Int. J. Engng Ed. Vol. 12, No. 3, p. 199–209, 1996
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Education for Automation in Structural Design*

S. D. JOST
Department of Computer Science and Information System, DePaul University of Chicago, 60604, USA
Y. L. MO
Department of Civil Engineering, National Cheng Kung University, Tainan 701, Taiwan, ROC
H. C. WU
Department of Management of Information Systems, Kung Shan Institute of Technology and Commerce, Tainan 710, Taiwan, ROC

This is a summary of the course 'automation in structural design'. The course can be taught at the upper-undergraduate or graduate level. Basically automated systems in structural design can be evolutionarily classified into four types: (i) traditional computer-aided design, (ii) database management systems, (iii) expert systems and (iv) neural networks. This paper is a survey of these four types of systems and gives both some theoretical background and actual software applications. These applications are SAP90, LMS, CCES and NNISE. Because structural design is likely to be more software dependent in the future, familiarity with applications such as these is becoming increasingly important.

INTRODUCTION

THIS paper presents an outline for a course in structural engineering titled 'automation in structural design'. It is hoped that this outline may provide some information about education for automation in structural design.

During the last 30 years, computers have been developing at a rate faster than any previous industrial invention, and great strides have been made in using computer tools to aid in structural design. Currently, a new generation of computers emerges every 4–7 years. With each new generation, computing power typically increases by several times (if not by an order of magnitude) with a corresponding decrease in physical size and price. Due to the rapid advances in computer technology, an evolution of the traditional discipline of structural design is underway. The most important feature of this evolution is that research and practice in structural design are becoming more software dependent and more software intensive.

The purpose of this paper is to sketch some of the directions that this automated structural design technology has taken and to provide some examples of actual automated systems that have been developed. Even many of the common acronyms used to describe these technologies, such as CAD (computer-aided design), GUI (graphical user interface) and AI (artificial intelligence), were unknown 40 years ago. In this paper the evolution of automation in structural design is introduced; the types of automation in structural design are described; and the mathematical background is explained. Finally, automation applications to structural design are demonstrated. These applications include traditional CAD, database management systems, expert systems and neural networks.

EVOLUTION OF AUTOMATION IN STRUCTURAL DESIGN

Automation is technology concerned with the application of mechanical, electronic and computer-based systems to operate and control production [1]. When this technology is employed for structural design, it is called automation in structural design. Therefore, the scope of this paper is limited primarily to automated systems used for structural design. Some examples of software for structural designs using these types of systems are SAP90 for general structures, ETABS for buildings, LMS for nuclear power plants, CCES for corrosion consultation, and NNISE for beams and plates.

Four basic types of automated design are considered in this paper: (i) traditional computer-aided design; (ii) database management; (iii) expert systems; and (iv) neural networks.

Traditional computer-aided design systems are usually written in procedural languages such as FORTRAN or C. These languages are often called third-generation languages [2]. Such a system might use finite element methods [3] to perform stress analysis first, and then to check for allowable

* Accepted 15 November 1995.

stresses in the design. Currently, most computer programs for structural design still belong to this type.

As modern large buildings designed by large computers become more and more complex, database management systems are increasingly being used to keep track of the vast amounts of information needed for their design. These systems have developed into computer languages in their own right and are called fourth-generation languages [2] because they are non-procedural. A well-designed database management system offers the following advantages [4]:

- Information which has been entered into the database is immediately available to all members of the design team, thus promoting consistency and eliminating potential design data communication pitfalls.
- Redundant information is eliminated as much as possible. Information is stored in specific, logically defined locations and is always kept current by the engineer responsible.
- Work can be partitioned to manageable areas of responsibility and assigned to more than one engineer.
- Information can be referenced in a logical manner using meaningful identifiers.
- Overall data handling is decreased, which helps reduce man-hours, calendar time and the potential for error.

Another modern approach to automated design is the use of expert systems, which is a branch of artificial intelligence. Some of the expectations of expert systems are the following [5,6]:

- They should be able to solve problems currently solved by experts, at least in certain restricted situations.
- They are developed by knowledgeable engineers with occasional interaction with experts.
- They can be quickly prototyped and expanded; new capabilities can also be added as the system is extended and modified.

Unfortunately, perhaps because of the catchy name, the expectations of expert system technology exceed reality in many cases. In addition, some expert systems that work well in simple, restricted situations fail to perform well when scaled up to realistic problems. However, with modern hardware and software advances, expert systems are making significant progress towards solving real-world structural design problems.

Neural networks are another form of automation which is beginning to find applications in structural engineering. An increasing number of researchers are investigating neural networks, although they have also suffered from overly optimistic expectations [7]. Unlike expert systems, which are built on rules which an expert in the field would use, a neural network is trained by trial and error based on observed information. These systems exhibit a learning and memory capability similar to biological brains, although at a much simpler level. While neural networks have certain advantages of biological brains, they also have a certain disadvantage: the mechanism by which stored information is used to do computations is often very complex. The differences in design and philosophy between expert systems and neural networks are discussed in more detail later. Figure 1 indicates the evolution of automation in structural design according to the four types discussed above.

TRADITIONAL COMPUTER-AIDED DESIGN

Mathematical background

Over the past three decades finite element analysis of structures has been extensively used in the areas of structural engineering and structural

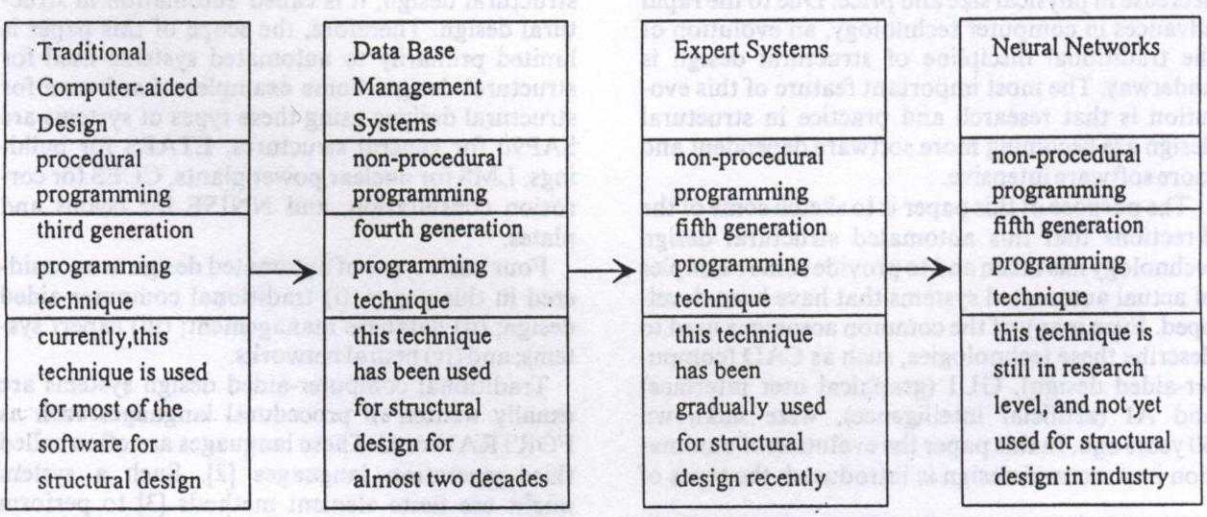


Fig. 1. Evolution of automated structural design.

design. This method is referred to traditional CAD. Its mathematical background is briefly summarized below.

The process of approximating the behaviour of a continuum by finite elements which behave in a manner similar to the real, discrete elements can be introduced through the medium of particular physical applications or as a general mathematical concept.

The approach is to divide the region Ω into a number of non-overlapping subdomains or elements Ω^e and then to construct the approximation function in a piecewise manner over each subdomain. The trial functions used in the approximation process can then also be defined in a piecewise manner by using different expressions in the various subdomains Ω^e from which the total domain is developed. In such a case, the definite integrals occurring in the approximating equations can be obtained simply by summing the contributions from each subdomain or element as

$$\int_\Omega W_1 R_\Omega \mathrm{d}\Omega = \sum_{e=1}^{E} \int_{\Omega^e} W_1 R_\Omega \mathrm{d}\Omega \tag{1}$$

$$\int_\Gamma \overline{W}_1 R_\Gamma \mathrm{d}\Gamma = \sum_{e=1}^{E} \int_{\Gamma^e} \overline{W}_1 R_\Gamma \mathrm{d}\Gamma \tag{2}$$

provided that $\sum_{e=1}^{E} \Omega^e = \Omega, \sum_{e=1}^{E} \Gamma^e = \Gamma$. Here W_1 and $\overline{W}_1$ denote weighting functions, R_Ω denotes the residual (error) in the domain, E denotes the total number of subdivisions of the region and Γ^e denotes that portion of the boundary of Ω^e which lies on Γ. Summations involving Γ^e are therefore taken only over those elements Ω^e which lie immediately adjacent to the boundary.

If the subdomains are of a relatively simple shape and if the definition of the trial functions over these subdomains can be made in a repeatable manner, it is possible to deal in this fashion with assembled regions of complex shapes quite readily.

The piecewise definition of the trial or shape functions means that discontinuities in the approximating function or in its derivatives will occur. Some degree of such discontinuity is permissible, and this will govern the choice of formulation used.

If the trial functions are to be defined in a piecewise manner, it is advantageous to assign to them a narrow 'base' and make their value zero everywhere except in the element in question and in the subdomain immediately adjacent to this element.

In many aspects of engineering the solution of stress and strain distributions in elastic continua is required. Special cases of such problems range from two-dimensional plane stress or strain distributions, axisymmetric solids, plate bending, and shells, to fully three-dimensional solids. In all cases the number of interconnections between any finite element isolated by some imaginary boundaries and the neighboring elements is finite. It is therefore difficult to see at first glance how such problems may be discretized. Zienkiewicz and Taylor [8] suggest several strategies to overcome this difficulty.

1. The continuum is separated by imaginary lines or surfaces into a number of finite elements.
2. The elements are assumed to be interconnected at a discrete number of nodal points situated on their boundaries. The displacements of these nodal points will be the basic unknown parameters of the problem, just as in the simple discrete structural analysis.
3. A set of functions is chosen to define uniquely the state of displacement within each finite element in terms of its nodal displacements.
4. The displacement functions now define uniquely the state of strain within an element in terms of the nodal displacements. These strains, together with any initial strains and the constitutive properties of the material, will define the state of stress throughout the element and, hence, also on its boundaries.
5. A system of forces concentrated at the nodes and equilibrating the boundary stresses and any distributed loads is determined, resulting in a stiffness relationship.

Application

SAP90 [9] is a finite element program for the linear static and dynamic analyses of structural systems. The structural systems that can be analyzed on SAP90 may be modeled by one or a combination of the following element types: three-dimensional frame (beam) element, three-dimensional shell element, two-dimensional solid element, three-dimensional solid element.

The two-dimensional frame, truss, membrane, plate bending, axisymmetric and plane strain elements are all available as special cases of the four elements named above. A boundary element in the form of translational or rotational spring supports is also available in the program. The type of loads allowed by the program include gravity, thermal, prestress, distributed or nodal forces, and prescribed displacements.

The program can perform static, steady-state, eigenvalue and dynamic analyses. The static and dynamic analyses may be activated together in the same run, and load combinations may include results from both analyses. The dynamic analyses may include response spectrum or time history analyses. In the time history analyses, loading can be nodal load or base acceleration and the solution is obtained using standard modal superposition or the Ritz vectors. Similar to SAP90, ETABS [3] has been developed especially for building design. Figure 2 shows a high-rise building modeled with ETABS for structural design.

DATABASE MANAGEMENT SYSTEMS

Overview of relational database systems

Many modern database management systems store and manipulate data in the form of a

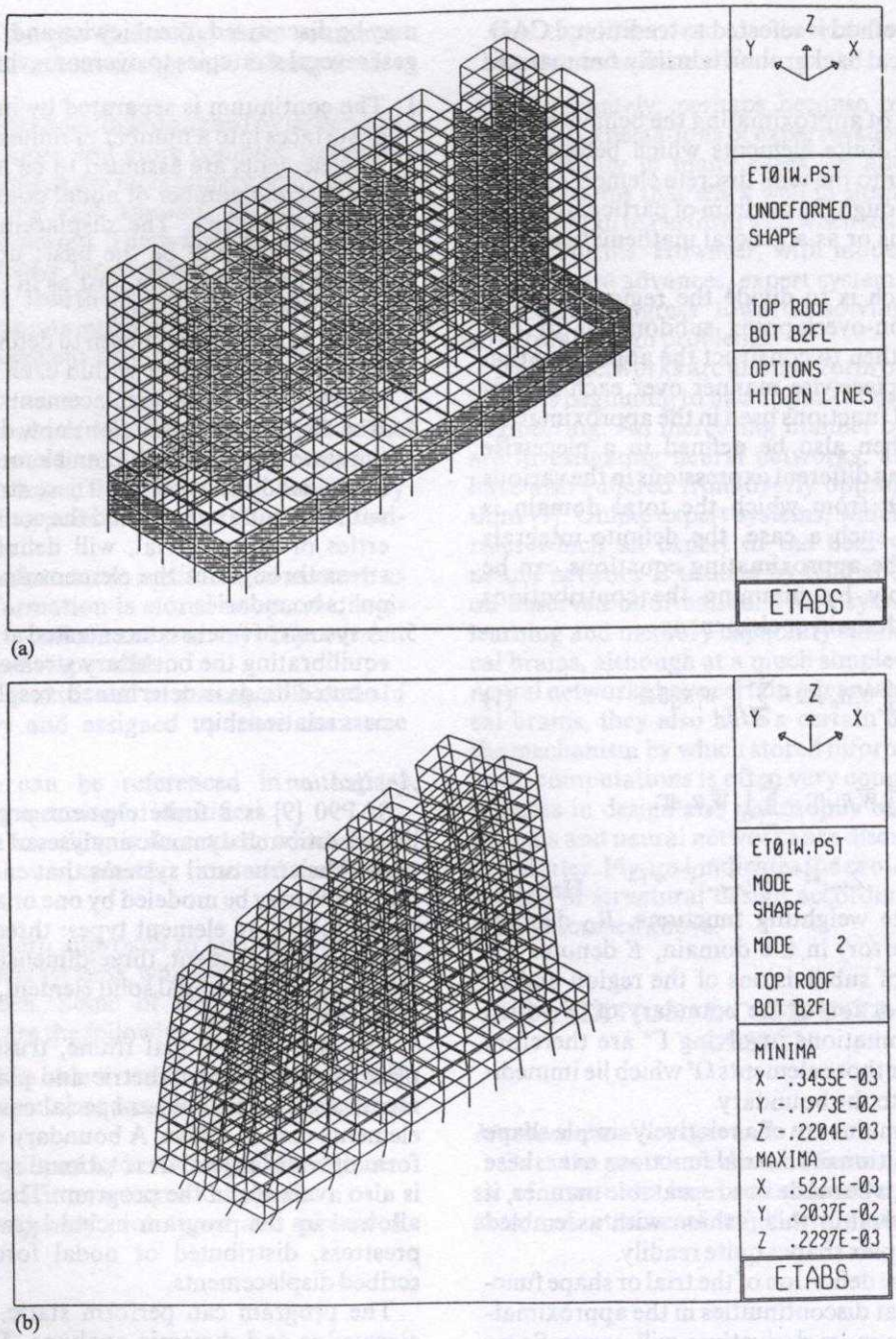


Fig. 2. A high-rise building modeled with ETABS: (a) undeformed shape; (b) twisting mode shape.

rectangular array called a relation or table. Such systems are called relational database management systems. Figure 3 gives two examples of tables. The rows of the table are called tuples, each of which represents information of interest about a single item or object stored in the database. The columns are sometimes called attributes, which are properties of the items in the database.

For example, the table *workers* in Fig. 3 contains information about the workers on a construction site who are not supervisors. The columns of *workers* are id, name, phone number, day off and floor of current assignment. The table *supervisors* contains the id, name and phone number of each supervisor. Note that a table is an abstract representation of the data that is independent of the physical representation of the data on a computer hard drive.

Operations on relations

A database user does not usually wish to look at all of the tuples in a table, but only those meeting certain conditions. A query is a search which locates all of the tuples that satisfy a given logical condition. In order to express queries about data in tables, several operations are useful. The selection operation selects some of the tuples from a table to create a new, smaller table. In order to foster compatibility between different database systems, SQL (standard

workers

id	*name*	*phone*	*dayoff*	*floor*	*superid*
48532	Roberts	554-4339	WED	4	44157
21853	Smith	554-2007	FRI	5	72313
33294	Patel	554-1187	WED	4	44157
42108	Zhou	554-2991	THU	4	58229
28111	Lee	554-5432	FRI	3	72313
20017	Negra	554-5498	WED	3	44157

supervisors

sid	*sname*	*sphone*
44157	Harvey	554-4339
58229	Chu	554-7018
72313	Sanchez	554-5342

Fig. 3. Relations of an example table.

query language) is used to express operations on databases [10]. This section gives a brief glimpse of how SQL can be used to express simple queries.

Tuples are often selected to satisfy some logical condition. This is called a selection operation. For example, the table containing the tuples of all of the workers who have WED as their day off on floor 4, is Table 1 in Fig. 4. This query is express in SQL as

```
select * from table workers where
  dayoff = 'WED' and floor = 4;
```

(The asterisk means keep all columns.)

The projection operator keeps some of the columns from a table to create a smaller table. In the preceding table, suppose we only need the two columns name and superid. Executing the SQL query

```
select name, superid from table workers;
```

produces Table 2 in Fig. 4.

The join operator combines the information from two tables to create a large relation containing all the columns from both tables. Tuples from the two tables are joined if they have the same value for the join variable. If a value of the join variable for a tuple in one table does not exist in any of the tuples in the other table, the tuple is not included in the join. Suppose we wish to know the names and phone numbers of supervisors in addition to the information in the table *workers*. We need to join the tables *workers* and *supervisors* using the join variables *superid* and *sid* to produce Table 3 in Fig. 4. This table is produced by the SQL query

```
select * from table workers,
  supervisors where superid = sid;
```

Finally, suppose we need to know a worker's name, and the name and phone number of that worker's supervisor. We can combine queries of type selection, projection and join to extract the required information:

```
select name, sname, sphone from
  table workers, supervisors
  where superid = sid and dayoff = 'WED'
    and floor = 4;
```

This produces Table 4 in Fig. 4.

In addition to selections, projections and joins,

Table 1

id	*name*	*phone*	*dayoff*	*floor*	*superid*
48532	Roberts	554-4339	WED	4	44157
33294	Patel	554-1187	WED	4	44157

Table 2

name	*superid*
Roberts	44157
Smith	72313
Patel	44157
Zhou	58229
Lee	72313
Negra	44157

Table 3

id	*name*	*phone*	*dayoff*	*floor*	*superid*	*sid*	*sname*	*sphone*
48532	Roberts	554-4339	WED	4	44157	44157	Harvey	554-4339
21853	Smith	554-2007	FRI	5	72313	72313	Sanchez	554-5342
33294	Patel	554-1187	WED	4	44157	44157	Harvey	554-4339
42108	Zhou	554-2991	THU	4	58229	58229	Chu	554-7018
28111	Lee	554-5432	FRI	3	72313	72313	Sanchez	554-5342
20017	Negra	554-5498	WED	3	44157	44157	Harvey	554-4339

Table 4

name	*sname*	sphone
Roberts	Harvey	554-4339
Patel	Harvey	554-4339

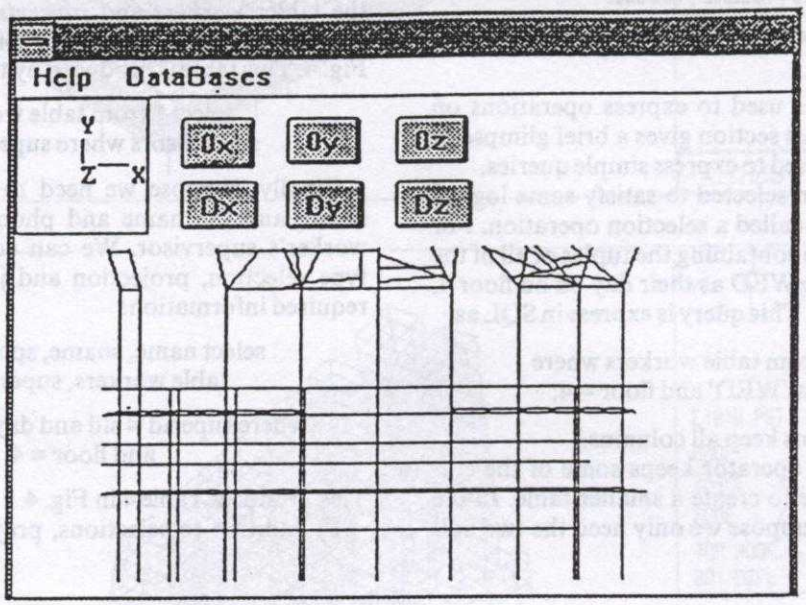
Fig. 4. Tables produced from queries.

SQL supplies an insert command for initially populating the tables with tuples or for adding new rows to existing tables [10]. It also supplies a delete command to remove rows from tables.

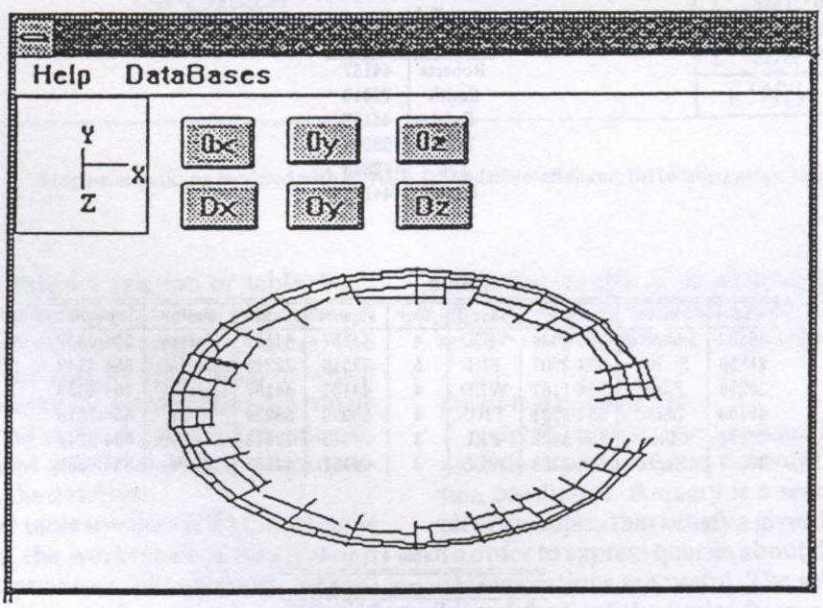
Application

LMS [11] stands for Load Monitoring System. It is used to aid engineers in monitoring the structural integrity of nuclear power plant structures, because of the constant changes in design loads due to plant modifications, unforeseen regulatory changes and the concurrent design/construction cycles necessary in such plants. Hence, it can be classified as a database management system for structural design.

The GUI [12] design of the system is object-oriented, i.e. can be graphically represented as a Booch diagram [2], and is implemented in C. The database management technology used includes a relational data model written in FORTRAN with a FORTRAN data manipulation interface. Its major capabilities include optional run-time binding, a query report model, a database editor, reorganizational utilities, performance timing fea-



(a)



(b)

Fig. 5. An industrial building: (a) elevation; (b) floor plan.

tures, sequential, direct, and keyed access levels, multiple key support, security tracking to record level, and user-defined access passwords for database access control. The data structure of the LMS includes structure geometry, structure loadings, static analysis, stress check, connection check and design investigation. It is also implemented in FORTRAN.

The combination of database management, load monitoring and GUI forms a useful tool for monitoring loads in nuclear power plants. This tool allows the user to temporarily change design data, to use graphic plots of the structure for reference and data selection, to perform quick online load monitoring limited to areas affected by changes, to review results of the monitoring, and to save changes permanently in the database. Figure 5 shows the application of the LMS to an industrial building.

EXPERT SYSTEMS

Theoretical background

Although expert systems often represent knowledge in a manner similar to a DMBS, they have increased capabilities for reasoning about the knowledge in the form of logical manipulations. An expert system stores initial data, and rules for reasoning, which in our example take the form of implications and assertions which are to be derived from the initial data.

An example is the following: suppose that the assertions are

A = 'The load on reinforced concrete beam no. 1 is 20,000 lb.'
B = 'The cross-section of reinforced concrete beams no. 1 is 10 in. by 15 in.'
C = 'The torsion reinforcement area of reinforced concrete beam no. 1 is 2.0 inch square.'
X = 'The maximum moment in reinforced concrete no. 1 is 50,000 ft.–lb.'
Y = 'The resisting moment is 60,000 ft.-lb.'
Z = 'Reinforced concrete beam no. 1 will not fail.'

Suppose in addition that the rules are

(1) A implies X
(2) (B and C) implies Y
(3) (X and Y) implies Z

When the size of the initial data and/or number of rules is large, computing output assertions may be time consuming. Deductions using forward chaining begin with the input data and computer possible output assertions. For example, from the above input data and rules, the output assertions X, Y and Z can be derived. However, computing all possible assertions can be time consuming. When desired output assertions are known in advance, backward chaining can be used to work backwards to verify using the rules. For example, if the desired output assertion Z is to be true, then X and Y must be true by rule (3), B and C must be true by rule (2), and A must be true by rule (1). Since A, B and C are all true by the initial data, Z can be derived.

The number of assertions may be reduced by introducing variables into them. For example, assertion A may be expressed as 'The load on reinforced concrete beams S is T' so that it applies to all reinforced concrete beams in the building, not merely to beam no. 1.

Complications arise when the initial data and rules change over time, because in that case the conclusions may depend on the order in which the rules are applied. Problems can also result when rules give rise to contradictory conclusions. Durkin [13] suggests several strategies to help decide the order in which to apply rules:

1. Execute the rules in a first come, first served order.
2. Place priority on more important rules.
3. Place priority on more specific over more general rules.
4. Place priority on rules most recently added to working memory.
5. Do not apply a rule which has already been applied. This avoids cyclical reasoning.
6. If applying a rule contradicts an already obtained conclusion, apply if anyway, but store its conclusions in a separate working memory to maintain separate lines of reasoning.

Application

Flue gas desulfurization (FGD) [14] has become a very common term within the utility industry and among those industries associated with air pollution control. As power generation facilities continue to expand their use of coal as a primary fuel, the need for effective and reliable FGD systems becomes important to maintain environmentally acceptable plants. Therefore, a corrosion consultant expert system (CCES) [15] was developed. CCES is an expert system, and is also developed with GURU [16]. CCES consists of a rule set and two databases. The rule set includes the basic rules to be used in the inference process. The first database maintains application and material property for various coatings and linings. The second database maintains historical performance data. When an FGD component and its service environment are specified, CCES can be used to obtain a list of recommended coatings tabulated in order of their ranking scores (see Fig. 6).

In the system a user interface module, written in GERU's procedural language, is used to direct the overall consultation process and to provide the link between the system and the user. The inference process is driven by GURU's inference engine according to the expert's knowledge stored in the rule set. Material property and historical performance data are stored in two separate databases and can be manipulated, added to and interrogated using the GURU expert system shell.

Ranking Score Summary:
Mat'l Property Wt = 0.2000 Atlas Cell Test Wt = 0.2000
Performance Wt = 0.6000

Coating Name	Score: Mat'l Prop.	Atlas Test	Perform.	Total	COST
PENNGUARD BLOCK SYSTEM	38	100	100	88	40
STEBBINS LINING SYSTEM	38	75	80	71	30
214 TL	75	75	64	68	20

Fig. 6. Ranking score summary for qualified coating materials.

NEURAL NETWORKS

Theoretical background

Definitions. An artificial neural network attempts to model mathematically a biological brain by forming an interconnected system of neutrons called nodes. Although the first mathematical model of a neuron was formulated in 1943 by McCulloch and Pitts [17], using neural networks to solve practical problems has only recently become feasible with modern computers. Figure 7(a) shows the essential components of this model: the inputs, summation unit, thresholding unit, and output. The inputs come from switches, sensory devices and other nodes. The summation unit computes a weighted sum of the inputs and passes this sum to the threshold unit. Then the threshold unit subjects the sum to a threshold function and passes the result to the output. The result is called the computed output.

The output is then compared to the desired output which is obtained from a training sample. If there is a discrepancy between the desired output and the computed output, the weights of the weighted sum are changed to reduce the discrepancy.

Two commonly used threshold functions are shown in Fig. 8. Figure 8(a) shows the hardlimiting threshold function. If the sum entering the threshold unit is negative, the threshold unit outputs 0 meaning 'no', 'false' or 'don't accept', but if the sum is zero or positive, the threshold outputs 1 ('yes' or 'true'). The function is named 'hardlimiting' because of the binary nature of the decision.

Figure 8(b) shows a smoother threshold function called the sigmoid threshold (the name is due to the S-shape of the curve). This function is defined as $f(x = 1/(e^{-x} + 1)$. For highly negative inputs, this

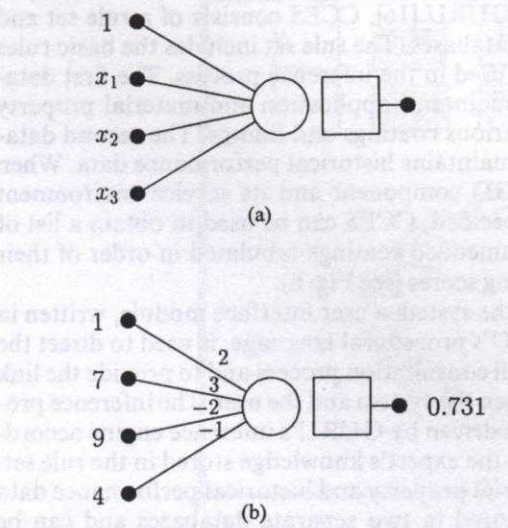


Fig. 7. (a) The McCulloch–Pitts model of a neuron. (b) A neuron with example weights. The top input is clamped at 1.

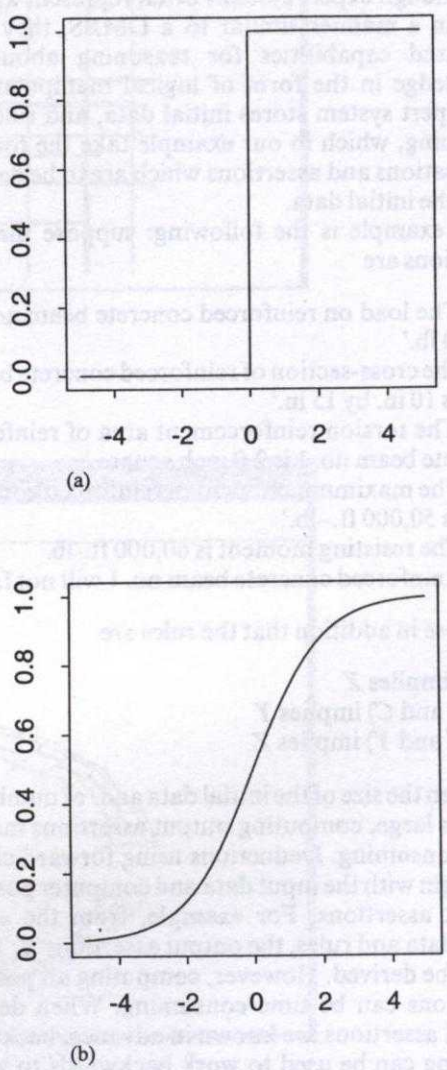


Fig. 8. (a) The hardlimiting threshold function. (b) The sigmoid threshold function.

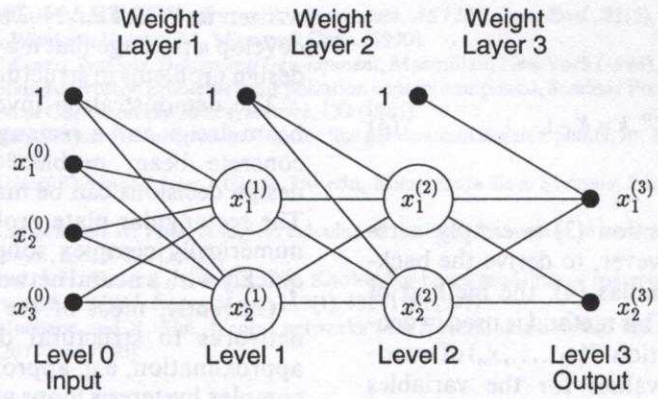


Fig. 9. A neural network with three layers.

threshold function outputs a value close to 0; for highly positive inputs, the function outputs a value close to 1. However, for intermediate values a definite decision is postponed: a number between 0 and 1 is returned. The closer the return value is to 0.5, the more indecision is present. The sigmoid threshold function is useful because it is differentiable, which will be useful in the back-propagation algorithm described later in this section. In this algorithm, the input weights of a node are changed or adapted to allow a better agreement between the desired outputs and the computed outputs.

As an example, consider the weights 2, 3 −2, −1 shown in Fig. 7(b). The top input is clamped at 1 and the inputs are $x_1 = 7$, $x_2 = 9$ and $x_3 = 4$. The weighted sum passed to the threshold unit is $2\cdot1 + 3\cdot7 + (-2)\cdot9 + (-1)\cdot4 = 1$. The sigmoid threshold unit then computes to $f(1) = 1/(e^{-1} + 1) = 0.731$.

Supervised learning. Neural networks are used in supervised learning situations. This means that training data are collected and used to train (or calibrate) the neural net. The inputs are presented cyclically to the neural net which calculates the output. If there is a discrepancy, the weights are adjusted and the training cycle is repeated. If the training is successful, the inputs will result in correct or nearly correct outputs for every possible input.

Ghaboussi *et al.* [18] use a neural net to approximate the hysteresis loops describing the behavior of concrete in the state of plane stress in two situations: monotonic biaxial loading and compressive uniaxial cycle loading. The stresses for the two axes (σ_1 and σ_2) and the strains for the two axes (ε_1 and ε_2) are obtained for a set of uniformly spaced time points. The differences $\Delta\sigma_1$, $\Delta\sigma_2$, $\Delta\varepsilon_1$ and $\Delta\varepsilon^2$ are computed from σ_1, σ_2, ε_1 and ε_2 respectively as the value for the current time point minus the value for the previous time point. Ghaboussi *et al.* [18] then construct a neural network with four levels of nodes (k = 3). The inputs (level 0) are σ_1, σ_2, ε_1, ε_2, $\Delta\sigma_1$ and $\Delta\sigma_2$ and the outputs (level 3) are $\Delta\varepsilon_1$ and $\Delta\varepsilon_2$. There are 40 nodes each in layers 1 and 2. The computed outputs—the changes in strain for the two axes—are then compared with the changes in strain which were obtained experimentally. The weights are adapted to produce a closer agreement and the training process is repeated.

In cases where complex relationships exist between the inputs and the outputs, the neural net requires many thousands of iterations before the net produces nearly correct responses. The neural net described in the previous paragraph required abut 30,000 training cycles. In general, a neural net will never produce a 'correct' answer; it will only produce successively more accurate approximations to the correct answer. A correct answer is 'found' when the change in weights from one iteration to the next becomes smaller than some predetermined amount.

Feedforward networks and the back-propagation algorithm. In order to perform calculations, the nodes are often interconnected so that the output from one node becomes one of the inputs to another node. Ideally, the nodes could be constructed on special hardware chips so that each node could run independent of every other node. However, such hardware is expensive to construct, so neural networks are also simulated in software which runs on conventional computers.

In principle, any node could be connected to any other node, including itself, but often it is useful to organize the nodes into layers as shown in Fig. 9, where the number of layers is $K = 3$. Outputs from one layer are passed to ever node in the next layer in the feed-forward phase. Then the computed outputs are compared with the desired outputs and the weights are corrected in back-propagation phase. The feed-forward and back-propagation phases are repeated, perhaps several thousand times, until the weights and outputs cease to change substantially. The actual computation formulas are shown below using the sigmoid function $f(x) = 1/(e^x + 1)$. The values $x_i^{(o)}$ are the inputs and $x_i^{(k)}$ is the value of the node i in layer k. The desired output values are $d_j, j = 0,\ldots,N^{(k)}$.

Feed-forward phase

$$x_j^{(k-1)} = f\left(\sum_{i=0}^{N_k} w_{ij}^{(k)} x_i^{(k-1)}\right),\ j = 1,\ldots,N_{k+1},\ k = 1,\ldots,K \tag{3}$$

Back-propagation phase

$$\Delta_j^{(K)} = x_j^{(K)}(1-x_j^{(K)})(x_j^{(K)} - d_j)$$

$$\Delta_j^{(k)} = x_j^{(k)}(1-x_j^{(k)})\sum_{u=1}^{M_k} \Delta_u^{(k+1)} w_{ij}^{(k+1)}, \; k = K-1,\ldots,1 \qquad (4)$$

$$w_{ij}^{(k)} \leftarrow w_{ij}^{(k)} - c\Delta_{ij}^{(k)} \quad k = 1,\ldots,k$$

The feed-forward equation (3) is simply a re-expression of Fig. 3; however, to derive the back-propagation update formulas (4), the method of steepest descent is used. This method is used in general for minimizing a function $F(x_1,\ldots,x_n)$ of n variables: choose starting values for the variables $x_1,\ldots,x_n$ and then iterate using the update formula

$$\Delta_i = \frac{\partial F}{\partial x_n}, \; x_i \leftarrow x_i - c\Delta_i \text{ for } i = 1,\ldots,n \qquad (5)$$

repeating (5) until the variables cease to change significantly.

$$E = \frac{1}{2}\sum_{j=1}^{N_k}(d_j - x_j^{(K)})^2 \qquad (6)$$

To obtain the back-propagation equations (4) for updating the weights, apply the method of steepest descent to the following error function E of the weights
and use the fact that $\partial f/\partial s = f(s)(1 - f(s))$ for the sigmoid threshold function. $f(x) = 1/(1 + e^{-x} + 1)$. Equation (4) is applied to all of the weights beginning at layer K and proceeding backwards to layer 1.

When using neural nets, the following design issues and parameters must be addressed: the number of hidden layers and the number of nodes in each of those layers, the method for initializing the weights, the choice of the constant c. The choice of parameters is crucial to the performance of the neural net, but predicting how changes in the parameters will impact on performance is difficult. More research is needed to reduce the uncertainty.

Application

NNISE [19] stands for Neural Networks in Structural Engineering. Similar to NNEPC, NNISE can be trained based on observed information, and differing network types are discussed. The system is written in FORTRAN. The goal for NNISE is to develop a package that may be widely used to solve design problems in structural engineering.

The demonstration involves a simple concrete beam design and a rectangular plate analysis. The concrete beam problem indicates that typical design decisions can be made by neural networks. The rectangular plate problem demonstrates that numerically complex solutions can be estimated quickly with a neural network.

Currently, most of the applications of neural networks to structural design involve function approximation, e.g. approximating the sometimes complex hysteresis loops which describe the force–displacement relationship of a structural member under cyclic loads. Describing these relationships with precise rules may be difficult, but by presenting a neural network with enough examples, the hysteresis loop may be reproduced under new situations. Where precise rules are available, neural networks work best. As research in these two areas matures, the two approaches can be expected to complement each other.

CONCLUSIONS

An evolution of the traditional structural design methodology is currently underway, driven by the rapid advances in computer technology. Structural design is becoming more software dependent and more software intensive. The success and pace of this evolution depends on the rapid and economic development of automated systems. The mathematical background for automation is described in this paper. Modern automation technologies are expected to lead to higher software productivity, and great improvements in software quality, reliability, maintainability, flexibility for modification, and extendability.

The presented applications to structural design are examples of the four types of automation. Computer hardware and software potential has created an unprecedented opportunity to develop automated systems for structural design. Although some progress has been made in this area, much remains to be done.

REFERENCES

1. M. P. Groover, *Automation, Production Systems, and Computer-Integrated Manufacturing*, Prentice-Hall, Englewood Cliff, NJ (1987).
2. R. S. Pressman, *Software Engineering*, McGraw-Hill, New York (1987).
3. A. Habibullah, ETABS, Computers & Structures, Inc., Berkeley, CA (1990).
4. C. J. Date, *An Introduction to Database Systems*, 2nd edn, Addison-Wesley, Reading, MA (1977).
5. M. L. Maher, *Expert Systems for Civil Engineers: Technology and Application*, ASCE (1987).
6. L. Brownston, R. Farrell, E. Kant and N. Martin, *Programming Expert Systems in OPS5*, Addison-Wesley, Reading, MA (1985).
7. J. M. Zurada, *Artificial Neural Systems*, West, St Paul, MN (1992).
8. O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method*, 4th edn, vol. 1, McGraw-Hill, New York (1989).
9. E. L. Wilson and A. Habibullah, SAP90-Static and Dynamic Finite Element Analysis of Structures, Computers & Structures, Inc., Berkeley, CA (1991).
10. M. Krohn, *Using the ORACLE Toolset*, Addison-Wesley, New York (1992).

11. Y. L. Mo, Static structural integrity monitoring system, *ASTM J. Test. Eval.*, **21**(5), 453–460 (1993).
12. Microsoft, Windows Version 3.1, Microsoft Corp. (1990).
13. J. Durkin, *Expert Systems, Design and Development*, Macmillan, New York (1994).
14. NACE, Solving corrosion problems in air pollution control equipment, Seminar Preprints. National Association of Corrosion Engineers, Denver, CO (1981).
15. Y. L. Mo, Expert system to choose coatings for flue gas desulphurisation plants, *Br. Corros. J.*, **28**(3), 188–193 (1993).
16. MDBS, *GURU Command User's Guide*, 3rd edn, Micro Data Base Systems, Inc., Lafayette, IN (1991).
17. W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys.*, **5**, 115–133 (1943).
18. J. Ghaboussi, J. H. Garrett, Jr. and X. Wu, Knowledge-based modeling of material behavior with neural networks, *ASCE J. Engng Mech.*, **117**(1), 132–153 (1991).
19. R. D. Vanluchene and R. Sun, Neural networks in structural engineering, *Microcomput. Civil Engng*, **5**, 207–215 (1990).

S. D. Jost is an associate professor of computer science and information system at DePaul University, Chicago, IL. He received his Ph.D. from Northwestern University, Evanston, IL. His research interest is in the area of application of artificial intelligence to engineering.

Y. L. Mo is a professor of civil engineering at National Cheng Kung University, Tainan, Taiwan. In the summer of 1995 he was Alexander von Humboldt visiting professor in the Institute of Structural Mechanics, the University of Hannover, Hannover, Germany. He received his DrIng. degree from the University of Hannover. He has been active in research involving tests on structural components, including bridges and buildings, and in the application of artificial intelligence to concrete structures. Dr Mo has published more than 50 technical papers in international journals and has authored a book on the dynamic behavior of concrete structures.

H. C. Wu is a lecturer of management of information system at Kung Shan Institute of Technology and Commerce, Tainan, Taiwan. She received her MS degree from DePaul University, Chicago, IL. Her research interest is in the area of commercial automation.