DOCUMENT RESUME

ED 084 157 SE 017 006

AUTHOR Alo, Richard A.
TITLE A Collaborative Learning Approach for Undergraduate Numerical Mathematics.
PUB DATE Jun 73
NOTE 14p.; Paper presented at the Conference on Computers in the Undergraduate Curricula (4th, Claremont, California, June 18-20, 1973)

EDRS PRICE MF-$0.65 HC-$3.29
DESCRIPTORS *College Mathematics; Computer Oriented Programs; Course Descriptions; *Course Organization; *Curriculum; Curriculum Development; Group Activities; *Instruction; Instructional Innovation; *Teaching Techniques
IDENTIFIERS Numerical Methods

ABSTRACT

Described is an undergraduate numerical analysis course organized around projects and tasks assigned to student teams. Most teams had five students within which the student with the most computer programing experience assumed the leadership role. The leaders' responsibility extended to distribution of work assignments and coordination of group interaction. Intragroup cooperation, leadership or lack of leadership and assignment of individual final grades are among the topics and problems discussed. (JP)

A COLLABORATIVE LEARNING APPROACH FOR UNDERGRADUATE NUMERICAL MATHEMATICS

by

Richard A. Alo

This paper is based on an undergraduate Numerical Analysis course given by the author at Carnegie-Mellon University during the fall semester of 1972. The course was offered again in the spring semester of 1973 by the author, incorporating the recommendations offered here. In the fall semester there were initially forty students who were broken up into eight groups of five members each. Although this was intended to be a first semester sophomore level course the students were a mix of sophomores, juniors, seniors, and some fulltime employees from industry.

During both semesters five projects to be run on our IBM 360/67 were given, and each group was required to write one report for each project. The students were required to code their own programs rather than just copy a detailed algorithm. Hence each member of the group was responsible for writing one report during the semester and for interacting with the group to complete the other assignments. We assigned three levels of homework assignments: simple problems which could be run off on a calculator, slightly more complicated problems which were better suited for computer programming, and more substantial full scale computer run projects.

The prerequisites for this course were one semester of linear algebra, and an introduction to computer programming. We assumed that the student already had some knowledge of the implications of number representation and finite memory from the latter of the above prerequisite courses. We also assumed that there was ample discussion of the notion of trade-off among speed, space, and accuracy and that there was sufficient practice to become competent in writing and in using simple programs. A good linear algebra prerequisite should include an introduction to numerical computation, for example, solutions of linear systems by Gaussian elimination (with appropriate discussion of scaling, pivoting, propagated and round-off errors) and eigenvalue-eigenvector computations.

This course was one semester in duration and carried nine units (that is, three semester hour credits). There were two meetings per week each consisting of one hour and fifteen minutes of lecture and a thirty minute problem session. The lectures were given by the author. Lecture materials were taken from the texts [2] and [4]. A graduate assistant was responsible for the computer laboratory work and problem sessions. The problem session was mostly devoted to computer projects, but the students, at their option, could discuss the other assignments.

This is the only undergraduate numerical analysis course in the Mathematics Department. The next course to sequel this is a first-year graduate course. In the spring semester 1973

we decided that a separate three unit (one semester hour credit) laboratory grade for the computer projects (in addition to the current nine unit course grade) was more appropriate. We will explain this more later.

The course consisted of the following topics:

a. an algorithmic approach to the notion of function

b. the nature of round-off and truncation errors (and how they are handled), propagated error, inverse error, computational instability, considerations of global analysis versus local analysis (for example, in root finding)

c. some discussion on the timing for modifying techniques (for example, singularities in quadrature), on ill-conditioned situations (small differences of large numbers, multiplicities, etc.)

d. the development of an appreciation of the possible interactions between analytical and computational approaches

e. introduction to useful algorithms concerning solution of nonlinear equations, interpolating polynomials, numerical differentiation and integration, numerical solution of ordinary differential equations, solution of systems of equations, and solution of boundary value problems.

We have been teaching this particular course for the past two and one half years. When it was initially taught there was no directly assigned computer work requirement. The material for this paper is based on these experiences. Some of the material for this paper was obtained by an anonymous questionnaire given to the students at the end of the semester and from individual interviews with the students during the term.

The aim of the course then was twofold: to foster the

ea of intercommunication and group effort toward a common

goal, and to provide conceptual foundations and computational skills which would be widely applicable. We will now look at the first aspect of the course, namely the group effort.

The groups were formed around the students with the most computer experience who were designated group leaders. The role of the group leader was to perceive how each project could be most efficiently divided among the group members. He was to be responsible for assigning the role to be played by each member and for guiding their efforts toward a smooth execution of the problem. It was hoped that in this manner even the most inexperienced computer programmer would rapidly gain expertise through the interchange of information and ideas in a group setting. It seems apparent that the art of computer programming can most rapidly be learned by working with a skillful programmer.

The five projects varied in complexity. The simple projects covered a broad enough area in order to be easily broken down into individual parts. The more complex problems were left to the ingenuity of the group leader for the assignment of individual parts. It was hoped that, in this manner, we would encourage more group effort.

Although in the fall semester each report was done by a separate member of the group, we tried to encourage the groups to actively participate with ideas for the analysis of the output. Much class time was spent discussing the value of certain techniques, how to improve results (by improving techniques to reduce errors and by reducing running time) and how to make comparisons of the techniques through analytical considerations.

Consequently no set rules for the reports were given other than to relate as much as possible the theory of the algorithm or algorithms under consideration with the actual computer output. Some guidelines were given however for comparing results when several algorithms were being considered. We were interested in seeing if the ingenuity of the group (as a whole), in conjunction with the emphases from the lectures, would result in some good analytical reports.

The results concerning the first semester work have been described in [1]. At the present time I would like to summarize what occurred there in the first semester. Of the eight groups formed, two functioned within the framework of the ideal and two were less so but still acceptable. The others were quite incohesive.

Some recommendations to improve the cohesiveness of the groups were made in [1]. The projects were based on problems from the text [2]. In the beginning the projects were made simple to encourage student efforts. In such cases, it was not imperative from the problem that the groups had to meet. Because of the simplicity, one would have expected decent results. For some groups, such was not the case. The errors went from programming problems to lack of understanding. However it is now clear that all projects should be made sufficiently complex to insure the students seeking out advice from the group or from the instructors.

Another problem occurred when the group leader, in some cases, did not assume the lead for the group. Even though these particular leaders had more technical knowledge and expertise than the other group members (as was the case with all group leaders) some would not assign the other members to particular parts of the project. However this problem found its own solution for these groups, as they determined their own leader who then organized the group effort.

For the first semester, grading for the course was obtained from the consideration of the cumulative grade obtained on three 75-minute exams, the grade obtained on one project report, and the cumulative grade obtained from other submitted assignments. The first two were averaged together and the last was used to determine borderline cases.

We had hoped that the students in a group would not only collaborate on the programming of the projects but that they would help the member who was writing the report. Each member of the group had to write a report discussing the results. Two weeks, on the average, was given for pursuing the project. Each person in the course then received a grade based on the quality of his report. There were also grades assigned to the group as a whole based on the accuracy of the output and the "quality" of the code for each project. Obviously each student had to realize that the more important grade for himself was the individual report grade.

It appears that one cannot do too much with this common grade. It was felt necessary to have such a grade to encourage the group effort. However it appears that it did not achieve this in some cases. but without it, it seems clear that less group success would have been obtained. In particular, some students would not work hard (if at all) on the programming of the projects and would concentrate on the report of the one project for which they were responsible. During the first semester some students told us that they were entirely on their own when it came to writing the report.

In [1], our recommendation was to pay less attention to the problems associated with programming. Given that the student has had the necessary prerequisite, we would prefer to now provide him with a package of routines to handle the problems. A semester course runs too rapidly to spend much time worrying about programming problems. It is more important here to carry out the necessary numerical techniques and analysis. The text [2] has some routines given (about two per chapter). However more (see [3]) should be provided. As a consequence, additional projects may be given in the time saved. For it is here, with the projects, that the student receives the most technical benefit. We will say more about this later.

Recapitulating we feel that group effort can and should be induced by more substantial projects especially early in the semester (to get the group familiar with collective effort), by a change in our grading of projects (possibly ask each group write five collective reports), and by reducing the size of

a group to three members (since this is the size in which the groups frequently worked). We have found that five of the above "mixed-types" of students will not generally work together cohesively, in an organized way. In each group someone should be designated who has the most computing experience and the group should be left to determine its leader who is responsible for computer time, final form of the report, etc.

I would like to now summarize our opinions based on the developments occurring when this course was given in the spring semester 1973 at Carnegie-Mellon University.

The class consisted of 34 students including seniors, juniors and sophomores. The majors were in engineering, mathematics, physics and chemistry. For the projects, seven groups were formed, for the most part again consisting of five students each. The material covered was somewhat more exhaustive than the previous semester. It included all the material listed under topics (a), (b), (c), (d) and (e), that is essentially the first eight chapters of [2], plus additional material in the way of introduction to: optimization, dynamic programming, linear programming and some methods for partial differential equations.

We do now feel, strongly, that the linear algebra (as a course) pre-requisite mentioned previously is not essentially needed but is still desired from the over-all point of view. However the course may be given satisfactorily with the pre-requisites of our first computer programming course (our 15.100) and Calculus II (our 21.122 whose material is also covered in Physics-Calculus joint course). For the level that the course

has attained (that is, to essentially appeal to sophomore students), we feel the course requirement of linear algebra is too much. However many students taking the Numerical Methods course have had a linear algebra equivalent. We have found (though), that all that material is not needed. What is not known can quickly be given to the students in one session and through the everyday classroom exercises.

Some mention should be made of the contents of the present prerequisites. The 15.100 course includes:

> The concepts of algorithms and programming. Planning the solution of problems with organizing techniques such as flow charts. The use of a general purpose programming language to solve problems. Basic concepts and logical design of the general-purpose digital computer. Selected problems involving both numerical and symbolic computations. Laboratory sessions involve preparing and solving problems using a digital computer.

There are two one hour recitations and one. one hour lecture.

The 21.122 course includes:

> Differential equations, vector functions, functions of several variables, partial derivatives, directional derivatives, multiple integrals, and Taylor's series.

There are either three one hour recitations or two one hour lectures and two one hour recitations depending on the class of students.

It does appear then that this material is sufficient to attain the required facility in the course. We have given the description of the computer course, to emphasize its strength when we make the next proposal.

The necessity to have student run problems on the computer is clear. What is needed is to acquire the facility in attacking problems, visualizing a solution, judging the proper techniques to be used, making the proper estimates and performing the proper error analysis. It is clear that it is here where the student's main efforts should lie. Unfortunately more time than what was desired, was spent on the basic programming. One reason for choosing [2] as the classroom text, was the availability of programs. However after the second semester our feeling is even stronger that all computer-run problems given to the students in class should be run from canned computer programs. Better use probably could be made of the graduate assistant by his supplying these programs. The students could spend their time discussing the several aspects mentioned above. Initially much time would have to be spent organizing these programs so that students would have some selection of methods available. The basic knowledge acquired in the 15.100 course should be sufficient to carry out any debugging or minor modifications to permit a particular program to fit a particular problem. There are also many routine type numerical mathematical programming problems from which to choose in the current edition of the Schaum's Outline Series in Numerical Analysis. The student should be encouraged to go there directly.

With the implementation of this proposal, we can then request more project-type problems from the students to be performed in the small group environment. As evidenced by the student questionnaire (another was distributed in April, 1973 to this class) and

by the results of the individual projects, it is clear that this is where the real learning is obtained. The students overwhelmingly agreed that additional units should be given for this portion of the course and that it should remain as an integral part of the course. Consequently we have now established the computer laboratory section of the course at 3 course units (one credit hour) and the lecture section at 9 course units (three credit hours). This better reflects the amount of time utilized in the various portions of the students' activities.

As we have explained previously the projects were performed in groups of five students each with one student selected, on the basis of his experience, as the group leader. However this semester we operated these groups in a slightly different way, which demonstrably developed as an improvement over last semester's operation.

Joint efforts were required in writing all of the reports due on the projects. The students were asked to meet with their groups at their leisure and to decide upon the division of the work. The group leader was to assist with the overall control and because of his greater experience resolve, if possible, technical problems that occurred. These may have been in the programming, in the detailing of the written discussion, in the assigning of tasks, etc. The actual writing was to take place by one individual on a rotating basis. Each student was to have an opportunity to collect the work of the individuals, synthesize and analyze it, and then present it in a suitable written format.

This semester the graduate assistant was requested to devote more of his portion of the class period in discussing the individual written reports. He utilized reports retained from the previous semester to demonstrate the desired format. He also discussed, in more detail, each of the current projects so as to demonstrate what ideas needed to be emphasized. This resulted in a better average quality of reports obtained. In the previous semester where this control was not maintained, the written reports ranged from poor (two) to very good (two) with four as average. This semester all reports were at least average (no poor ones). As would be expected this level was obtained immediately and maintained.

One group lost their group leader (through his withdrawal from the course). This group had one bright young lady with a poor preparation and three other participants who were more or less disinterested. It is interesting to note that this group was "carried through" its various projects by the incentive and outright assistance of this one young lady.

Of the seven groups formed, the performance was very good. A joint grade was given to each project. That is, the grade on the written report was shared by the five members. The average grade on these reports was then computed as one-third of the final grade. There is some objection to having such a "joint grade". The objections come about especially when some students felt that their individual participation was more than that demonrated by the grade. On the other hand, the benefit comes

from the encouragement obtained through strong collaborative effort. Individual student sentiment pertaining to "not being taken for granted", eliminated rather successfully, one individual performing more than an equal share of work. From personal interviewing and from the questionnaire, it appears that the present system yielded active participation by all members.

Four exams were held during the semester. In our semester system, this appears as a good number. The average grade of these exams formed a two-thirds part of the final grade. There was no comprehensive final. The individual exams were qualitative in nature. Some minor calculations were necessary. The exams were geared to emphasize material from the class and to demonstrate what ideas had been learned from the individual projects and the individual homework assignments.

The homework assignments were of a quick calculational nature. Many of them could be done on a desk calculator if they had been available. However since such were not available, the students either performed them by hand or they ran quick computations on the computer. It was desirable to have the latter additional experience.

We do now envision a full year's course in Numerical Mathematics run in essentially in the "group collaborative effort" format. It would be desirable to spend much of the time of the first semester obtaining a thorough but transparent approach to iteration, interpolation numerical integration, systems of equations and solutions of ordinary differential ations. The second semester can then be well spent on

elliptic, parabolic and hyperbolic partial differential equations, numerical double interpolation, multiple integration, curve fitting and, approximation of functions. Due to the good response to the above course, the students have been demanding this second level course. The author did offer such a course in the Spring 1972 to a group of students who had taken the above course in the previous fall. It was run on a seminar basis.

REFERENCES

[1] Alo, Richard A. and R. W. Smith, "Student group efforts in studying numerical analysis", Fourth Conference on Computers in Undergraduate Curricula. Claremont, 1973.

[2] Gerald. C. F., Applied Numerical Analysis, Addison-Wesley Reading, Massachusetts, 1970.

[3] Haggerty, G. B., Elementary Numerical Analysis with Programming, Allyn and Bacon, Boston, 1972.

[4] Henrici, P., Elements of Numerical Analysis, John Wiley and Sons, New York, 1964.

CARNEGIE-MELLON UNIVERSITY
Pittsburgh, Pa. 15213

elb/5/29/73