added for, say, honours students).

(D) These proposals are at least worthy of consideration, for Professor Ralston has wide experience in both computer science and mathematics and backs up his suggestions with an exhaustive study.

More questions are asked here than are answered. For example, consideration needs to be given to the feasibility of such topics for various types of student, ranging from students of management through to honours mathematics students. But space permits no more comment, and for answers to such questions the reader must either consult [4] and [5] or, if Ralston [5] page 484 is correct, undertake experiment for himself or herself.

Educational problems are not usually very well defined; they are likely to be controversial and to raise temperatures. Indeed it may be that Ralston's criticism does not apply here and that all is well. If not, and this article creates some discussion or starts people thinking about the problems raised here, then it will have achieved its purpose. We hardly need reminding in 1983 that computer science is a major undergraduate subject. But what has perhaps not been widely recognised yet is the fact that the next generation of students will be taught computer science in secondary schools by those currently studying it at third-level. Future incoming students may therefore elect to study computer science "because it is familiar" just as many do now, I suspect, in the case of mathematics.

References

[1]. E.W. Dijkstra, Programming as a Discipline of Mathematical Nature, Amer. Math. Monthly, 81 (1974), 608-612.

[2]. S.P. Gordon, A Discrete Approach to Computer Oriented Calculus, Amer. Math. Monthly, 86 (1979), 386-391.

Irish Mathematical Society Newsletter, March 1983

[3]. D.E. Knuth, Computer Science and its Relation to Mathematics, Amer. Math. Monthly, 81 (1974), 323-343.

[4]. A. Ralston, Computer Science, Mathematics and the Undergraduate Curricula in Both, Technical Report 161, Department of Computer Science, SUNY at Buffalo, 1980.

[5]. ———— Computer Science, Mathematics and the Undergraduate Curricula in Both, Amer. Math. Monthly, 88 (1981), 472-485.

Department of Mathematics,
University College,
Cork.

USING MICROCOMPUTERS IN UNIVERSITY MATHEMATICS TEACHING

A.W. Wickstead

Until the last few years, most attempts to involve computers in the teaching process involved sitting a student at the terminal of a large computer and attempting to perform the whole teaching process by presenting information which was then tested. Apart from being very limited in what could be taught in this way, such projects have tended to be very expensive. They involve relatively large computers, many terminals to them and a lot of (expensive) programming effort. All of this expenditure has to be incurred before the technique can be tried, so its use has, of course, been very limited.

The advent of cheap microcomputers in recent years has enabled the less well financed University Mathematics department to acquire a computer and, experiment with its use in teaching. With such limited resources (and no programming assistance) it would be foolish to hope to emulate the large

projects that have been attempted in the past. There are however uses that can be made of a single microcomputer in front of a class, especially if reasonably good quality graphics is available. All our uses of microcomputers in teaching Pure Mathematics at Q.U.B., since we started three years ago, have been of this nature, although other departments make microcomputers available to students for numerical work in statistics, numerical analysis etc. Our approach was easy to introduce as it fitted easily into our existing, fairly traditional, teaching methods. This meant that the technique could be tried by one individual without needing any departmental policy decision.

Typically I have used a microcomputer to introduce a topic before teaching it in the normal way. The idea has been to augment the normal exhibiting of a few examples to illustrate a definition, such as that of uniform convergence. The computer can draw graphs more accurately and quicker than I can, so that many more examples can be provided to drive home the idea of uniform convergence before trying to work with it. In order that students have a record of the session they must either be given time to copy what they see or printed copies of the graphical output of the computer. The temptation to proceed too fast is even worse here than with pre-prepared transparencies! One of the most useful features of using a microcomputer for illustrative purposes, rather than many drawings prepared in advance, is that students can suggest examples that they would like to see and then have them drawn immediately. I have found students far more willing to make suggestions in such a context than they are in any normal teaching session and that the increased interaction seems to persist beyond the actual session (at least for a few days).

Topics that we have approached in this way include uniform convergence, Riemann integration, continuity, differentiation, Turing machines, approximations of the Binomial distribution and a naive approach to Fourier series. We hope

to add to this list in the next few months. The bias towards Analysis in this list of topics reflects the teaching commitments of myself and others interested in using the computer rather than any inherent limitation in the technique. We have had most success with students who might have difficulty even understanding the definition of a concept (more than might be supposed). These can progress further and more quickly than might otherwise be the case. The good students gain little (this reflects the results of a survey done with schoolchildren recently) but it does mean that we are helping those who need it most. Opportunities for using microcomputers seem to lessen as the level of courses increases, although there are exceptions.

All our teaching has been on an ITT2020 (virtually identical to an APPLE II except for improved graphics). This works at a reasonable speed and can provide tolerably good black and white graphics (but not such good colour). We have found it desirable to improve its speed by adding an Arithmetic Processor Card, and (at the time of writing) have just taken delivery of improved graphics for it. We have not found much use for colour, but some programs certainly benefit from it. For example, one of our programs compares the graph of a continuous function with that of a polynomial approximation to it. Once the approximation is reasonably good, colour is essential to distinguish the two graphs. It is not realistic to load a sizeable program from cassette tape in front of a class, so disk storage (or something similar) is vital. We have also found it useful to have a printer available. This has greatly aided program development and is also useful for providing a printed record for the class of some of the computer's output. In some cases, such as simulation of the action of a Turing machine, it would not be possible for students to make a copy as they watched.

From our experience, not only in teaching at University level using a microcomputer but also from training teachers to use them, we have isolated several critical features of

programs that are to be useful. First and foremost, the program should teach rather than do. There are many programs that multiply two matrices together, but very few that try to teach how it is done. A good program is as flexible as possible. This allows the maximum interaction with a class and also allows more people to fit it into their own teaching style. Often it is little more work to write a program that will handle many examples rather than just one, although that is not always the case. If many people are to use a program it should be as easy to use as possible. No-one is going to read a twenty-page manual before using a program for half an hour, yet that is what some programmers expect! Choices that need to be made should be clearly stated and a relatively small number presented at once. Far more could be done to guide people through a program than is being done. For instance, there is little point in offering the choice of saving a graph onto a disk file to a person who has not yet drawn a graph! By such restrictions it is possible to lead a person gently into a program. The final feature of a program that we have found desirable is frequent opportunities to save its output onto disk files. If this has been done in advance of a class they can be printed for distribution to a class or many of them can be displayed to a class in far less time than it took to produce them. Before we obtained our Arithmetic Processor Card we frequently found it necessary to do this in order to present the number of examples that we wanted in the time available.

There is, unfortunately, no source of programs for teaching mathematics at the University level. About eighteen months ago I contacted about twenty Universities, Polytechnics and Teacher Training Colleges in the U.K., that I knew to have an interest in the field, with a view to setting up a program exchange. The response was negligible. What has been done elsewhere tends to be on larger machines or on exotic hybrid systems. Anyone setting up such a system should be reconciled to "going it alone" on program development as any specific combination will be rare. Even though little is available now, it is worth bearing the possibility of exchanging software in mind when selecting a system and when programming for it. For instance, all our programs can be modified in a few minutes to run on a standard APPLE II, yet we can also use our extra features if they are available. If Universities in the Republic of Ireland were to standardise on a microcomputer and exchange software for it, they would soon find themselves in a far more favourable position in this respect than the U.K. Universities.

Our programs require a 48 K APPLE II with a language card and at least one disk drive. Anyone seeking copies is welcome to write to me for further details. The programs are all written in APPLE PASCAL 1.1 and I have described some of the programming difficulties in "Interactive Mathematical Programming", in COMPUTER AGE 14 (January 1981) 15-17.

Department of Pure Mathematics,
Queen's University,
Belfast BT7 1NN.