Proceedings of the 1991 IEEE International Symposium
on Intelligent Control
13 - 15 August 1991, Arlington, Virginia, U.S.A.

A GAUSSIAN NEURAL NETWORK IMPLEMENTATION FOR CONTROL SCHEDULING

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

ABSTRACT

Using neurons with gaussian nonlinearities, a neural network is designed to implement a control law scheduler. For the implementation discussed here, the neural network is supplied information about existing operating conditions and then responds by supplying control law parameter values to the controller. 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 neural network implementation performs best when the operating points are equidistant and has some drawbacks when used to implement multi-parameter schedulers.

1 INTRODUCTION

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. The gaussian neural network presented here for satisfying the specifications of the control law scheduler performs best when the operating points are equidistant and has some disadvantages when used to implement multi-parameter schedulers.

Of the numerous curves that satisfy the specifications for the interpolation between the operating points, the neural network design procedure presented in this paper represents one possible class. Clearly, there exist other methods besides the one 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 approach described here instead of one of these schemes is that, in addition to a straight forward design scheme, 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.

In Section 2, the type of scheduler implementation considered in this paper is defined. The implementation using a gaussian neural network is presented for both single-parameter and multi-parameter operating points in Sections 2.1 and 2.2, respectively, and discussed in terms of equidistant and non-equidistant operating points in Sections 2.1.1 and 2.1.2, respectively. In Section 2.1.3, a comparison to other neural network design methods is included. Finally, examples demonstrating the neural network scheduling implementation presented in this paper are supplied in Section 3.

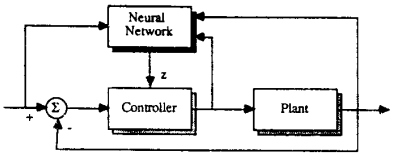

Figure 1 Neural network used as a scheduler.

2 GAUSSIAN NEURAL NETWORK IMPLEMENTATION

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.

2.1 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 $u \in [v(j) - \epsilon_{j-}, v(j) + \epsilon_{j+}]$ and $u \ne v(j)$, 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.

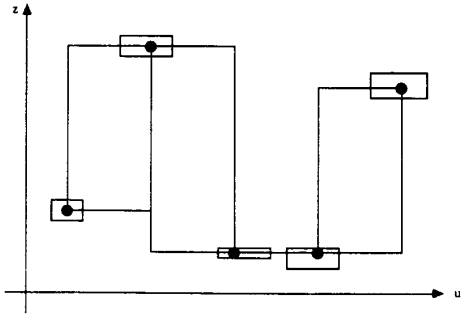

Figure 2 Illustration of interpolation curve specifications.

With $\mathbf{z} \in \mathbb{R}^{n \times 1}$, an individual output of the neural network is described by

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

where $1 \le i \le n$, 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 $\mathbf{W} \in \mathbb{R}^{h \times n}$ 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 $u \in \mathbb{R}^{1 \times 1}$. The output of the k^{th} gaussian neuron in the neural network's hidden layer is described by

CH3019-7/91/0000-0400$01.00 ©1991 IEEE

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

and

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

where $1 \le k \le h$, $g:\mathbf{R} \rightarrow \mathbf{R}$ 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 $\mathbf{s} \in \mathbf{R}^{hx1}$ denote the vector of widths, and $\mathbf{c} \in \mathbf{R}^{hx1}$ 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 $\mathbf{W}$, $\mathbf{s}$, $\mathbf{c}$, 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.

2.1.1 Design with Equidistant Operating Points: For the first case, the operating points v(j) for $1 \le j \le p$ 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 $1 \le k \le h$. 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 $k \ne j$ and $k \ne j+1$ 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

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

for $1 \le f \le h$ 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 $\varepsilon_{k-} \le \Delta_{k-1}$ and $\varepsilon_{k+} \le \Delta_k$. 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 $1 \le k \le h$ 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 $\alpha \ge 1$.

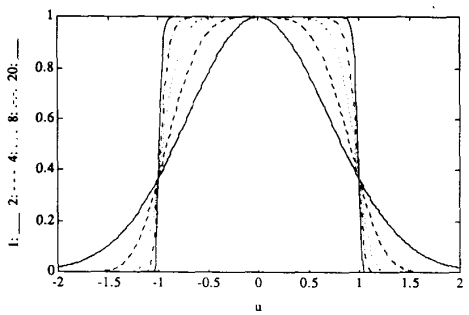

Figure 3 Family of gaussian-type curves for varied α.

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 $\mathbf{G} \in \mathbf{R}^{pxh}$ 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

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

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).

2.1.2 Design with Non-Equidistant Operating Points: For the second case, the operating points v(j) for $1 \le j \le p$ 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

$$F = \frac{1}{2}\sum_{j=1}^{p}\left(1 - \sum_{k=1}^{h} g(x_k(v(j)))\right)^2 + \frac{1}{2}\sum_{j=1}^{q}\left(1 - \sum_{k=1}^{h} g(x_k(u(j)))\right)^2, \tag{10}$$

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. The gradients of (10) are computed with respect to s_k, c_k, or even α, and the parameters of the gaussian neural network are adjusted via an iterative gradient descent procedure. The details of this approach are in [Sartori91].

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 procedure 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.

2.1.3 Comparison to Another Method: Here, the design procedure of [Moody89] and [Moody88] is compared to the approach presented here. 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,

$$s_k = \frac{2}{(c_k - c_{k-1})^2 + (c_{k+1} - c_k)^2} \tag{11}$$

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 (11).

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

$$F = \frac{1}{2} \sum_{j=1}^{p} (d(j) - z(v(j)))^2 \tag{12}$$

which can be compared to the one proposed in (10). In (12), the neural network is assumed to have only one output, while the gaussian neural network here may have more than one.

2.2 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 $v(j) \bullet \mathbf{R}^{mx1}$ denote the operating point vector for $1 \le j \le p$. As described previously, $\mathbf{d}(j) \bullet \mathbf{R}^{nx1}$ and $\mathbf{D} \bullet \mathbf{R}^{pxn}$ 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 $\mathbf{u} \bullet \mathbf{R}^{mx1}$ denote the neural network's input, and let the neural network have multiple outputs denoted by $\mathbf{z} \bullet \mathbf{R}^{nx1}$. 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

$$z_i = \sum_{k=1}^{h} w_{ik}\, g(x_k) \tag{13}$$

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

$$g(x_k) = \exp(-x_k^{\alpha}) \tag{14}$$

and

$$x_k = \sum_{r=1}^{m} s_{kr}(v_r - c_{kr})^2 \tag{15}$$

where $1 \le k \le h$. Let $\mathbf{S} \bullet \mathbf{R}^{hxm}$ denote the matrix of widths, and $\mathbf{C} \bullet \mathbf{R}^{hxm}$ denote the matrix of centers. Compared to the gaussian neurons proposed by [Moody89], the widths s_{kr} in (15) 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 2.1. 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

$$c_{kr} = v_r(k) \tag{16}$$

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

$$s_{kr} = \frac{4g^{-1}(0.5)}{T_r} \tag{17}$$

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 2.1.2 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 (14) and (15). This is illustrated in Example 3.3.

3 EXAMPLES

Example 3.1:

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 $\bar{L}$) 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. Using the results of Section 2.1.1 for the single-parameter scheduler with equidistant operating points of distance T = 0.5, a gaussian neural network is designed to implement the controller scheduler.

Table 1 Initial Disturbances and Parameter Sets.

Amplitude	σ_1	σ_2	L	$\bar{L}$
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)

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

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 4, the localized properties of the gaussian neurons are evident. Forming the matrix $\mathbf{G} \bullet \mathbf{R}^{9x9}$ from the outputs of the hidden layer and forming the matrix $\mathbf{D} \bullet \mathbf{R}^{9x4}$ of the desired outputs of the neural network scheduler from the entries in Table 1, the weights W for the linear neurons in the output layer are found by solving (9). Figure 5 shows the output of the gaussian neural network scheduler for σ_2, and for comparison the straight line approximations between the operating points are included. As can be seen, $z_i(k) = d_i(k)$ for $1 \le k \le 9$ and $1 \le i \le 4$, 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.

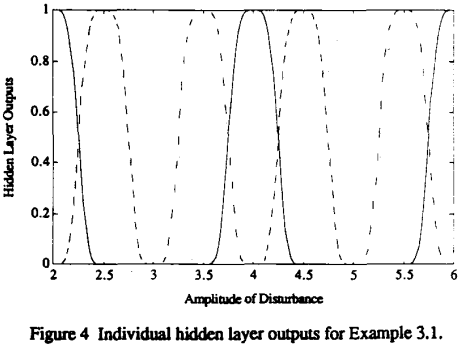

Figure 4 Individual hidden layer outputs for Example 3.1.

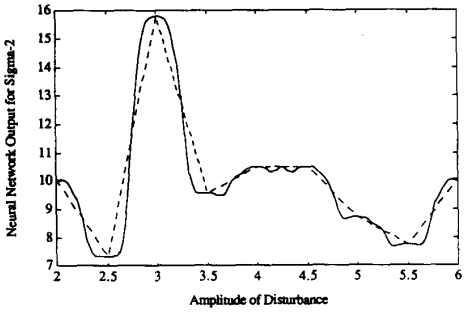

Figure 5 Gaussian neural network output for σ_2 for Example 3.1.

For 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 $1 \le k \le 9$. The weights **W** are found by solving for **W** in (9). In Figures 6, the neural network output for σ_2 is displayed along with the straight line approximation (dotted line). Specification (i) is met due to the solving of (9) for **W**. However, specification (ii) can only be satisfied for small boxes, and specification (iii) is not met at all.

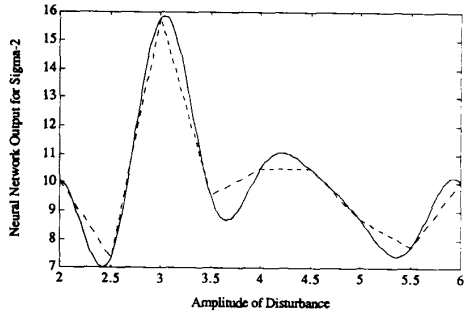

Figure 6 Moody's neural network output for σ_2 for Example 3.1.

Example 3.2:

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 2.1.2 are applicable. To cope with the disturbance amplitudes not being equidistant, extra gaussian neurons are added to the hidden layer, and the vector of centers is

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

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 $1 \le k \le 21$. Adding pseudo-desired outputs, the matrices $\mathbf{G} \bullet \mathbf{R}^{21x21}$ and $\mathbf{D} \bullet \mathbf{R}^{21x4}$ are formed, and the unique linear system of equations in (9) is solved for **W**. The gaussian neural network's output for σ_2 is shown in Figure 7. 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.

Table 2 Extra Initial Disturbances and Parameter Sets.

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

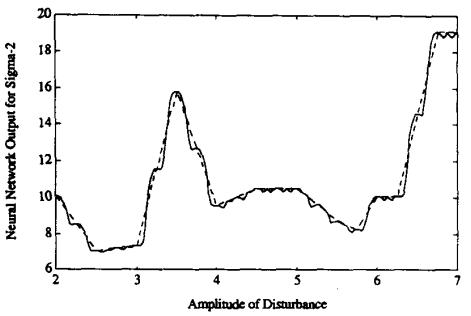

Figure 7 Gaussian neural network output for σ_2 for Example 3.2.

Example 3.3:

In [Hackney77], linear models of a F100 engine are developed for various flight points based on altitude and mach number. In Table 3, 6 of these flight points are listed with fictitious controller parameters, and in Figure 8, 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 2.2 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

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

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 $1 \le k \le 12$. Solving (9) for w, the interpolation curve is shown in Figure 9. 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.

Table 3 Selected Flight Points for F100 Engine.

Amplitude (1K 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

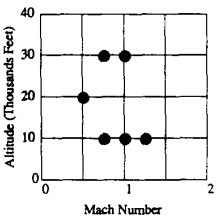

Figure 8 Selected F100 flight points for Example 3.3.

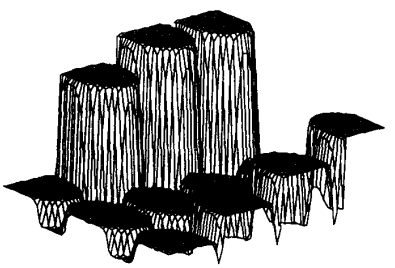

Figure 9 Gaussian neural network output for Example 3.3.

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 10. 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 9, specification (iii) is better satisfied by the curve of Figure 10.

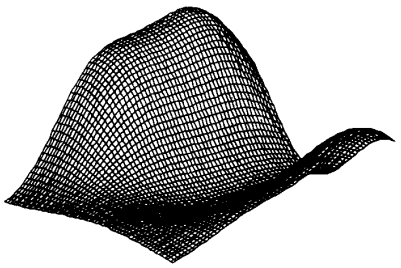

Figure 10 Moody's neural network output for Example 3.3.

4 CONCLUDING REMARKS

Using a neural network with gaussian nonlinearities, a method is 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 implementation provides a class of curves to satisfy these specifications, which is accomplished by designing the generalization behavior of the neural network. The method performs 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. Currently, methods to alleviate these problems are being investigated.

The results reported in this paper also appear in [Sartori91], where the design procedure is explained in more detail and extended examples are described.

Acknowledgement

The authors wish to acknowledge the partial support of the Jet Propulsion Laboratory under Contract No. 957856.

5 REFERENCES

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

[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.

[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.

[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.