Structural Optimization 4, 90–98 (1992)

Structural Optimization

Applications of artificial neural nets in structural mechanics*

L. Berke

Chief Scientist for Structures, NASA Lewis Research Center, Cleveland, Ohio 44135, USA

P. Hajela

Dept. of Mechanical Engineering, Aeronautical Engineering and Mechanics, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Abstract First the fundamentals of neurologically motivated computing are briefly discussed. This is followed by presenting two of the many possible applications in structural mechanics. Both of these are oriented towards structural optimization. In the first mode a neural net model of the structural response is created and then attached to any conventional optimization algorithm. In the second mode a neural net model of an experienced designer is created knowledgeable within a narrow class of structural concepts.

1 Introduction

There has been considerable recent activity in the development and application of a certain class of trainable network paradigms, namely the biologically motivated artificial neural nets (ANN). This upsurge of developmental activities is expected to contribute to the availability of powerful new capabilities in the near future. It appears that only a few structural design applications of neural nets have been presented. Examples dealing with simple oscillators and a beam design can be found in the book by McCauley (1988) and in the paper by Rehak *at al.* (1989). As will be discussed in subsequent sections, there are many other potentially productive applications.

Software for artificial neural network simulations are now available in a wide range of capabilities and can also be accompanied by accelerator boards for PC-s and workstations. Dedicated machines with capabilities based on the concept of virtual nets, an idea similar to virtual memory, can accommodate very large net architectures. The number of artificial neurons, and the number of their connections that can be implemented will be in the millions in the near future.

The history of computational structures technology (CST) is closely linked to the history of utilization of developments of increasing capabilities in computer science and technology. More recently, this includes the utilization of artificial intelligence (AI) capabilities, mostly in the form of rule-based expert systems.

Artificial neural nets are also a class of AI paradigms, and provide new opportunities for applications of computer science developments in CST. This brief note proposes a few potentially profitable applications and presents results of feasibility studies associated with automated structural design. Applications in other areas of structural mechanics are now also being investigated. The fundamental ideas of artificial neural nets are presented to aid in the discussions that follow. The feasibility studies were conducted using well-known "toy" problems. Research is underway to explore the limits of applicability. This includes increasing problem complexity and dimensionality as indicated by the number of input-output variables, and the nature of nonlinearities in their functional dependence.

*Presented at the CISM Course "Shape and Layout Optimization in Structural Design", Udine, Italy, July 1990

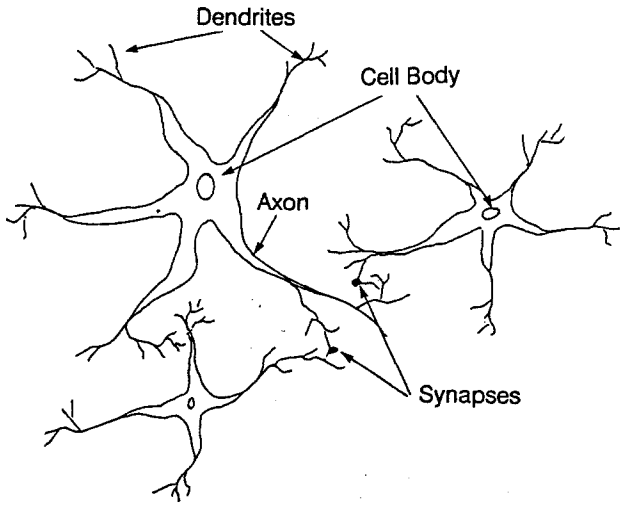
2 Basic concepts of artificial neural nets

Only a brief and incomplete introduction is given here, and the papers by Rumelhart *et al.* (1988) and Pao (1989) are suggested as introductory reading. Many other texts are available in any good technical library, and the body of available publications is increasing very rapidly.

Figure 1a shows a simplistic representation of a biological neuron with the following components of interest: a cell body with the mechanism which controls cell activity, the "axon" that transmits stimulus from one neuron to others, the "dendrites" which also receive electrical signals from connected neurons or from an external source, and the "synapses" which define interconnections and their respective strengths. In a human brain the number of neurons approaches a trillion, each connected to perhaps to tens of thousands of other neurons forming an immense network. Figure 1b shows a small segment of this network in the cerebral cortex.

Artificial neural nets were conceived as very simple models of certain brain activities. Of interest for us here are those aspects of biological neural net activities that are associated with learning, memory, and generalization from accumulated experience. Learned information is thought to be represented by a pattern of synaptic connection strengths that modify the incoming stimuli, strengthening or inhibiting them. When the accumulation of the received stimuli in the neuron reaches a certain threshold, it "fires", sending out an electrical stimulus to all connected neurons. Learning in turn is thought to be associated with the development and retention of a pattern of the connection strengths in various regions of this immense network, somewhat similarly to holograms that also contain complex information in a vast arrangement of simple patterns.

Artificial neural nets simulate the above activities in brain tissue through very simple concepts. An artificial neuron re-



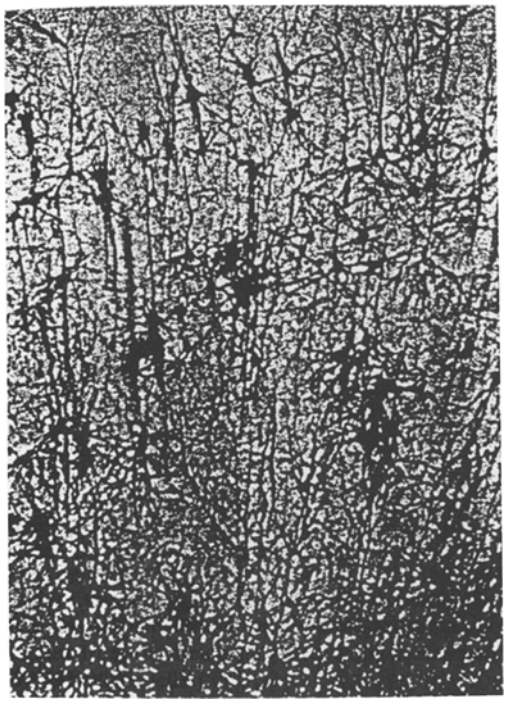
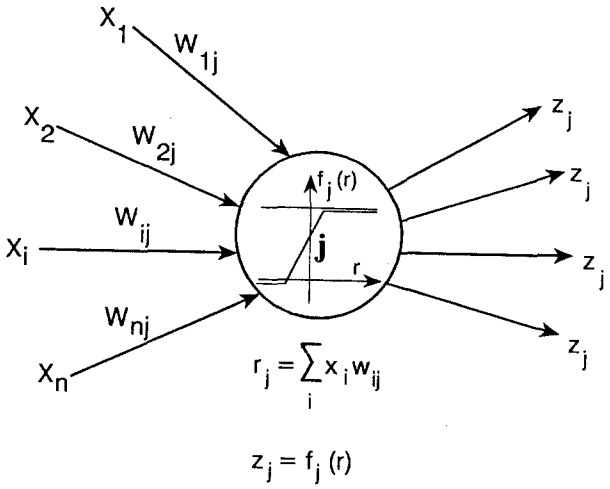


Fig. 1. Biological neuron and neural net



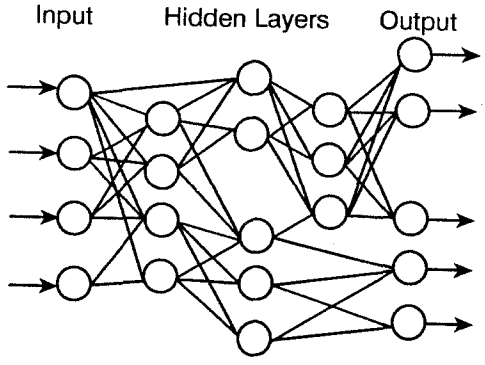


Fig. 2. Artificial neuron and neural net

ceives information labeled x_l through the incoming n connections from other neurons as indicated in Fig. 2a. Such neurons and their connections can be assembled in principle into any architecture of connectivities as indicated in Fig. 2b. The information x_l sent out by the connecting neurons, and received by the j-th neuron of a net, are modified by connection strengths w_{lk}. The j-th neuron performs a summation of the modified information as also indicated in Fig. 2a, and processes the value r_k of the sum through an activation function producing an output z_k. This output is then sent as a stimulus to all connecting neurons and determines, in turn, the activity of those neurons. Figure 3 shows some activation functions, with the sigmoid function being the most popular. More complex neuron activation functions can be devised for various special purposes.

The training of neural nets involves the establishment or evolution of the connection strengths w_{lk} everywhere in the net. Once trained, the network responds to a new input within the domain of its training by "propagating" it through the net and producing an output. This output is an estimate within certain error, of the output that the actual computational or physical process would have produced.

Several neural net paradigms have emerged as a result of over four decades of research, each with its own purpose and capabilities. The particular class of neural nets that are of interest to us here fall in the category of "feed forward" nets because the input data given to the network is propagated forward towards the output nodes. The "delta-error backpropagation" algorithm (see Rumelhart *et al.* 1988; Pao 1989) is used usually for its "supervised" learning. It is essentially a special purpose steepest descent algorithm to adjust the w_{lk} connection strengths, and other additional internal parameters that are sometimes added to increase flexibility, to reproduce the output of given input-output training sets within a required error tolerance. In principle other optimization methods can also be used, and the development of efficient learning algorithms is an active area of research.

Most currently available neural net capabilities are simulations of the distributed parallel processing concept on serial machines, and such simulations were also used in this study. Neural nets present premier applications for parallel machines or for the developments of special purpose hardware. These approaches are all being investigated, and neural nets enjoy

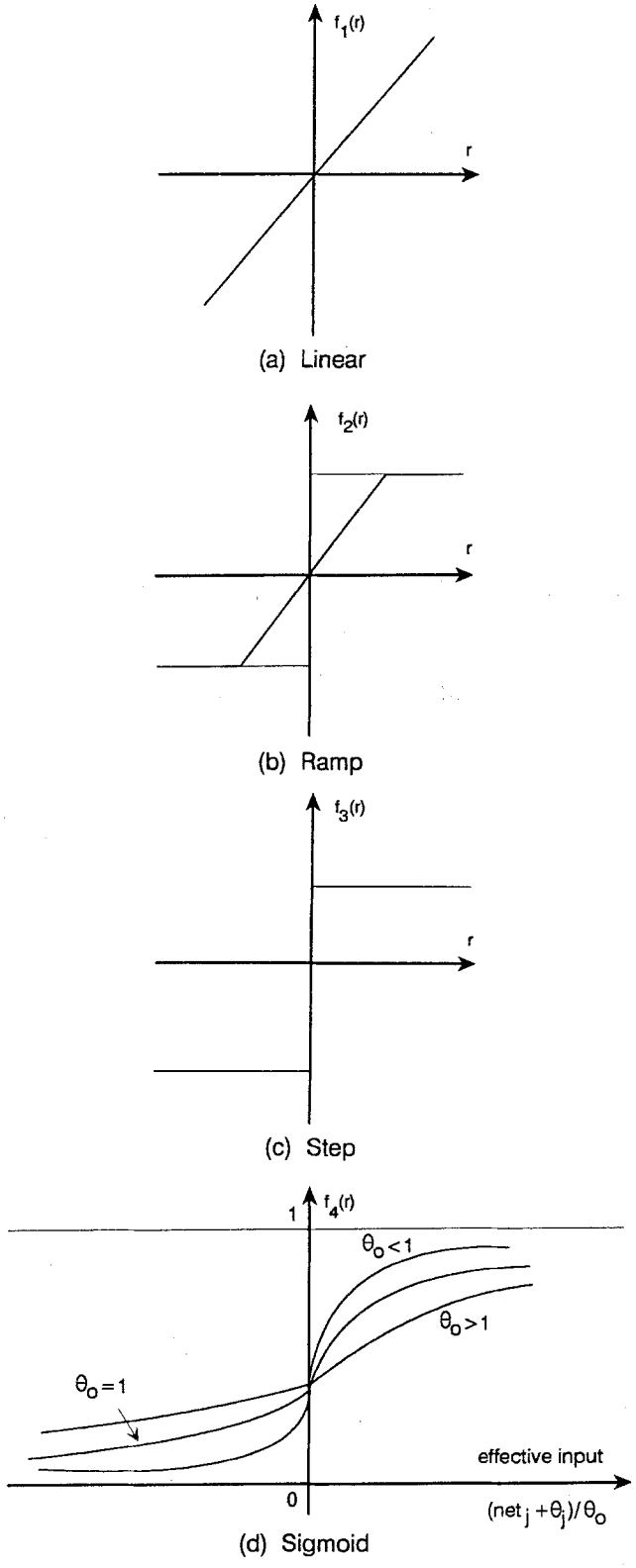


Fig. 3. Activation functions

vigorous funding and developments worldwide. It is this fact that served as motivation for the present study, and other CST applications should also be vigorously investigated in view of expected increases in capabilities.

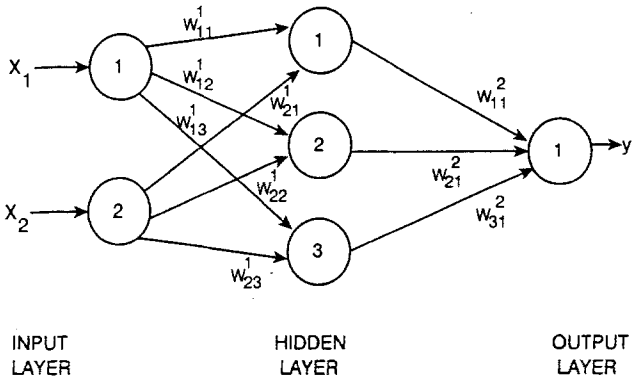


Fig. 4. Six neuron neural net with hidden layer

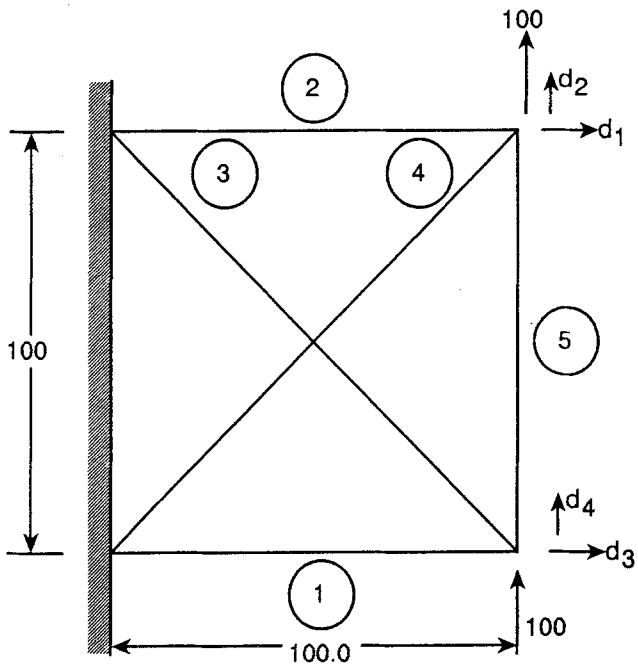


Fig. 5. Five-bar truss

To start out with an application one requires a set of known input and output pairs that must be generated by the "real" process one is planning to simulate. The number of training pairs, and how they span the intended domain of training, is part of what is still an art in ANN requiring experimentation and experience. The same statement is also valid for the architecture of the neural net one intends to use. The examples given later will provide some idea of what is required for a successful application. For the engineering applications presented here, it is perhaps worthwhile to think of neural nets as a peculiar automated hypersurface fitting capability. What one would accept as a representative input-output set to produce a useful surface fit, is most likely a good start to determine the training pairs for the neural nets.

For the present application it is sufficient to discuss the simplest forms of net architectures. A single layer net is called a "flat" net and is of little interest here in its basic form. It has limited capabilities to represent nonlinearities unless these are specifically captured in the input. An example of this is the use of reciprocal variables in problems involving

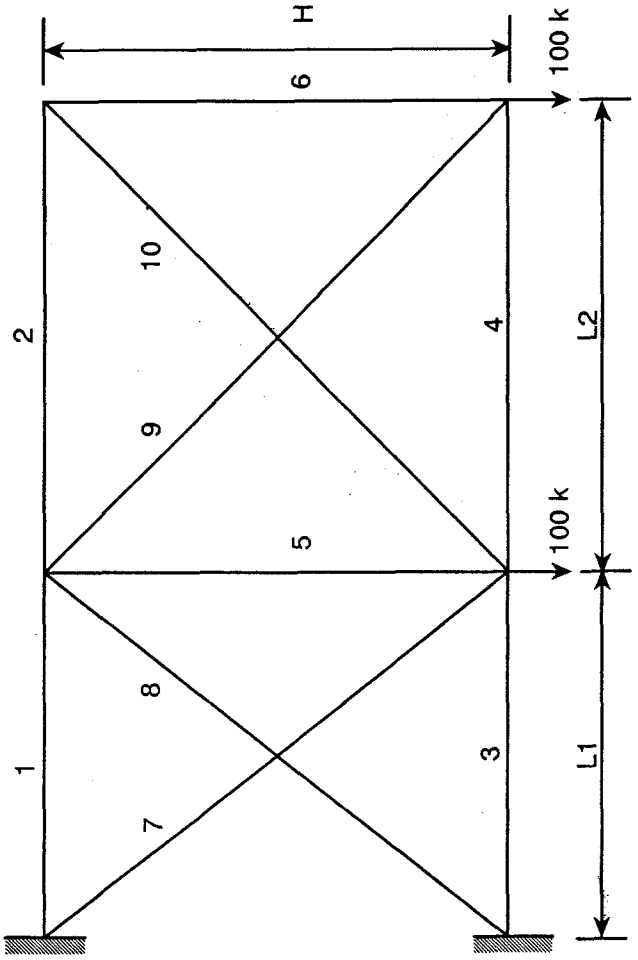


Fig. 6. Ten-bar truss

structural stiffness, a case to be discussed later. Pao (1989) provides a powerful generalization of this concept referred to as "functional links". In general, other nets have an input and output layer with the number of neurons in each of these matching the number of input and output variables, respectively. As an example, Fig. 4 has two nodes in its input layer, three nodes in its single hidden layer, and one node in its output layer. In later discussions this net will be designated a $(2,3,1)$ net signifying the number of nodes in its three layers. A net can provide an n-to-m mapping, which, for the case of Fig. 4, is a 2-to-1 mapping.

The number n and m of nodes in the input and output layer is determined by the number of input and output variables in the training set. It is however, important to determine the necessity of one or more "hidden" layers in the network. A single hidden layer with nodes numbering somewhere between the average and the sum of the input and output nodes is suggested in the literature as a good first start. To add more layers for added flexibility is a temptation which must be resisted in the simple cases addressed in this note. A general suggestion is to try to use as few nodes as possible; as in any optimization problem one should avoid needless increase in the number of optimization variables.

Once an architecture has been selected, the training starts out with a random set of connection weights w_{lk} usually generated automatically by the particular capability used. These connection weights are then adjusted by the error backpropagation learning algorithm to minimize the difference between the training output values and the values produced by "propagating" the associated input through the net. The training is sensitive to the choices of the various net learning parameters. The principal parameters are the "learning rate" which essentially governs the "step size", a concept familiar to the optimization community, and the "momentum coefficient" which forces the search to continue in the same direction so as to aid numerical stability, and furthermore, to go over local minima encountered in the search.

Table 1. Summary of training results with no hidden layer

Number of training sets	Network description	Cycles of training	Error description
50	(5,2) - five areas as inputs, two vertical displacements as outputs	1500	ε=0.087 error not decreasing
50	(10,2) - five areas and five reciprocal areas as inputs - two vertical displacements as outputs	50000	ε=0.03927 error decreasing slowly
50	(15,2) - five areas and ten area products of type A_iA_j $(i \neq j)$ as inputs, two vertical displacements as outputs	50000	ε=0.05235 error decreasing slowly
50	(20,2) - five areas, five reciprocal areas and ten values of type $\sin(A_i/A_{\max})$ used as input - two vertical displacements as outputs	50000	ε=0.0398 error decreasing slowly

(xx, yy) denotes xx input nodes and yy output nodes

During supervised learning these parameters are adjusted periodically based on the changing convergence trend during iterations. In the "ten-bar truss optimum design expert" example discussed later, a publicly available NASA developed capability NETS 2.0 was used. Its user's manual (Baffes 1989) provides a good introduction for someone who would like to experiment with neural nets. NETS 2.0 has a number of other learning parameters and provides good default values for them, including some adaptive features during training iterations. A few possible applications within CST are suggested next, followed by a representative set of the results obtained in preliminary feasibility studies.

3 Neural nets in computational structures technology

The history of the exploitation of computer technology by CST can be viewed as attempted simulations of the brain processes of an expert designer at higher and higher levels of abstraction. Procedural codes, expert systems, and neural nets represent this higher and higher levels of abstraction from number "crunching", "expert judgements" and finally a "feel" for a problem area, respectively. These three levels represent increasing intellectual levels and ability to provide quick expert estimates for solutions with less participation re-

quired of the human user. The final aspiration of researchers in CST is the development of automated expert design capability; neural nets perhaps provide an intriguing approach towards that goal.

Table 2. Optimal design for five-bar truss using trained neural net for analysis

Network description		Design variables X_1	X_2	X_3	X_4	X_5	Objective functions
(5-7-4) 200 training sets, all four output displacements mapped	initial	1.0	1.0	1.0	1.0	1.0	58.28
	final	2.131	2.032	2.679	2.766	1.0	128.626
(5-7-4) 500 trainings sets, all four output displacements mapped	initial	1.0	1.0	1.0	1.0	1.0	58.28
	final	1.952	2.013	2.763	2.760	1.0	127.759
(5-7-2) 100 trainings sets, two vertical displacements mapped	initial	1.0	1.0	1.0	1.0	1.0	58.28
	final	1.535	1.778	2.29	2.265	1.0	107.56
(5-7-2) 100 trainings sets, two vertical displacements scaled as constraints and mapped	initial	1.0	1.0	1.0	1.0	1.0	58.28
	final	1.505	1.584	2.138	2.211	1.0	102.399
	exact solution	1.5	1.5	2.121	2.121	1.0	100.0

(xx-yy-zz) denotes a three layer architecture with xx input layer nodes, yy hidden layer nodes, and zz output layer nodes

On a philosophical level one can view the application of neural nets as an attempt to experiment with simple models of the very complex brain activity of accumulating, generalizing, and retaining expert knowledge. As described earlier, artificial neural nets perform these functions by developing specific "patterns" of their connection weights. These patterns, and not any individual value serves as the storage of the knowledge. It would be naive to make much of this supposed similarity, but as will be shown, the feasibility of the concept is easy to illustrate through a few examples. Much has been learned about the electrochemical activities of brain cells and of the vast neural nets they form. What all that means is poorly understood if at all, and the functioning of the brain remains largely unknown. A somewhat negative view of the class of neural nets employed in the present study would be that they are essentially glorified curve fitting capabilities. It still remains an intriguing proposition that, as opposed to established mathemetical curve fitting procedures, an entirely different concept of a pattern of connection weights produces the same results. Further, this process is somewhat similar to accumulation of human expert feel for a problem solution.

The major advantage of a trained neural net over the original (computational) process is that results can be produced in orders of magnitude less computational effort than the original process. This effort, once the net is trained, is also insensitive to the effort it takes to generate an output by the original process. Consequently benefits can be higher for those problem areas that are computationally very intensive, such as optimization, especially in multidisciplinary settings. There is, of course, a catch, namely that in those cases the generation of sufficient training data is also more expensive. Practical applications can be envisioned where a problem is frequently solved within limited variations of the input parameters. Organizations with specific products for slightly changing applications could develop or evolve trained neural nets based on sets of past solutions. New solutions could then be obtained with negligible efforts. Machine components that are of a certain basic configuration slightly changing from application to application could be good practical examples. A neural net could be trained to capture past design variations and provide the design for new applications.

Table 3. Optimal design for ten-bar truss using trained neural net for analysis

des. var.	network description $(10-6-6-2)^*$ 100 training sets used in a range of ±25% about optimum	$(10-6-6-2)^*$ 400 training sets used in a range of ±25% about optimum	$(10-6-6-2)$ 100 training sets used in a range of 0.01-55.0 in^2 output scaled to reduce range of variation	solution from exact analysis
X_1	30.774	30.967	30.508	30.688
X_2	0.112	0.100	0.100	0.100
X_3	17.40	19.136	26.277	23.952
X_4	11.425	14.279	11.415	15.461
X_5	0.108	0.100	0.100	0.100
X_6	0.487	0.434	0.413	0.552
X_7	5.593	5.593	5.593	8.421
X_8	22.953	20.031	21.434	20.606
X_9	20.886	19.966	22.623	20.554
X_{10}	0.100	0.100	0.100	0.100
obj. func.	4692.49	4666.71	5010.22	5063.81

* Lower bound of design variables used as initial design - was infeasible

In CST, a number of applications can be envisioned that could prove profitable in the long run. One obvious application would be to train neural nets to produce the element stiffness coefficients of frequently used sophisticated finite element models, within some range of their input parameters. Frequently used codes that produce (nonlinear) composite material properties could be simulated by trained neural nets. Constitutive relations could be captured by neural nets trained either with test data or with input-output sets produced by running available codes for a set of load conditions of interest. Life prediction or fracture mechanics are other possible areas to capture trends by neural nets. The w_{lk} weights, as they develop, may contain information concerning hidden functional relationships between the variables for some of these applications providing "feature extraction" capabilities. A version of neural nets designed for "unsupervised learning" has such feature extraction capabilities. Multidisciplinary design optimization provides particularly intriguing possibilities. For example, nets could be trained for each of the participating disciplines, and hard connected

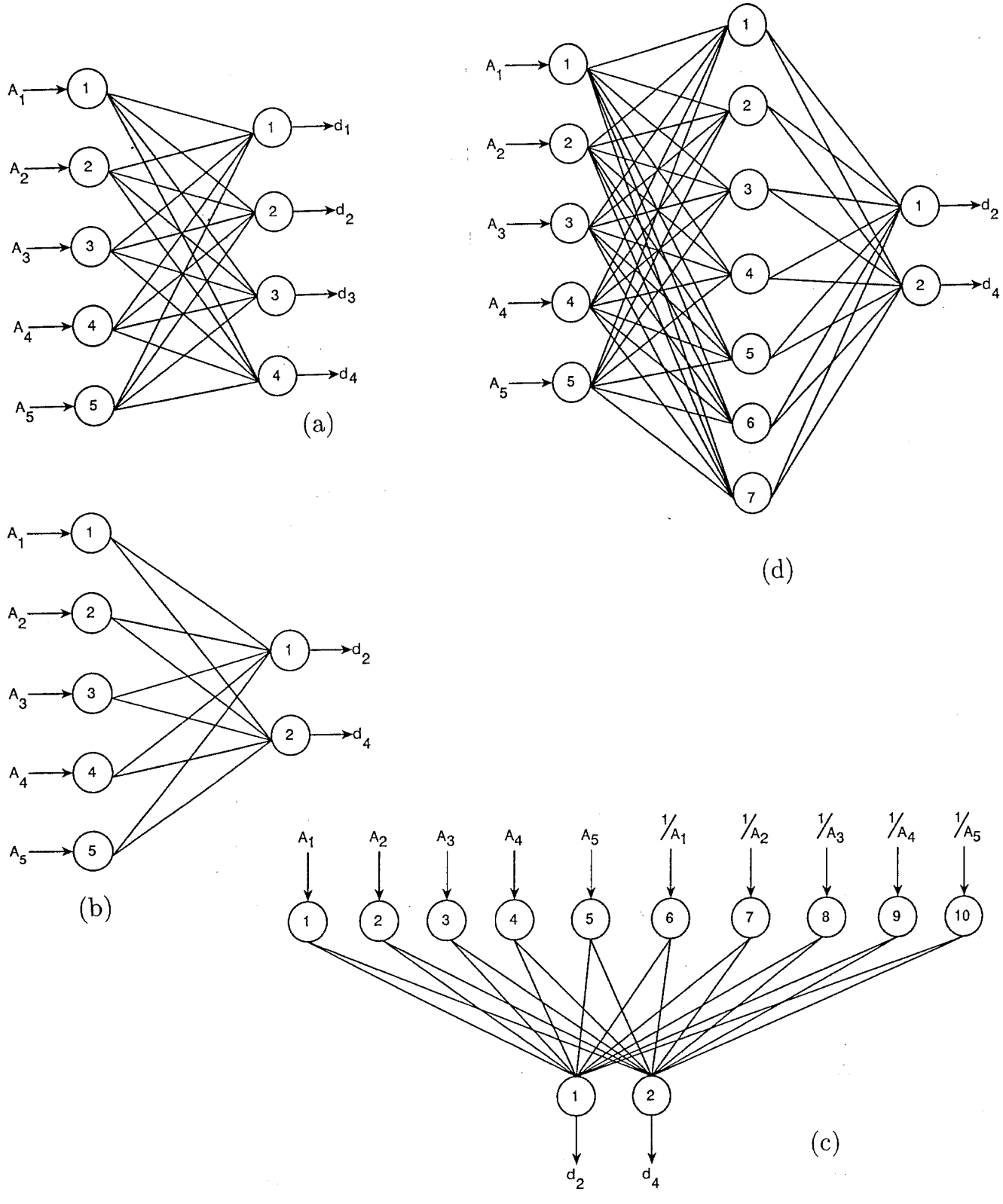


Fig. 7. Neural nets for five-bar truss problem

to represent appropriate coupling or to use an additional net that develops the important coupling functionalities through feature extraction. Initial efforts are now underway in all these areas of research.

The feasibility of two particular applications at two distinct levels of abstraction were studied in some detail and the

results are presented next. The first one involved training a neural net to replace analyses of given structural configurations during optimization iterations. The second exercise was to train a neural net to provide estimates of the actual optimum structures directly, circumventing the usual analysis-optimization iterations.

4 Neural net assisted optimization

This first feasibility study to simulate analysis with the quick response of neural nets was motivated by the approximation concepts in structural optimization. The idea here was to train a neural net to provide computationally inexpensive estimates of analysis output needed for sensitivity evaluations, which in turn is needed by most optimization codes. The numerical experimentation also served to gain initial experience with neural nets. For a detailed description of the effort, see the paper by Hajela and Berke (1990). The neural net capability used in these studies was a research level code.

The familiar five-bar and ten-bar truss "toy" problems, shown in Figs. 5 and 6, respectively, were used for this initial feasibility study. First, various sets of input-output training pairs and network configurations were examined to find the combination that reduced the training effort and produced trained nets which yielded good results as measured by their ability to generalize.

Once an acceptable trained neural net was obtained, it was attached to an optimizer, and all analysis information was obtained from it instead of invoking a conventional analysis capability. Mixing neural net predictions with occasional conventional analyses was not explored, but it is an approach that could possibly exploit the advantages of both.

Table 4. Ten-bar truss optimum designs with H as design condition

H	$A1$	$A2$	$A3$	$A4$	$A5$	$A6$	$A7$	$A8$	$A9$	$A10$	Wt
300	9.53	.1	9.67	4.73	.1	.1	6.34	6.15	6.15	.1	1749.6
305	9.37	.1	9.51	4.65	.1	.1	6.28	6.09	6.09	.1	1733.2
310	9.22	.1	9.36	4.57	.1	.1	6.22	6.03	6.03	.1	1717.4
315	9.07	.1	9.21	4.50	.1	.1	6.16	5.98	5.98	.1	1702.3
320	8.93	.1	9.07	4.43	.1	.1	6.11	5.92	5.92	.1	1687.9
325	8.79	.1	8.92	4.36	.1	.1	6.05	5.87	5.87	.1	1673.5
330	8.66	.1	8.79	4.29	.1	.1	6.00	5.83	5.83	.1	1660.9
335	8.53	.1	8.66	4.23	.1	.1	5.96	5.78	5.78	.1	1648.3
340	8.40	.1	8.54	4.16	.1	.1	5.92	5.73	5.73	.1	1635.9
345	8.28	.1	8.41	4.10	.1	.1	5.87	5.68	5.68	.1	1624.5
350	8.16	.1	8.29	4.05	.1	.1	5.82	5.65	5.65	.1	1613.5
355	8.05	.1	8.17	4.00	.1	.1	5.78	5.60	5.60	.1	1603.2
360	7.93	.1	8.06	3.93	.1	.1	5.74	5.68	5.68	.1	1593.1
365	7.83	.1	7.95	3.88	.1	.1	5.70	5.53	5.53	.1	1583.5
370	7.72	.1	7.84	3.83	.1	.1	5.66	5.50	5.50	.1	1574.3
375	7.61	.1	7.73	3.77	.1	.1	5.63	5.45	5.45	.1	1565.2
380	7.51	.1	7.63	3.73	.1	.1	5.60	5.42	5.42	.1	1557.1
385	7.42	.1	7.53	3.68	.1	.1	5.56	5.39	5.39	.1	1549.1
390	7.32	.1	7.44	3.63	.1	.1	5.53	5.36	5.36	.1	1541.4
395	7.23	.1	7.34	3.58	.1	.1	5.49	5.32	5.32	.1	1534.1
400	7.14	.1	7.25	3.54	.1	.1	5.46	5.30	5.30	.1	1527.1

To create the training sets, optimum designs were obtained first for two reasons. Firstly, optimum designs were required for comparisons with designs obtained using neural net simulation of the analyses. Secondly, analyses were to be performed with random sets of the values of the design variable, in this case the bar areas, within certain preset variations of their optimum values. The optimization involved constraints on the nodal displacements. Consequently, the input-output training pairs consisted of the bar areas as inputs and the nodal displacements as output variables, respectively. How many pairs to use, and within what range of variations, is itself a research question. Because of the nature of the sigmoid function at least the output variables are to be scaled by the user or automatically by the neural net code, to within the most active range of the sigmoid function. Scaling minimum and maximum values to 0.1 and 0.9 is usually suggested.

At this point one has to prescribe the number of iterations for which the network must be trained to obtain desired levels of accuracy. A number from a few hundred to tens of thousands is routinely accepted in neural net applications, even for small nets as in this study. For this level of experimentation one often initiates a run on a PC or a workstation and lets it run to a large number overnight in somewhat of an overkill.

Some of the net configurations examined for the five-bar truss exercise are shown in Fig. 7. A 5-to-4 mapping with a $(5,4)$ net and no hidden layer is shown in Fig. 7a. The four output variables were the four nodal displacements indicated in Fig. 5. Since the active constraints were essentially related to displacements d_2 and d_4, the rest of the nets considered only these two displacements as output. Figure 7b consequently is a $(5,2)$ net with reduced training effort. Figure 7c is a $(10,2)$ net with the reciprocals of the bar areas also included to help the net capture the inverse relation between bar areas and nodal displacements. Table 1 contains data on the results of these initial training efforts with other functional relationships also included to try to capture nonlinearities. These attempts with flat nets were not totally satisfactory in terms of obtained accuracy or number of required training cycles.

Table 5. Training accuracy with (1, 6, 3) net

Training pairs				Training accuracy (RMS= 0.9%)				
Input	Output							
H	$A1$	$A2$	Wt	$A1$	%	$A2$	Wt	%
300	9.53	.1	1749.6	9.543	.14	.101	1744.4	.30
310	9.22	.1	1717.4	9.240	.23	.101	1721.0	.21
320	8.93	.1	1687.9	8.955	.28	.100	1693.4	.33
330	8.66	.1	1660.9	8.668	.11	.100	1665.6	.28
340	8.40	.1	1635.9	8.401	.01	.100	1640.3	.27
350	8.16	.1	1613.5	8.159	.012	.100	1618.1	.29
360	7.93	.1	1593.1	7.930	.00	.100	1597.2	.26
370	7.72	.1	1574.3	7.711	.12	.100	1577.2	.19
380	7.51	.1	1557.1	7.507	.07	.100	1558.2	.07
390	7.32	.1	1541.4	7.326	.08	.100	1542.1	.05
400	7.16	.1	1527.1	7.187	.38	.100	1529.3	.14

Including a hidden layer, as shown in Fig. 7d, produced acceptable results. Table 2 presents the results of optimization using various net and training set combinations. Using the $(5,7,2)$ net and scaled variables, an optimum design was obtained within 2.4% of the exact optimum design proving the feasibility of the basic concept of neural net assisted optimization. Table 3 presents the results of similar experimentation for the ten-bar truss supporting the same conclusion.

Hajela and Berke (1990) also present similar results for a higher dimensionality wing box problem.

5 Neural net as experts for direct optimum design estimates

The next set of experiments were conducted to explore the idea of training a neural net to estimate optimum designs directly for given design conditions and bypass all the analyses and optimization iterations of the conventional approach. It is conceivable in practice that successful similar designs could be collected within some domain of design conditions, input-output pairs defined, and then a neural net trained to serve as "intelligent corporate memory" that can provide a new design for different design requirements.

Table 6. Test set of neural net estimates

Input	Optimum			N-N estimates				
H	$A1$	$A2$	Wt	$A1$	%	$A2$	Wt	%
305	9.37	.1	1733.2	9.364	.11	.101	1733.5	.00
315	9.07	.1	1702.3	9.097	.30	.100	1707.1	.28
325	8.79	.1	1673.5	8.801	.12	.100	1679.0	.33
335	8.53	.1	1648.3	8.529	.01	.100	1652.8	.27
345	8.28	.1	1624.5	8.283	.04	.100	1629.5	.31
355	8.05	.1	1603.2	8.043	.09	.100	1607.5	.27
365	7.83	.1	1583.5	7.818	.15	.100	1587.0	.22
375	7.61	.1	1565.2	7.604	.08	.100	1567.4	.20
385	7.42	.1	1549.1	7.361	.79	.100	1550.2	.07
395	7.23	.1	1534.1	7.252	.30	.100	1535.4	.08

Now let us suppose that we work in a company that markets equipment that is mounted in all cases on ten-bar trusses as shown in Fig. 6. These trusses have to carry the equipment weight (2×100 K) at the two lower free nodes while these support points cannot deflect more than 2 in. The dimensions $L1$, $L2$ and H of the trusses can vary between 300 and 400 inches, depending on the particular installation. The engineer who was designing the trusses for the past 30 years and could simply tell the optimum bar areas for any combination of those dimensions has just retired. Can we create an accurate simulation of this departed expert? Yes we can, and rather simply!

To experiment with various training sets, optimum designs were generated by conventional methods for varying first only H in 5 inch increments. The results are given in Table 4. There are three kinds of output numbers in the set. These are the bar areas that change, areas that are at the preselected minimum value of .1 for all designs, and the weight, which is of a different order of magnitude. A representative A_1, A_2, and the optimum weight WT were considered in the first numerical experiments. The neural net code NETS 2.0 (Baffes 1989) was used for all of the direct optimum ten-bar truss design exercises.

A number of small net configurations were tried for these 1-to-3 mappings. Table 5 shows the results of training with a $(1, 6, 3)$ net, a probable overkill with too many nodes in the hidden layer. Table 6 gives the results of design estimates of the trained net for the remaining check cases of Table 4 that were not included in the training set. The training was performed to 1% RMS accuracy within 200 hundred iterations. As can be seen, both the training accuracy and the estimates for the new cases is around a third of one percent

Table 7. Ten-bar truss optimum designs for training with $L1$, $L2$, and H as design conditions

Input			Output										
$L1$	$L2$	H	$A1$	$A2$	$A3$	$A4$	$A5$	$A6$	$A7$	$A8$	$A9$	$A10$	Wt
310	350	380	6.89	.1	7.00	3.62	.1	.1	5.24	5.08	5.35	.1	1356.7
345	326	360	7.40	.1	7.50	3.56	.1	.1	5.62	5.45	5.31	.1	1456.4
371	329	310	8.95	.1	9.10	4.17	.1	.1	6.33	6.14	5.74	.1	1684.5
360	300	340	7.69	.1	7.83	3.47	.1	.1	5.91	5.73	5.25	.1	1492.6
315	340	340	7.64	.1	7.76	3.93	.1	.1	5.53	5.36	5.56	.1	1407.6
380	355	390	7.47	.1	7.60	3.58	.1	.1	5.67	5.50	5.32	.1	1605.7
322	319	400	6.35	.1	6.46	3.13	.1	.1	5.21	5.05	5.03	.1	1314.2
400	300	400	9.25	.1	9.41	3.93	.1	.1	6.77	6.56	5.56	.1	1780.9
300	400	300	9.27	.1	9.39	5.25	.1	.1	5.73	5.57	6.57	.1	1593.8
311	350	315	8.33	.1	8.45	4.36	.1	.1	5.70	5.53	5.88	.1	1464.6

for the individual values and can be considered quite satisfactory. It is also of some interest to note that the net had to evolve its w_{lk} connection weights and other internal parameters provided by NETS 2.0 in such a manner that it could also reproduce a constant .1 value for any input while also producing accurate values of variables or different orders of magnitude for the same inputs.

Table 8. Ten-bar truss optimum designs for checking estimates of the trained neural net

Input			Optimum solutions										
$L1$	$L2$	H	$A1$	$A2$	$A3$	$A4$	$A5$	$A6$	$A7$	$A8$	$A9$	$A10$	Wt
342	351	383	7.18	.1	7.29	3.60	.1	.1	5.44	5.27	5.34	.1	1466
360	360	360	7.93	.1	8.06	3.93	.1	.1	5.74	5.56	5.56	.1	1593
320	350	360	7.38	.1	7.50	3.82	.1	.1	5.43	5.26	5.49	.1	1417
340	370	340	8.29	.1	8.41	4.28	.1	.1	5.74	5.56	5.82	.1	1578
310	350	380	6.89	.1	6.99	3.62	.1	.1	5.24	5.08	5.35	.1	1356
345	326	360	7.39	.1	7.51	3.56	.1	.1	5.62	5.45	5.30	.1	1456
371	329	310	8.95	.1	9.16	4.17	.1	.1	6.33	6.14	5.74	.1	1684

After the above limited exercise, the 3-to-11 mappings were finally performed with a $(3, 14, 11)$ net between $L1$, $L2$, H and the ten bar areas $A_1, \ldots, A_{10}$ and the optimum weight Wt, creating the "ten bar optimum design expert" to replace our retired expert designer. A rather limited training set was created as the optimum designs for only ten random sets of $L1$, $L2$ and H. It is interesting to note that only ten training pairs were used, and that they proved adequate. Of course, in this problem only three variables are varied to cover a domain. The ten optimum designs providing the training pairs are given in Table 7. Optimum designs were then obtained for another seven random sets of $L1$, $L2$ and H, as checks on the estimates to be obtained from the trained network. Table 8 shows these seven optimum designs.

During experimentation with various options during training, it was found that it is beneficial to code the 0.1 minimum sizes as 0.5. The active midpoint of the sigmoid activation function is the explanation. This value will be shown to represent net accuracy in better detail. NETS 2.0 worked very well, and 1% RMS accuracy was obtained with 200 iterations in around 30 secs on a SUN 386i, and using only the default values for the learning parameters. Exercises were also conducted to overtrain the net by letting the training run for 5000 iterations to an RMS accuracy of .0062% was obtained. Overtraining is to be avoided because the neural net at that point becomes a memory with lessened ability to generalize. The overtrained net actually reproduced the

training results exactly, but it did a little worse, if anything, against the seven check conditions than the net trained to only 1% RMS accuracy.

Table 9 shows comparisons and the percentage error of net estimates for the seven check cases of Table 8 for the net trained to 1% accuracy. As can be seen, the results are quite satisfactory and certainly would be good enough information to produce the ten-bar trusses. The net produced its estimates by computing a few sums of products in practically no computer time. The mental activities our retired expert designer employed to come up with his optimum designs have been replaced by a trained $(3, 14, 11)$ neural net of similar capability for this limited task.

Table 9. Trained neural net estimates of ten-bar truss optimum designs

$A1$	7.138	8.000	7.365	8.398	6.908	7.310	8.916
%	0.580	0.780	0.200	1.300	0.260	1.080	0.380
$A2$	0.503	0.505	0.502	0.507	0.502	0.501	0.504
$A3$	7.245	8.160	7.485	8.558	7.013	7.448	9.089
%	0.620	1.240	0.200	1.760	0.330	0.820	0.770
$A4$	3.554	3.957	3.777	4.354	3.551	3.533	4.177
%	1.230	0.690	1.130	1.730	1.900	0.760	0.170
$A5$	0.499	0.499	0.500	0.499	0.500	0.500	0.500
$A6$	0.501	0.501	0.500	0.501	0.500	0.500	0.500
$A7$	5.456	5.783	5.461	5.781	5.309	5.586	6.359
%	0.290	0.750	0.570	0.710	1.320	0.610	0.460
$A8$	5.293	5.594	5.301	5.598	5.164	5.416	6.157
%	0.470	0.610	00.780	0.680	1.650	0.620	0.280
$A9$	5.326	5.574	5.477	5.863	5.336	5.311	5.740
%	0.260	0.250	0.240	0.740	0.260	0.210	0.000
$A10$	0.499	0.500	0.499	0.499	0.497	0.503	0.500
Wt	1466.000	1598.000	1417.000	1585.000	1362.000	1451.000	1693.000
%	0.000	0.310	0.000	0.440	0.440	0.340	0.530

6 Closing remarks

It has been shown that artificial neural nets have intriguing applications in computational structures technology. What has been presented are some of the results of what most likely are the first systematic explorations of some of the possibilities. There are now efforts underway to explore multidisciplinary design applications, to "package" composite material property generation codes as quick response neural net simulations, and to develop structure/component life prediction (see Troudet and Merrill 1990) capabilities. Applications to coupled multidisciplinary design optimization problems are particularly intriguing and investigations in this area have been initiated.

Received Jun. 26, 1990
Revised manuscript received Feb. 3, 1992

References

Baffes, P.T. 1989: NETS 2.0 users guide. *LSC-23366*, NASA Lyndon B. Johnson Space Center

Hajela, P.; Berke, L. 1990: Neurobiological computational models in structural analysis and design. Paper given at the 31st AIAA SDM Conf., Long Beach, California

McCauley, A.D. 1988: *Optical neural network for engineering design.* NAECON

Pao, Y.-H. 1989: *Adaptive pattern recognition and neural networks.* Addison-Wesley Publishing Co.

Rehak, D.R.; Thewalt, C.R.; Doo, L.L.; Nelson, J.K. (eds.) 1989: *Neural network approaches in structural mechanics computation. Comp. Utilization Struct. Enging.* ASCE Proc. Structures Cong.

Rumelhart, D.E.; McClelland, J.L. 1988: *Parallel distributed processing, volumes 1 & 2.* Cambridge, Massachusetts: The MIT Press

Troudet, T.; Merrill, W. 1990: A real time neural net estimator of fatigue life. *NASA TM 103117*