SATS: a microcomputer-based support for teaching structural analysis

Ronald Corovic, José A. Pino and Mauricio Sarrazin*

*Depto. de Ciencias de la Computación and *Depto. de Ingeniería Civil, Universidad de Chile, Casilla 2777, Santiago, Chile*

This paper describes an educational software system for structural analysis. The system is intended to develop the students' intuitive skills to build effort diagrams for plane frames. It allows the students to draw the shear and moment diagrams for a given problem on the terminal screen, and then, to compare them with the correct ones. The instructor can generate an arbitrary number of new problems and store them on disk. During his session, the student may choose the problems he wishes to work on, according to his knowledge and interest. Moreover, the student can modify a problem statement to ponder the effect of those changes on the effort diagrams.

1. Introduction

Brohn & Cowan [1] have published data concerning the understanding of structural behaviour by civil engineering students and graduates. Their results are discouraging: many subjects who have followed or are following a standard curriculum in structural engineering are unable to sketch the shear force and bending moment diagrams for a plane frame without solving the corresponding equations.

An intuitive knowledge of structural behaviour is then needed to really understand the subject [2]. Moreover, such knowledge must be based on visual experiences, rather than numerical analysis.

A number of batch computer programs exist to compute structural behaviour, such as CAL [3], but such systems can be considered more like an extension of a calculator than a device to improve the student's understanding. CAL-GRAF [4] is a CAL extension which allows the student to define a structure and to display deflections and stresses using a set of commands.

Moss, Knowles & Ahmad [5] built a package to allow students to interactively design steel beams. The computer checks, after each decision, its correctness and convenience.

Smith [6] developed STP, a program that displays a menu of ten problems to work on. For each problem, the student is asked to give the value of the bending moment diagram at the critical points in a scale from −10 to +10; STP then plots the diagram thus defined and the correct diagram as well as a mark awarded so that the student can assess his performance.

Slater, Petrossian & Shyam-Sunder [7] describe MACAVITY, an expert system which generates example beam problems or allows them to be input by the student. Then, the program accepts student input equilibrium equations, solves them simultaneously if required and judges the correctness of the student's solution. Diagnostics on the sources of errors in a set of equations are given to the student, who can interactively try again.

2. The design of SATS

Structural Analysis Teaching Support (SATS) is a program designed to execute in a normal microcomputer environment. It runs on an IBM PC (or compatible) computer with graphics capabilities and a mouse.

Those facilities are enough to provide an extensible and playful environment where the student can improve his intuitive abilities in structural analysis by sketching shear and bending diagrams on a variety of problems.

SATS consists of two modules: PRECEPTOR and LEARNER. The first is used to update a plane frame problems database, whereas the second must be run to practise solving those problems.

The problems database consists of a set of structural analysis problems of the *plane frame* type, entered by the instructor by means of the PRECEPTOR module. A plane frame problem is defined by a number of straight *elements*—also called *members*—laying on a plane and joined at their ends by *nodes*, that is, ficticious points where the continuity of displacements and rotations is maintained. In a few cases, the continuity is only partial, as it is the case of the *internal hinge* node.

The members are subjected to external loadings consisting of concentrated and/or distributed loads. The nodes may also have external loads or, instead, displacement restrictions, in which case they are called *supports*.

Thus, a model includes members, nodes, external loadings and supports. A *three-span continuous beam*, for instance, is represented by three consecutive straight lines connected to four nodes, each of them externally supported. Almost any problem of the plane frame type can be modeled using the components provided by the program.

The LEARNER module offers a menu containing the database problem names. Once the student chooses a statically determinate or indeterminate problem, he can use the mouse as a pointing device to draw the shear or bending diagrams on top of the frame presented to him (see Figure 1). Then he can ask the system to plot the true diagram for a visual comparison. To do this, the system uses the direct rigidity matrix method. (Figure 2).

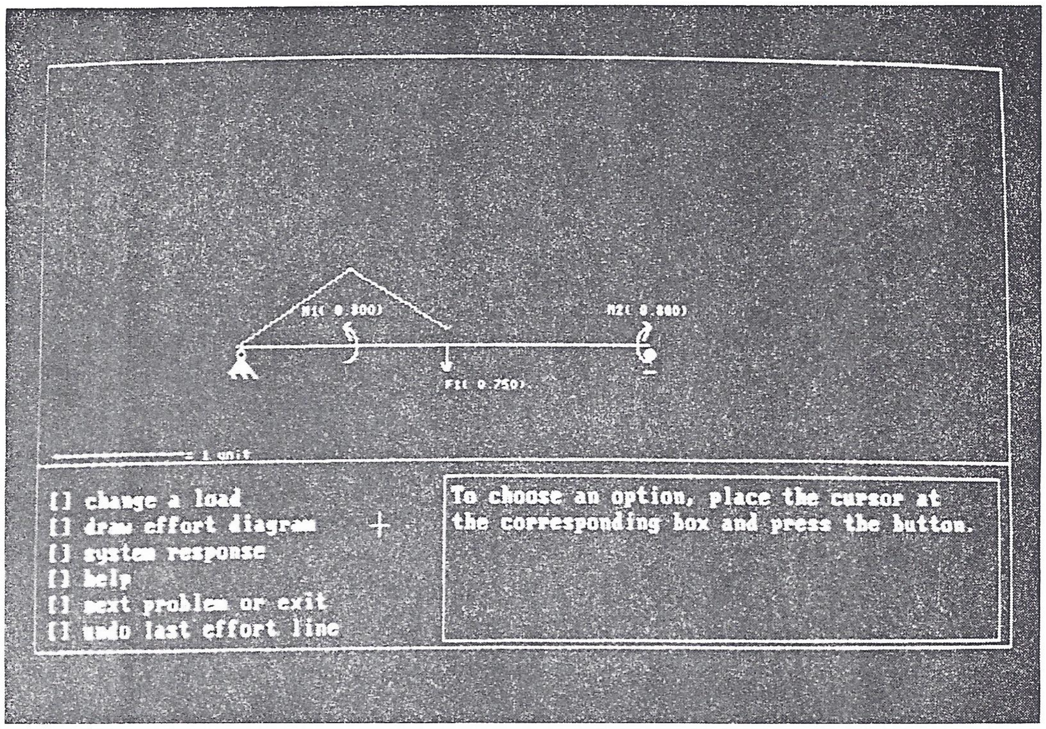
The student can then work on a different problem, or if he feels he needs more practice on the current problem, he can modify certain data (such as load values and angles) and try again.

LEARNER does not attempt to grade the student's achievements. The reason for this is that grading is a complex task done better by the instructor (e.g., how many points should be awarded to an answer including a curve which is short by 25% on one critical point *vs* another giving a straight line instead of a curve?).

A context sensitive HELP option is always present to explain how to continue in the current situation (see Figure 3), and error messages are also informative on the procedures to either do more work or exit.

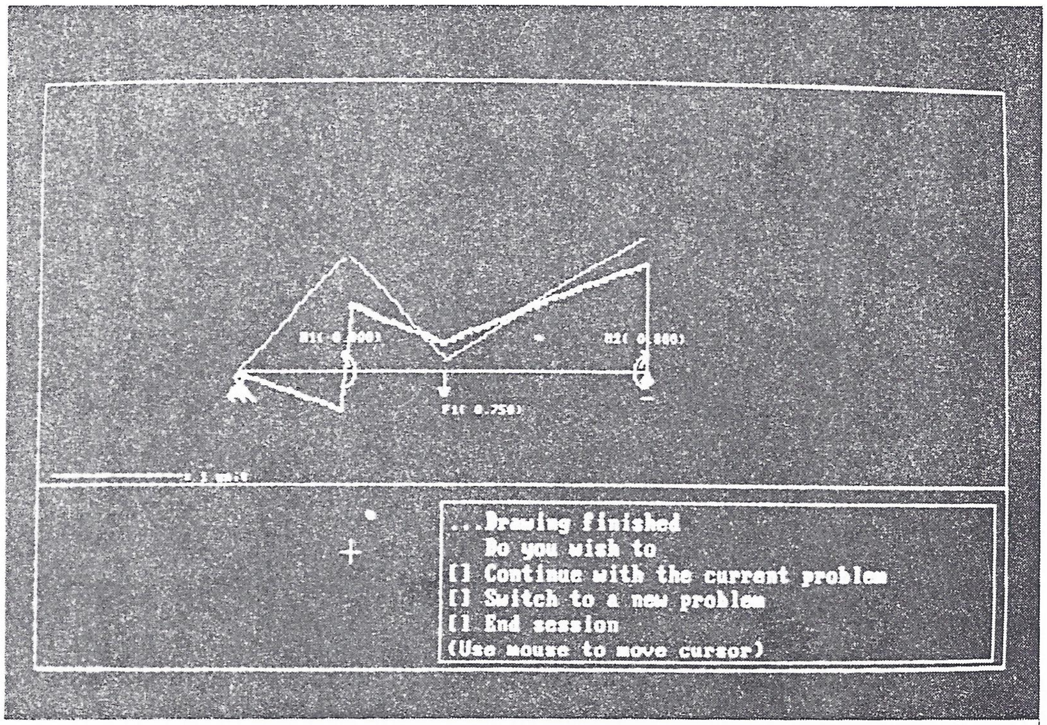
The PRECEPTOR module provides an interactive menu-based graphics environment to create, change and delete models of the problems database.

To add a new model to the database, the instructor chooses length units, and draws nodes, elements, supports, internal pins, loads (arbitrary angle), and moments. A simple editor helps to iteratively complete the drawing. Again, a HELP option provides the necessary guidance. Once the model is ready, the instructor may store it in the database by simply choosing the corresponding menu option. For example, all problems proposed by Brohn & Cowan [1] can be stored in the system.



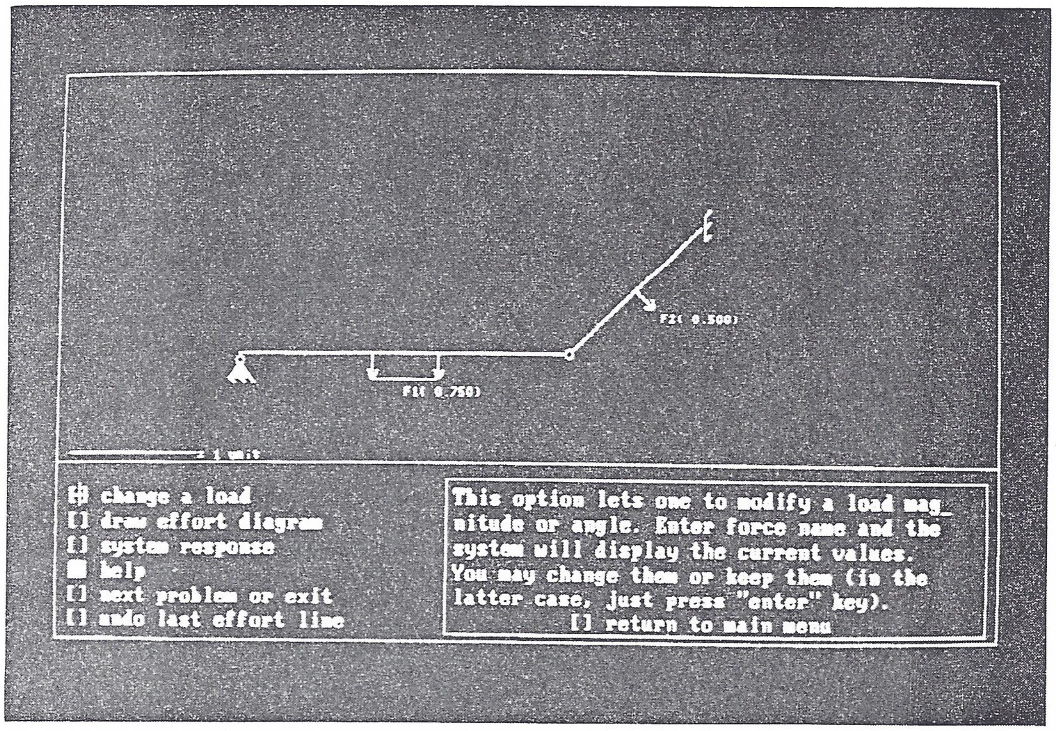
To choose an option, place the cursor at the corresponding box and press the button.

Figure 1. Using LEARNER to draw the bending diagram for a statically determinate problem.



...Drawing finished
Do you wish to
[] Continue with the current problem
[] Switch to a new problem
[] End session
(Use mouse to move cursor)

Figure 2. LEARNER has depicted the correct bending diagram (with thick line) for the same problem.



```
This option lets one to modify a load mag_
nitude or angle. Enter force name and the
system will display the current values.
You may change them or keep them (in the
latter case, just press "enter" key).
          [] return to main menu
```

Figure 3. HELP has been asked to LEARNER to explain load modification.

3. Implementation of SATS

The programs are written in TURBO-PASCAL [8], which provides graphics primitives through its package TURBO-GRAPHIX [9]. The object code of the LEARNER and PRECEPTOR modules take 91 and 75 Kbyte of main memory, respectively. Since TURBO-PASCAL requires that the object program must fit a 64 K storage assignment, an overlay structure for each program was implemented.

A models database is stored in floppy or fixed disks as a set of files of the following types:

Parameter file: a unique file containing the name of each problem of the database with an entity file name associated with it.

Entity file: there is one of these for each model. It contains all the relevant information, i.e., descriptions of every element, support, internal pin, load or moment of the model. The data structure of this type of file is described below. Figure 4 illustrates a portion of the entity file contents for the model shown on Figure 1.

There are two types of records in an entity file. A node record contains its name, X and Y coordinates, and a pointer to the next node record. A component record contains the following fields:

a specification of the component type,

its name (only in the case of loads and moments),

pointers to two node records, specifying the location of the component within the model,

two real fields containing data, which is specific for each type of component. For example, for loads, these fields contain the value of the load and the angle with respect to the X axis.

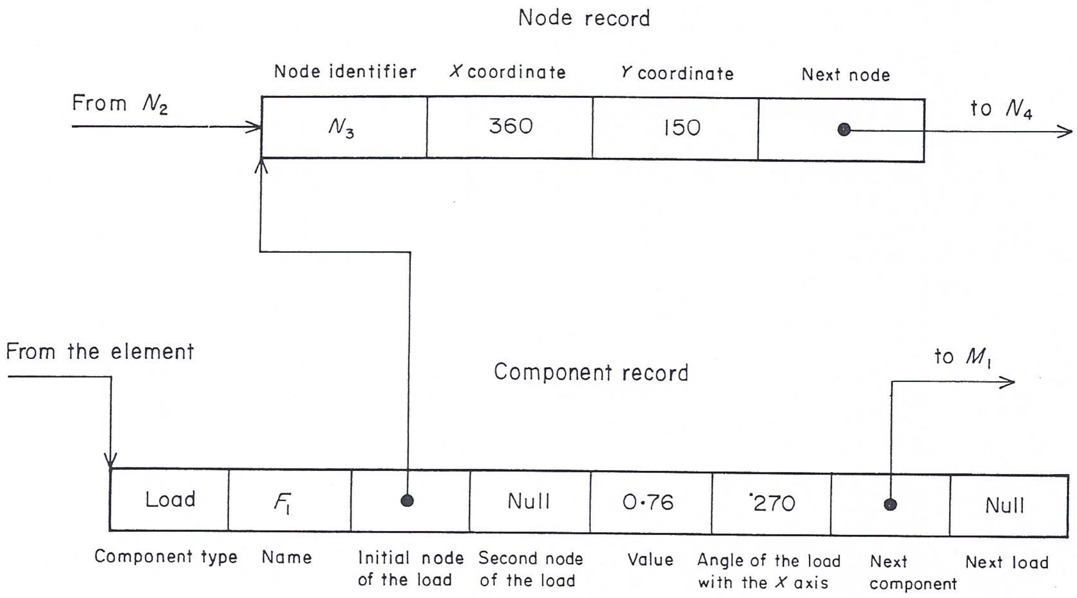


Figure 4. Component record for load F_1 and node record for node N_3 for the model shown in Figure 1. Node N_3 is unnamed in the diagram. The *second node of the load* pointer in the component record is used for distributed loads.

a pointer to the next component record, and

a pointer to the next component record of the same type; this field is used only for loads.

This data structure was designed to get efficiency during execution, besides economy in storage space.

Concerning efficiency during execution, LEARNER loads into memory the parameter file. When the student chooses to work on a specific problem, LEARNER also loads the corresponding entity file, and draws the model using the component records (and its pointed node records). The linked list of load records speeds up the remaining processing, particularly when the student modifies the model.

Storage space used is very small. On the average, a model is described by 350 bytes in an entity file. The parameter file is similarly small: 88 bytes per problem is the average space used in a typical database. Therefore, a complete problems database fits a normal diskette.

In our implementation, a 4 MHz CPU was used to run the system. The response time was adequate; the largest delay (10 s) was to obtain the plot of the true diagram of a 10-element problem. That problem size is the maximum allowed by LEARNER 1·0. Another version of the program—LEARNER 1·1—uses disk storage for intermediate results during computations; it allows larger problems, but its response time for plotting the true diagram is increased nearly twofold. It was found that a 10-element maximum problem size is enough for a normal database and therefore, LEARNER 1·0 is being used. Of course, a faster CPU should provide quicker response times.

Due to its modular design and construction, the system can be modified to be useful in other intuitive learning applications. In particular, the problem components and the true diagram computing algorithm can be easily changed.

4. Conclusion

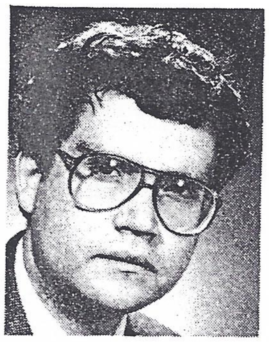
With very modest hardware, a software system has been built to provide a helpful and amusing environment to develop intuitive abilities in structural analysis students.

The high interaction allowed by SATS lets the student to practice sketching diagrams and immediately obtaining the right answers. This can be done at the student's own pace, doing non-repetitive work on a problem for as long as the student feels he knows it.

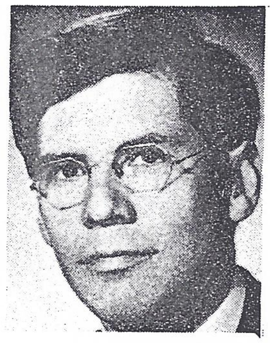
The instructor's role in the use of SATS is important in preparing (or updating) the problems database, motivating the students to learn with the system, and helping them to explain why a certain correct diagram has the given shape. Of course, an expert system could be built to assist in the latter task.

References

1. D. M. Brohn & J. Cowan 1977. Teaching towards an improved understanding of structural behaviour. *The Structural Engineer,* **55** (1), 9–17.
2. D. M. Brohn 1983. Academic priorities in structural engineering—the importance of a visual schema. *The Structural Engineer,* **61A** (1), 17–19.
3. E. L. Wilson 1979. CAL—A computer analysis language for teaching structural analysis. *Computers & Structures,* **10,** 127–132.
4. K. R. Leimbach 1981. **14,** CAL-GRAF—A computer-graphics supplement for CAL. *Computers & Structures,* **14,** No. 1–2, 135–141.
5. W. D. Moss, P. R. Knowles & K. Ahmad 1979. CAL packages for civil engineering hydraulics and structural design. *Computers & Education,* **3,** 391–399.
6. J. W. Smith 1984. Using computers to teach structural analysis. *Computers & Education,* **8** (1), 101–105.
7. J. H. Slater, R. B. P. Petrossian & S. Shyam-Sunder 1985. An Expert Tutor for Rigid Body Mechanics: ATHENA CATS—MACAVITY. *Proceedings of the Expert Systems in Government Symposium*, IEEE/CS, McLean, VA, 1–11.
8. Borland International 1985. *Turbo-Pascal Reference Manual (version 3.0).*
9. Borland International 1985. *Turbo-Graphix Reference Manual (version 1.0).*



R. W. Corovic (computer engineer, Universidad de Chile) was a research assistant at the Computer Science Department of the Universidad de Chile. He currently works for the Banco de Chile. His interest areas are discrete event simulation and educational systems development.



J. A. Pino (MS, MSE, The University of Michigan, USA) is associate professor of computer science at the Universidad de Chile. He is the author of several papers and co-author of three books. His current research interests are human-computer interfaces and information retrieval systems.

M. Sarrazin (MS, DSc, Massachusetts Institute of Technology, USA) is professor of civil engineering at the Universidad de Chile and Dean of the School of Engineering and Science at the same university. He has published works in the area of structural engineering, particularly in computer assisted structural analysis and design.