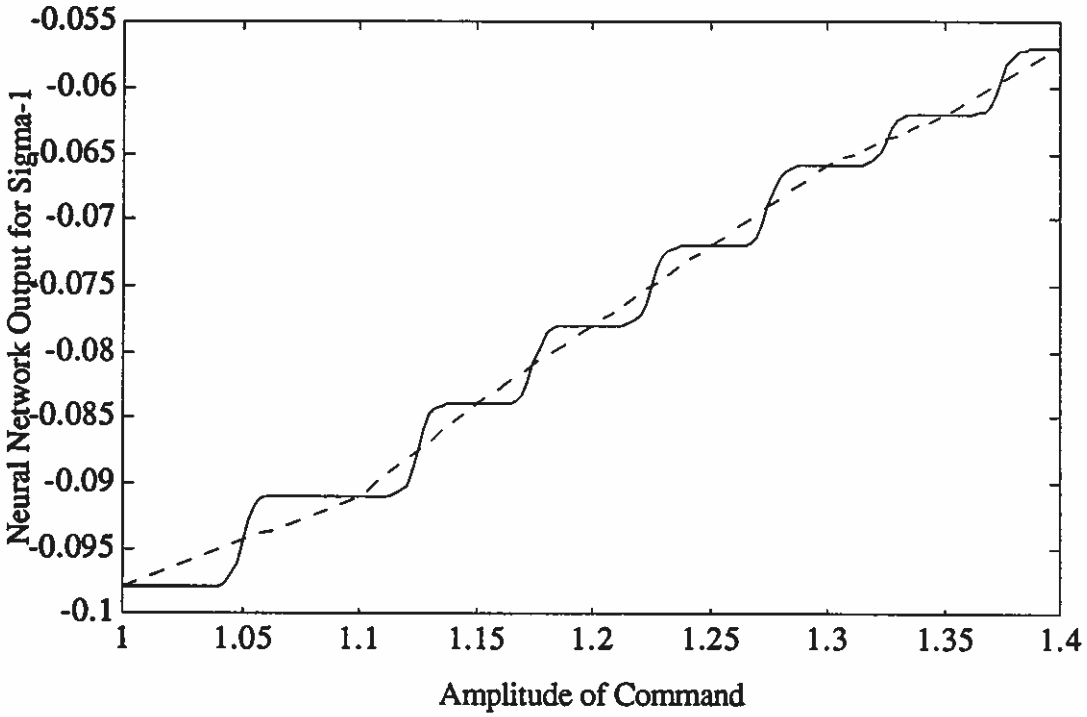

Figure 48 Neural network output for σ_1 with s_k = 400 for Example 3.2.2.

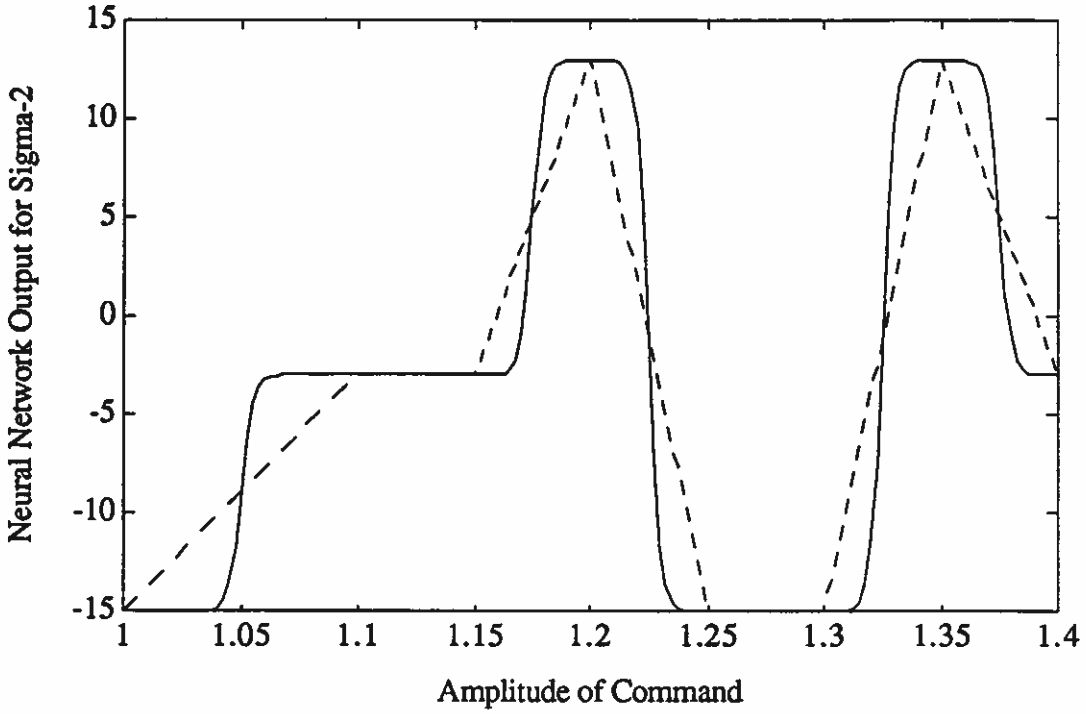

Figure 49 Neural network output for σ_2 with $s_k = 400$ for Example 3.2.2.

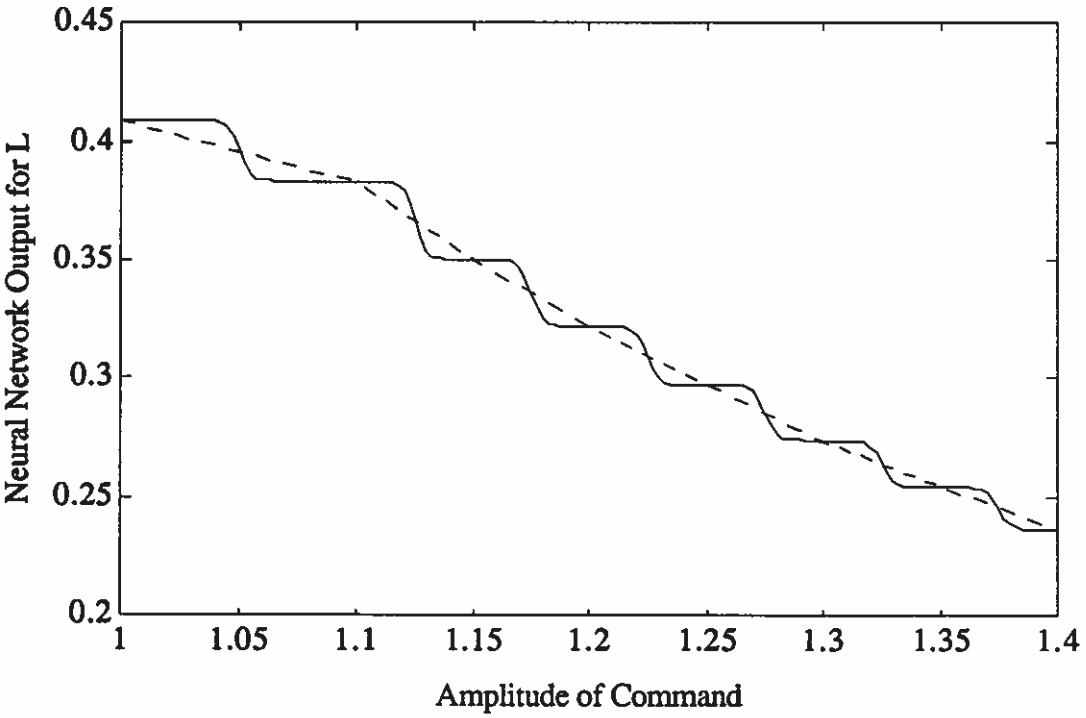

Figure 50 Neural network output for L with $s_k = 400$ for Example 3.2.2.

Table 7 Bounds for σ_2 in Table 5.

k	Bound from (42)	Bound from (41)
1	76.0504	64.5372
2	-5.1986	136.0479
3	22.1182	24.7896
4	17.0342	41.1643
5	136.7681	6.5010
6	x	x
7	18.8092	149.6761
8	19.4157	14.4222
9	108.8484	25.4367
10	136.1366	136.1366

Table 8 Specification (ii) for σ_2 of Table 3.

k	v(k)	ε_{k+}	ε_{k-}	d(k)	γ_{k+}	γ_{k-}
1	1.00	0.04	0.01	-15	0.5	0.5
2	1.1	0.01	0.015	-3	0.3	0.01
3	1.15	0.01	0.01	-3	0.2	0.2
4	1.2	0.015	0.005	13	0.1	0.05
5	1.25	0.005	0.005	-15	0.2	0.2
6	1.3	0.02	0.02	-15	0.05	0.01
7	1.35	0.02	0.02	13	0.01	0.01
8	1.4	0.01	0.01	-3	0.1	0.2

Table 9 Bounds for σ_2 in Table 3.

k	Bound from (42)	Bound from (41)
1	104.7	313.5
2	x	x
3	253.5	291.3
4	246.7	563.1
5	x	x
6	1587.4	1587.4
7	291.3	1475.4

Example 3.2.4:

In [Hackney77], linear models of a F100 engine are developed for various flight points based on altitude and mach number. In Table 4, 6 of these flight points are listed with fictitious controller parameters, and in Figure 32 these six are diagrammed. In Example 3.1.4, two interpolation curves were developed using gaussian neural networks. It is desired here to form an interpolation curve between these flight points using a sigmoid neural network. Since the given scheduler desired is a multi-parameter one, the results of Section 2.2 are applicable. Since the operating points are not equidistant, it is decided to add 6 extra pseudo-operating points such that equidistance is achieved, and p = 12. Now,

$$\mathbf{V}' = \begin{bmatrix} 10 & 10 & 10 & 10 & 20 & 20 & 20 & 20 & 30 & 30 & 30 & 30 \\ 0.50 & 0.75 & 1.00 & 1.25 & 0.50 & 0.75 & 1.00 & 1.25 & 0.50 & 0.75 & 1.00 & 1.25 \end{bmatrix}$$

and

$$\mathbf{d} = [1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 5 \quad 6 \quad 6]',$$

which are the same as the pseudo-desired outputs added in the implementation of Example 3.1.4. The resulting vectors of operating points are

$$\mathbf{v}_1 = [10 \quad 20 \quad 30]'$$

and

$$\mathbf{v}_2 = [0.50 \quad 0.75 \quad 1.00 \quad 1.25]',$$

where r = 1 corresponds to the altitude and with r = 2 corresponds to the mach number. The equivalent to (44) for this is example is

$$z(\mathbf{u}) = \sum_{k_1=1}^{2} \sum_{k_2=1}^{3} w_{k_1 k_2} \, g_{1k_1}(u_1) \, g_{2k_2}(u_2).$$

Using (48), the vectors of centers are

$$\mathbf{c}_1 = [15 \quad 25]'$$

and

$$\mathbf{c}_2 = [0.625 \quad 0.875 \quad 1.125]'.$$

The slopes are chosen as $s_{1k_1} = 20$ for $1 \le k_1 \le 2$ and $s_{2k_2} = 400$ for $1 \le k_2 \le 3$. The weights $\mathbf{w}$ are found by solving (9), and the interpolation curve produced is shown in Figure 51. The lowest corner corresponds to the point (8, 0.4), the most left corner to (32, 0.4), and the most right corner to (8, 1.35). Specifications (i) and (iii) are clearly satisfied, and specification (ii) is satisfied for small thin boxes for a rectangular region around the operating points.

Instead of adding extra operating points such that equidistance is achieved, it is decided to hand pick the generalization regions for the interpolation curve, which are shown in Figure 52 and are assigned values according to Table 4. The equivalent to (51) for this example is

$$z(\mathbf{u}) = \sum_{k=1}^{6} w_k \, g_{1k}(u_1) \, g_{2k}(u_2)$$

Only 4 sigmoids are needed to form the 6 regions, and the vectors of centers are

$$c_1 = [20]$$

and

$$c_2 = [0.625 \quad 0.875 \quad 1.125]'$$

where r = 1 corresponds to the altitude and r = 2 corresponds to the mach number. By choosing $s_{1k} = 20$ and $s_{2k} = 400$ for $1 \leq k \leq 6$ and by solving (38) for w, the interpolation curve formed is shown in Figure 53. Once again, the lowest corner corresponds to the point (8, 0.4), the most left corner to (32, 0.4), and the most right corner to (8, 1.35). Specifications (i) and (iii) are clearly satisfied, and specification (ii) is satisfied for small thin boxes for a rectangular region around the operating points. This interpolation curve is only one possible curve, as is the one in Figure 51, and many other are possible. In comparing the curves in Figures 51 and 53 to the curve developed using a gaussian neural network in Figure 33, the "valleys" present in the previous work are no longer evident here, and the number of hidden layer nodes is considerably reduced using the implementation shown in Figure 53. Specification (iii) is also clearly met using the design methods of this chapter. In comparing the two interpolation curves developed for this example with the one produced using Moody's method in Figure 34 of Example 3.2.4, the curves here are better able to satisfy specification (ii).

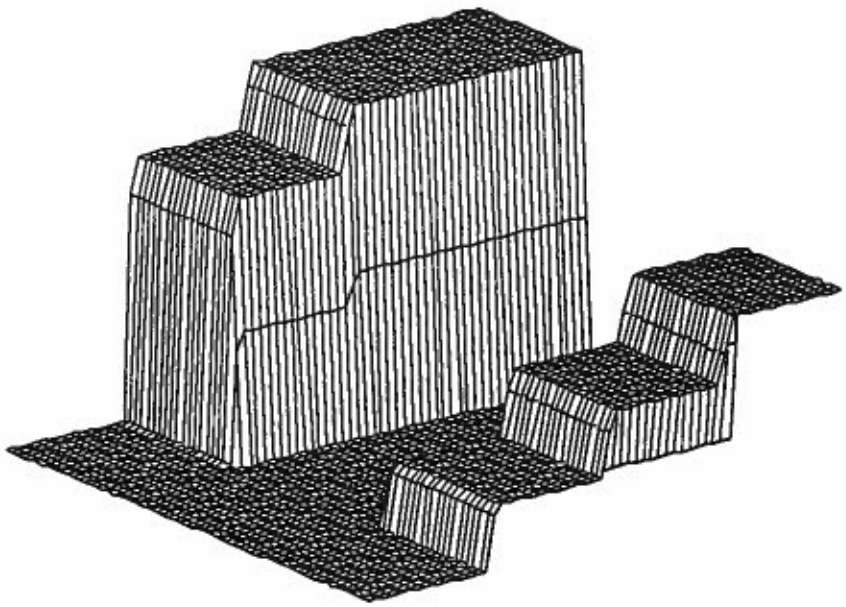

Figure 51 Neural network output for equidistant operating points.

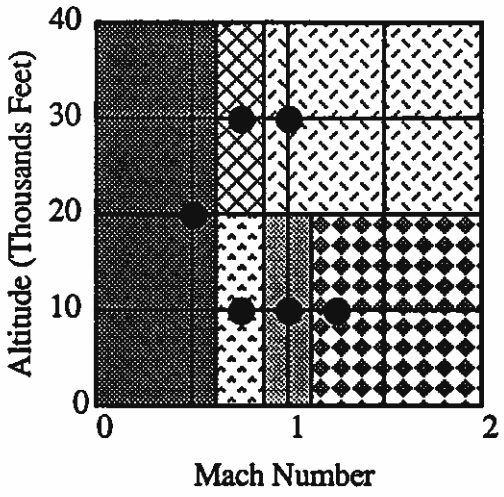

Figure 52 Selected localized regions for the F100 flight points of Example 3.2.4.

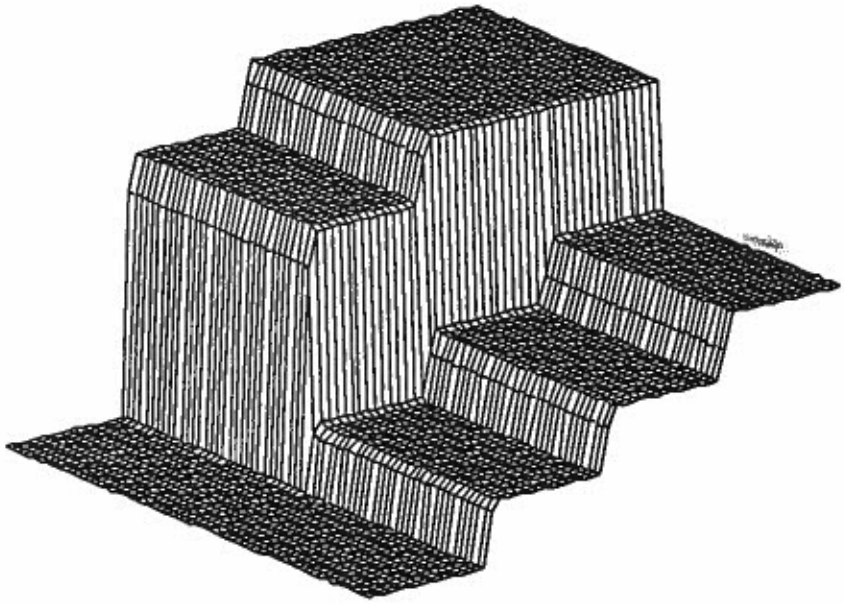

Figure 53 Neural network output for selected localized regions.

4 CONCLUDING REMARKS

Using neural networks with either gaussian nonlinearities or gaussian-type nonlinearities formed by combining sigmoid functions, two methods are described here for scheduler implementation. Given the operating points and the control law parameter values to be sent to the controller, several specifications are made concerning the shape of the interpolation curve. The neural network implementations provide two classes of curves to satisfy these specifications, which is accomplished by designing the generalization behavior of the neural network. The first method using gaussian nonlinearities works well for a designed scheduler with one-dimensional equidistant operating points but not as well when the operating points are not equidistant. For designed schedulers with multi-dimensional operating points, the method has some drawbacks: a potentially large number of hidden layer neurons and the elliptical shape of the multi-dimensional gaussian function contributing to unwanted "valleys" in the interpolation curve. The second neural network implementation using sigmoid nonlinearities eliminates these problems; the method works well for a designed scheduler with either equidistant or non-equidistant operating points and for either one-dimensional or multi-dimensional operating points. However, the problem of a potentially large number of hidden layer neurons still exists for the multi-parameter case.

The results reported in this paper also appear in [Sartori91].

5 REFERENCES

[Antsaklis92] Antsaklis P.J., Sartori M.A., "Neural Networks in Control Systems," *Systems and Controls Encyclopedia, Supplement II*, to appear.

[Farrell90] Farrell J., Goldenthal B., Govindarajan K., "Connectionist Learning Control Systems: Submarine Depth Control," *Proceedings of the 1990 Conference on Decision and Control,* pp. 2362-2367, 1990.

[Gonzalez90] Gonzalez T.F., "Covering a Set of Points with Fixed Size Hypersquares and Related Problems,"*Proceedings of the 1990 Annual Allerton Conference on Communication, Control, and Computing*, pp. 838-847, 1990.

[Hackney77] Hackney R.D., Miller R.J., "Engine Criteria and Models for Multivariable Control System Design," *Proceedings of the 1977 International Forum on Alternatives for Multivariable Control*, pp. 14-28, 1977.

[Huang89] Huang Q., Graupe D., Huang Y.-F., Liu R.-W., "Identification of Firing Patterns of Neuronal Signals," *Proceedings of the 1989 Conference on Decision and Control*, pp. 266-271, 1989.

[Leonard90] Leonard J.A., Kramer M.A., "Classifying Process Behavior with Neural Networks: Strategies for Improved Training and Generalization," *Proceedings of the 1990 American Control Conference*, pp. 2478-2483, 1990.

[Moody88] Moody J., Darken C., "Learning with Localized Receptive Fields," *Proceedings of the 1988 Connectionist Models Summer School*, Carnegie-Mellon University, pp. 133-143, 1988.

[Moody89] Moody J., Darken D., "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, vol. 1, pp. 281-294, 1989.

[Peek90] Peek M.D., Antsaklis P.J., "Parameter Learning for Performance Adaptation," *IEEE Control Systems Magazine*, vol. 10, no. 7, pp. 3-11, December 1990.

[Poggio90] Poggio T., Girosi F., "Networks for Approximation and Learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481-1497, Sept. 1990.

[Sartori91] Sartori M.A., *Feedforward Neural Networks and Their Application in the Higher Level Control of Systems*, Ph.D. Dissertation, Department of Electrical Engineering, University of Notre Dame, April 1991.

[Yao90] Yao S.C., Zafiriou E., "Control System Sensor Failure Detection via Networks of Localized Receptive Fields," *Proceedings of the 1990 American Control Conference*, pp. 2472-2477, 1990.