Int. J. Engng Ed. Vol. 8, No. 6, pp. 472–476, 1992
Printed in Great Britain.

0742–0269/91 $3.00+0.00

Teaching Computing with a New Tool: Computational Recipes*

G. AKHRAS†
Department of Civil Engineering, Royal Military College of Canada, Kingston, Ontario, Canada, K7K 5L0

As computers make more and more incursions into all fields of engineering, undergraduate education must undergo some necessary changes to incorporate the various new technologies. To do so, new approaches are needed. One such approach is the concept of computational recipes. In this paper, the above issues are discussed in detail and the new concept of computational recipes is presented with examples related to numerical analysis for civil, mechanical, aeronautical and aerospace engineering. This new concept, if adopted, will greatly influence the computing part of the engineering curriculum.

INTRODUCTION

SINCE computers first appeared, computing has been introduced gradually in many, if not all, fields of engineering curriculum, so that now it is considered a fundamental pillar of engineering education. For courses requiring heavy mathematical and numerical calculation, the change was important. With every generation of computers, the contents of these courses have to be adapted accordingly. The introduction of the microcomputer produced a drastic change which is still going on. Now that the evolution of the computing industries is occurring so quickly, changes to the course content have to be continuous.

Most of the engineering educational programs accredited in North America (ABET, Accreditation Board for Engineering and Technology, USA; CEAB, Canadian Engineering Accreditation Board, Canada) include computing as an important and integral part of any engineering curriculum. The teaching of computing consists primarily of developing the computer programming, expertise and know-how of the student. Implicitly, this requirement consists of learning how to develop, write, debug and run a program, as well as how to properly use and run existing programs and software. In both cases, the student learns not only how to get the right results but exactly how the computer reaches the solution. Until recently, the attempt to reach this idealistic objective was more or less successful. Nowadays, it is extremely difficult, if not impossible, to perform this task in a satisfactory manner. Despite this fact, the above requirement is still present in all curricula.

The rate of transforming the curriculum to reflect and adopt all the new computing techniques is not following the fast rate of their evolution. Consequently, many engineering educational programs are lagging behind, and very often, it is left to the individual faculty member to incorporate the required changes to the courses.

On the other hand, the miniaturization of hardware and the introduction of the microcomputer produced a drastic change in computing and in the way we deal with computing. What was expensive, remote and inaccessible, is now available and cheap. Suddenly, most of the complex numerical and computational techniques developed specifically to reduce cost and effort, lost their '*raison d'être*' and consequently their value for educational purposes. This situation is likely to repeat itself because computer technology is evolving at a very fast rate.

To deal with the above concerns, a reassessment and a reorganization of the teaching of computing has to take place, and new approaches have to be developed. In this paper, the concept of computational recipes is introduced and compared to the present approach of teaching computing. This new concept, if adopted, will greatly influence the computing part of the engineering curriculum. In a conventional engineering curriculum, computing is introduced to the student throughout the full academic program. This is usually done both in a direct and indirect fashion, and examination of both techniques is considered below.

DIRECT LEARNING

The direct teaching of computing can generally be divided into three separate but interrelated courses.

Introductory course

All the engineering schools and universities have in their programs, for their preparatory or first year, a general computer course introducing the

* Paper accepted 20 November 1992.
† George Akhras is Associate Professor of Civil Engineering.

student to the computing world. The contents of this course varies tremendously and can be summarized by one or more of the following items:

(1) a general introduction to computing and computing methods for engineering problems;
(2) an introduction to computing languages with emphasis on a particular one. Traditionally, the most taught language has been FORTRAN, but PASCAL, APL, BASIC or even C may be taught.
(3) an introduction to numerical methods and analysis;
(4) an introduction to what the author defines as the basic utility software: spreadsheets, word processors and graphics, via well known commercial software, i.e., Lotus, WordPerfect, AutoCad, etc.

Numerical analysis course

This intermediary course is a follow-up to the previous course and is taught in the middle of the Program. It mainly consists of exposing the student to the numerical techniques required in the various disciplines of engineering. The basic objective is to provide the numerical expertise, and practical experience with systems and devices, so that the student will progress easily and comfortably in the computing manipulation of the succeeding engineering courses.

The contents of this course include many of the following topics:

(1) review of linear algebra including matrix operations;
(2) solution of a system of equations, the Gauss solver and its many variations;
(3) ordinary and partial differential equations, and their applications to a particular discipline;
(4) eigenvalues;
(5) applied regression analysis;
(6) numerical integration;
(7) finite difference method;
(8) finite element method;
(9) stochastic processes;
(10) system approaches.

The computer is used systematically in teaching these topics. At least five or six of them constitute the basis of this course. The selection of a particular topic versus another depends solely on the teacher's background and ease in teaching the subject. Also, courses have been set and organized differently from one university to another, to complement the particular need of a curriculum program.

Advanced computer/numerical analysis course

In the final year of the curriculum, an advanced computer/numerical course is normally devoted to a particular discipline; for example, the computer analysis of structures is adapted for civil, mechanical, aeronautical and aerospace engineering. Usually, this course is considered an important one and is heavily oriented toward using and manipulating computer programs and software. In practice, computer manipulation is included in each assignment and classwork. This includes reading, writing, debugging, and manipulating computer codes and graphics, assuming that, as a result of the two previous courses, the student is comfortable and familiar with the programming languages, techniques and styles used for this course.

INDIRECT LEARNING

Many courses have adapted the computing process to their particular needs and, indirectly, they are reinforcing the computing knowledge of the student. It is difficult to assess the benefit to the student of this indirect exposure, but it certainly contributes to the learning process.

Another form of indirect learning is due to the fact that more and more teachers are requesting students to use the basic utility software (spreadsheets, word processing and graphics) to prepare their assignments. This practice encourages the student to become familiar with the computer.

ASSESSMENT OF THE PRESENT APPROACH

The field of computing is evolving very rapidly due to the fast growth of new ideas, methods and approaches. To take advantage of the accumulated expertise and knowledge, more and more software development groups tend to be multidisciplinary. These groups are made up of individuals with different backgrounds, for example, computer science, numerical analysis, mathematics, optimization, and each provides a specific expertise to the team.

Consequently, many computer programs are the products of the knowledge, experience and expertise of more than one developer. They incorporate a great deal of the collected expertise on a specific subject, in order to assist and guide the user in solving problems and reaching the correct results. The sophistication of computer programs has evolved so fast and in such a way that it is more and more difficult, if not impossible, to perceive how the results are produced when you use a computer program. This process is so time-consuming, that more and more of these programs are used without complete details of the computing process being known.

If, as expected, teachers are continuously updating and adding to the contents of their courses, to reflect the new technologies, how will an 'average' student cope with all the extra information and pressure? Unfortunately, not very well.

With all the added complexity and sophistication in computing, most of the students are lost between learning the fundamentals of a topic, and learning

the computing related to that particular topic. Ideally, they would be learning both. In fact, they end up learning a little bit of both and not very well. Worse still, because computing is considered 'fun' and 'smart', they may end up learning the computing part of the course rather than the actual course material.

A typical example

The following example, related to numerical analysis, will help explain the situation described above and facing educators in all fields of engineering. Many recent textbooks of structural analysis [1–4] include a detailed chapter or appendix on two or three different equation-solvers with corresponding programs and subroutines. One may find, for example, a routine for a straightforward resolution of a system of equations, another for the solution of a system of banded equations, and another using the active column approach, etc. The authors of these textbooks expect educators and students to spend some time in writing codes, as well as debugging and thoroughly understanding these routines; otherwise they would have not included all the details in their books.

On the other hand, every computer centre in any educational institution has commercial engineering/scientific/mathematical software with one or more versions of the above equation solvers [5, 6]. These commercial programs are structured and written by professional programmers and/or computer scientists, and may outperform, in space requirements and resolution time, the corresponding programs written by the academic engineering community.

An undergraduate student obviously has a fixed amount of time to spend on learning equation-solver techniques. So, what is the best alternative for him/her: learning the basis of equation-solving or learning the computing that is related to equation-solving? Developing the computing expertise of equation-solvers or developing the computing know-how, again assuming that there is not enough time to learn both? Is it better to spend time on deciphering computer programs or to learn the course material by manipulating these programs, assessing their intrinsic values and selecting the right and appropriate program for the application at hand?

The academic community may surmise that manipulating computer codes is a good learning exercise. The same can be said of any course material, especially of structural analysis and design, which is more appropriate to the civil, mechanical, aeronautical and aerospace engineering professions. To deal with this concern, a new approach, a revision or a reorganisation of the present method of teaching and handling computing is required.

GUIDELINES FOR NEW APPROACHES

The following few guidelines should be considered before proposing any change:

Engineers vs computer scientists

- The aim and ultimate objective of the engineering curriculum is to teach and educate engineers, not computer scientists. Unfortunately, many courses focus more on computing than on engineering.
- In the last few years, the volume of knowledge has expanded significantly and there is a tendency among administrators and faculty members to expand the undergraduate curriculum. Increasing computing courses seems to overshadow the increase in engineering courses in the curriculum.
- If a student has some free time and wants to learn something new, it should be linked to a topic related to engineering not to computing. Unfortunately, and with the approval of academics, many students spend hours learning about new computing 'gadgets' instead of dealing with engineering material.

Teaching all the students not just the best ones

The objective of any faculty member is to teach *all* the students in a class. In fact, the best students in a group are going to excel with, or maybe without, faculty help. What about the other ones? If the right priorities are not set properly in a course, an average or below average student may become lost and spend more time and effort tinkering with the computer than learning the course material. Mastering the fundamentals, the assumptions and the formulation of a particular engineering application, is far more critical to the education of an engineering student than learning programming techniques and tricks to produce the software that handles the application.

Leaving pure computing to the computing community

If the subject at hand is pure computing, it should be left to the computing community which has the appropriate knowledge and expertise to handle it. The engineering community should benefit from all the expertise it can get without recreating it. The typical example of equation-solvers mentioned above is a good example for this guideline: the programs and the professional software for the solvers should be used.

Recently, new ideas, methods and approaches have been put forward to tackle this problem. For example, in the advanced structural analysis course, appropriate computer programs [7, 8] have been used to teach structural analysis while programming manipulations are left to graduate courses, and to the research community. Another approach is to use computational recipes.

COMPUTATIONAL RECIPES

The idea behind computational recipes is based on the fact that the ultimate goal of teaching some of the numerical and mathematical topics mentioned above, is to identify their capabilities and limitations, and to exploit them as tools to solve problems. All the available technological means should then be used to reach this goal.

Definition

A computational recipe is an intelligent tutorial system (ITS) [9] which encapsulates all the available expertise on a particular topic and offers judgment and advice to the user's particular application. It is divided into three parts: a preprocessor, the core system and a postprocessor.

The preprocessor is an expert system which will capture the information from the user, process it, make comments on the nature of the problem and propose recommendations on how to solve it. Then with the help of the user, it will select from the core system the best procedure to solve the problem, and run the appropriate corresponding programs.

The core system regroups all the routines, programs and procedures required to run different types and variations of the particular topic of the recipe.

The postprocessor, another expert system, will help the user digest the results.

Features

An attractive feature of this approach is that the student can use the recipe at increasing levels of sophistication, while gaining more experience. The pre- and post-expert systems will provide all the necessary explanation to allow novice and expert users to manoeuvre with ease, at their own pace.

Another feature is that the students will concentrate their efforts on exploiting the recipes to define, circumscribe and solve the problem at hand, instead of manipulating programs and computer codes. Moreover, the intimidation and overload of computer instructions will be replaced by developing the ability of the student to exploit available computer software as a tool.

All other computer-aided instruction (CAI) systems and intelligent tutorial systems (ITS) could be used to develop or improve the pre- or postprocessor part of any recipe.

The basic idea of computational recipe may appear controversial, but it is a very simple and efficient way to continuously upgrade the computing of the numerical methods in engineering. To illustrate this new concept, two examples of computational recipes are presented: a recipe for the equation-solver of a system of linear equations, and a recipe for statistical analysis.

A recipe for equation solvers

The computational recipe for the equation-solver of a system of linear equation is divided in three parts:

(1) The expert sytstem which constitutes the preprocessor of the recipe will analyse the topological properties of the main matrix (density, sparsity, bandedness, profile, wavefront, frontwidth, etc.) and after providing the user with all the alternatives and the corresponding explanations, the system will guide the user to the appropriate action. For example, if the matrix is symmetric and sparse, the system may propose one of the following:

- to reduce the bandwidth before solving the equations with a band-solver;
- to use a frontal solver;
- to reduce the profile in the matrix before using a skyline technique;
- to use any other appropriate program.

(2) the core system will include all the different programs and routines needed to cover all the aspects of the recipe, for example:

- relabelling for bandwidth or profile reduction, etc.;
- Gauss solvers, band-solvers, frontal solvers, skyline solvers;
- Jacobi, Gauss–Seidel, etc. for the iterative method.

(3) The postprocessor will present the results and provide all the appropriate analysis: i.e. CPU time and space used, precision, roundoff errors, etc. Cost may also be provided.

The advantage of this computational recipe is that it collates all the available techniques to solve the systems of equations. Also, it presents them in an overall integrated view so that the user may:

- have an overall view and appreciate the features of each technique;
- have a greater flexibility in selecting one of them;
- exploit the various advantages and features associated with each method;
- use different techniques to check the accuracy and the validity of the solution.

A recipe for statistical analysis

Statistical analysis could be another good application of a computational recipe. Many users of statistical analysis are ill-prepared or not very comfortable with the collection and analysis of data. Users may drop a few odd points, or select tests where assumptions have been violated, and carry on their analysis unaware of the consequences of their decision. The expert systems of the recipes will tell the user what is the nature of the data at hand, and provide advice on what to do, why and how, with all the necessary explanations. This will help increase the awareness of the student to the sensitivity of the data.

ADVANTAGES OF THE COMPUTATIONAL RECIPES

- Different levels of complexity can be included in the recipes, so that the same recipe can be used at different stages of the curriculum.
- The pre- and postprocessor expert systems are structured in such a way that the interactive nature of preparing the input, and interpreting the results, shifts the learning process from passive to active.
- The learning is personalized. Users will follow their own pace in exploiting the same recipe.
- Since the core system is composed of professionally developed programs and routines, a recipe could be useful not only for educational purposes but in the real world. This is an asset for the students: becoming familiar with a recipe, students could use it throughout their careers as engineers.
- The programs and routines in the core system could be changed, updated or rearranged at will without affecting the user.

CONCLUSIONS

There is no simple way to teach computing in engineering. Based on the alternatives of teaching computing expertise versus teaching computing know-how, a detailed evaluation of the present common approach is discussed thoroughly and guidelines for new approaches are suggested.

A proposal for using a new approach in teaching numerical methods in various engineering disciplines is introduced in this paper. The concept of computational recipes is presented and explained with examples. If it is adopted, this new concept will greatly change the computing part of the engineering curriculum.

Acknowledgements—The support of the Department of National Defence of Canada is gratefully acknowledged.

REFERENCES

1. S. H. Holzer, *Computer Analysis of Structures: Matrix Structural Analysis*, Elsevier, New York (1985).
2. C. K. Wang, *Structural Analysis on Microcomputers*, Macmillan, New York (1986).
3. R. J. Melosh, *Structural Engineering Analysis by Finite Elements*, Prentice Hall, New Jersey (1990).
4. T. R. Chadrupatla and A. D. Belegundi, *Introducton to Finite Elements in Engineering*, Prentice Hall, New Jersey (1991).
5. IMSL, International Mathematical & Statistical Library, Problem Solving Software Systems, Houston, Texas, U.S.A.
6. NAG, Numerical Algorithms Group, Inc., Downers Grove, Illinois, U.S.A.
7. P. Jayachandran and S. G. B. Leblanc, Structural Analysis using Microcomputers and Graphics, *Proceedings, ASCE Congress, Computer Applications in Structural Engineering*, Florida (1987).
8. G. Akhras, Teaching Structural Analysis with Computer Programs, *International Journal of Applied Engineering*, **6**, 461–464 (1990).
9. E. Wenger, *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, Inc., California (1987).

Dr Georges Akhras is an Associate Professor of Civil Engineering at the Royal Military College (RMC) of Canada, Kingston, Ontario. For ten years, he was project manager for computer applications, numerical analysis and engineering projects in government and industry before joining RMC in 1987. He is a member of many technical and professional associations. His current interests include expert systems, numerical modelling, composite materials and engineering education, and he has published many papers on those subjects.