A Mathematica-*based CAL matrix-theory tutor for scientists and engineers*

M.A. Kelmanson, S.B. Maunders and S.Y. Cheng

Department of Applied Mathematical Studies, University of Leeds, Leeds LS2 9JT

Abstract

Under the TLTP initiative, the Mathematics Departments at Imperial College and Leeds University are jointly developing a CAL method directed at supplementing the level of mathematics of students entering science and engineering courses from diverse A-level (or equivalent) backgrounds. The aim of the joint project is to maintain – even increase – the number of students enrolling on such first-year courses without lowering the courses' existing mathematical standards.

A CAL tutor for matrix theory is presented in this paper, in the form of Mathematica Notebooks. This constitutes one of a list of specific A-level mathematics core options required by science and engineering departments. The module has been written so as to recognize students' errors and advise accordingly. Questions are generated randomly, at run time, in order to preclude copying between users. The module incorporates automated performance indicators so as to impinge minimally on existing staff resources. As an aid to other CAL authors considering the use of Mathematica Notebooks, idiosyncratic difficulties encountered within Mathematica Notebooks are catalogued and discussed in detail.

Introduction

The problem

Students entering science and engineering courses are frequently faced with the problem of weakness in their mathematical preparation for further education; this is due to a number of factors, not least of which is the vast number of possible combinations of A-level mathematics syllabi. Many university science and engineering departments continue to offer courses requiring a 'reasonable' level of mathematical comprehension, so that students deficient in specific A-level mathematics topics are often inadequately prepared for such courses.

The problem, as described, affects a number of first-year courses at universities throughout Great Britain; it has, in recent years, been further exacerbated by the introduction of the GCSE, since the subsequent jump to A-level mathematics is often considered to be too large. Moreover, many engineering departments accept students with a very wide range of entry qualifications besides A-levels, BTEC certificates and diplomas being the most notable examples. Unfortunately, the mathematical content of such syllabi is much lower than that of A-level. The impending reorganization of the many A-level mathematics syllabi will lead to an inevitable dilution in many areas – especially calculus – and will further aggravate the situation currently experienced by many university science and engineering departments.

In later academic years, deficiencies in mathematical grounding affect almost all the theoretical courses taught in science and engineering departments, which are noticeably reluctant to defer their inherent course material in favour of the necessary extra mathematics material. This recalcitrance towards course postponement is wholly understandable in the light of the inevitable knock-on effect which would both require an increase in the overall course duration, and affect the course validation with professional institutions. The worst possible scenario would be that a serious hiatus in degree standards between British and European universities might occur.

In the light of the above information, there is clearly a need to establish a convenient and flexible mechanism for teaching this *transitional* mathematics material in order to bring first-year students up to the required standard. (*Transitional* is used here in preference to *remedial* since the latter is definitively associated with slow-learning: in the context of this paper, this is clearly not the case.) Finally, any such mechanism should ideally impinge only minimally on existing staff resources.

The suggested remedy

In late 1992, Imperial College (IC) and Leeds University (LU) were jointly successful in procuring a bid, under the TLTP Initiative of the UFC (now the HEFCE), which described a *Mathematica*-based system for presenting mathematics tutorials to scientists and engineers. The suggested remedy for the problems described above was the production of a suite of CAL interactive tutorial modules based on *Mathematica* and its inherent front-end, *Notebooks*, which allows a certain degree of control over the navigation through the tutorial material.

Mathematica (Wolfram, 1991) was suggested since, as a sophisticated and versatile integrated package incorporating programming, numeric and symbolic computation, text processing and graphics, it would allow flexible production of an extremely wide range of detailed mathematical material. Moreover, *Mathematica* has in recent years enjoyed a high profile in numerous teaching projects; see, for example, Brown, 1991; Gray, 1991; Burbulla, 1992; *The Calculus Reader*, 1993.

Further to the above, *Mathematica* could be launched from a variety of platforms and operating systems, making it portable both within and between sites, primarily for development, ultimately for nation-wide distribution. Finally, through the interactive medium of *Mathematica Notebooks*, it was envisaged that a certain degree of genuine interaction between student and CAL module might be achieved, allowing students to use the same module on different levels solving different problems. More detailed discussions concerning *Mathematica* are deferred until later.

Since the project began, on 1st November 1992, all necessary module development staff – three at each site – have been appointed, and all required hardware and software is in situ. In the (ongoing) initial phase of the project, IC are responsible for modules in complex numbers, trigonometry and vectors, while LU are producing modules in differential calculus, integral calculus and first-order differential equations; both sites are producing separate modules on matrices, for comparison. It is LU's matrix module which will be presented and discussed, from both pedagogical and computational points of view, in subsequent sections of this paper.

Module design

Interaction philosophy

A criticism which might be made of existing CAL modules is that much of their interaction is 'hard wired' into them, so that after a few attempts at the same module, a student might perhaps achieve correct responses by memorizing problems that he/she has already tackled, rather than by actually understanding the principles underlying the module's material. A natural extension of this is to question wherein lies the difference between such a module and a textbook. In the light of that question, the ensuing criticism would be that a textbook would, in fact, rank more useful than such a CAL module, since the former is truly portable. Therefore, an essential feature of our approach was to ensure that our modules were somehow more 'human' in their interaction with the student: this constitutes a considerable challenge from both programming and pedagogical points of view. With regard to this aspect of a quasi-human helpful approach, the module should further be able not only to point out errors, but also to recognize, trap, correct and advise accordingly in the light of student difficulties. If errors are merely indicated as either right/wrong, our experience dictates that user-frustration will ensue, at which point the module ceases to be effective.

Interaction implementation

One of the authors of this paper had already written a CAL Fortran 77 course given to first-year mathematics students at Bradford University. In that course, students were initially shielded from the predictably confusing side-issues of compilers and loading, since these are, in effect, system-dependent. By the same token, the LU team decided that module users should be entirely shielded from the *Mathematica* 'maths engine' in the present project. We felt that the students should certainly not be expected to assimilate the language of *Mathematica*, since it might detract from their concentrating on the issue at hand, to wit: the transitional mathematical material. This was not as easy to achieve as one might think, since the only (partially suitable) interactive medium available to us (on which the initial project proposal was based) was *Notebooks*, *Mathematica*'s inherent front end. Moreover, it was envisaged that the modules were to be developed for a range of platforms, but since the majority of UK universities and sixth-forms are more likely to be IBM-PC based, all further discussion will be thus directed. We are also developing *Mathematica* software for Unix machines, but these presently do not support *Notebooks*.

Mathematica Notebooks

For the purposes of subsequent discussion, it is necessary briefly to outline the module-authoring medium, *Mathematica Notebooks*. This is a Windows-based menu-driven front end through which the user may interact with the *Mathematica* kernel to produce mathematical and graphical output. *Notebooks* is presently limited to the PC, NeXT and

Macintosh platforms, the last of which has been used extensively for Project CALC (*The Calculus Reader*, 1993), which has been used (in the USA) to teach first-year university calculus in a laboratory environment.

Mathematica Notebooks constitute an integrated environment in which the user can be prompted to perform calculations and/or exercises. Information appears in *Notebooks* in cells, which are of differing types: input, output, text, graphics, etc. Hence, from a pedagogical point of view, *Notebooks* enjoy the property that they are able to unite demonstrative text with supporting graphics. Consecutive cells pertaining to a particular area or problem may be *grouped*, in which case only the first cell in the group is subsequently visible, the rest being *closed*; the closed cells may be *opened* again by double-clicking on designated *group-boxes* which appear on the screen. Similarly, groups of cells may themselves be grouped and closed; in this way, a potentially large document, although in the *Notebooks* in a linear form, may be 'condensed' so that it can be viewed rather like the contents page of a book. The user may then choose a particular topic title from within the *Notebooks* titles by double-clicking on its group-box, which will then open and display either subtitles or text. Hence it is easy to see that notes, in the form of text, constitute the most-nested level of information, with section banners residing at the outermost level of cell nesting. Figures 1, 2 and 3 illustrate some of the points raised so far.

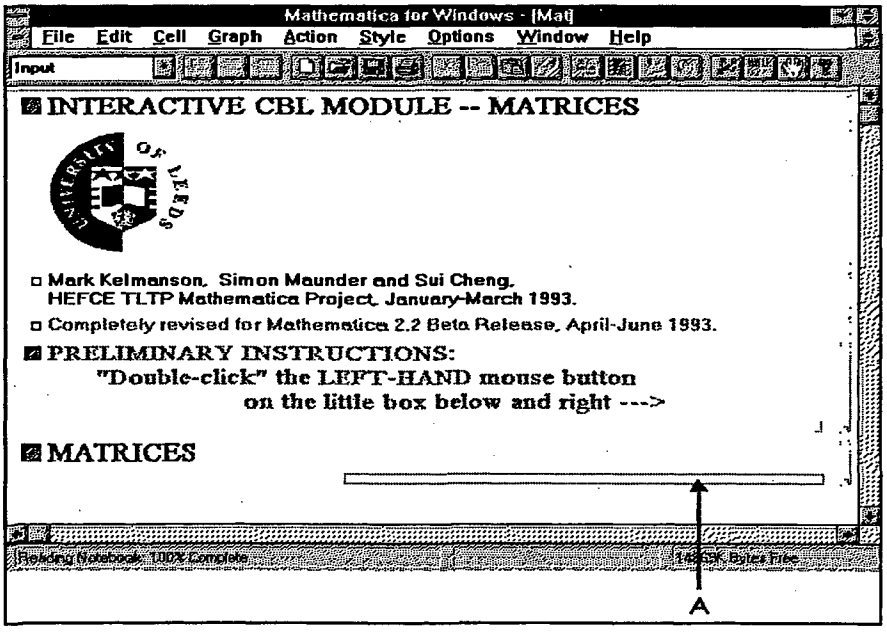

Figure 1: The Matrix Notebook as it first appears

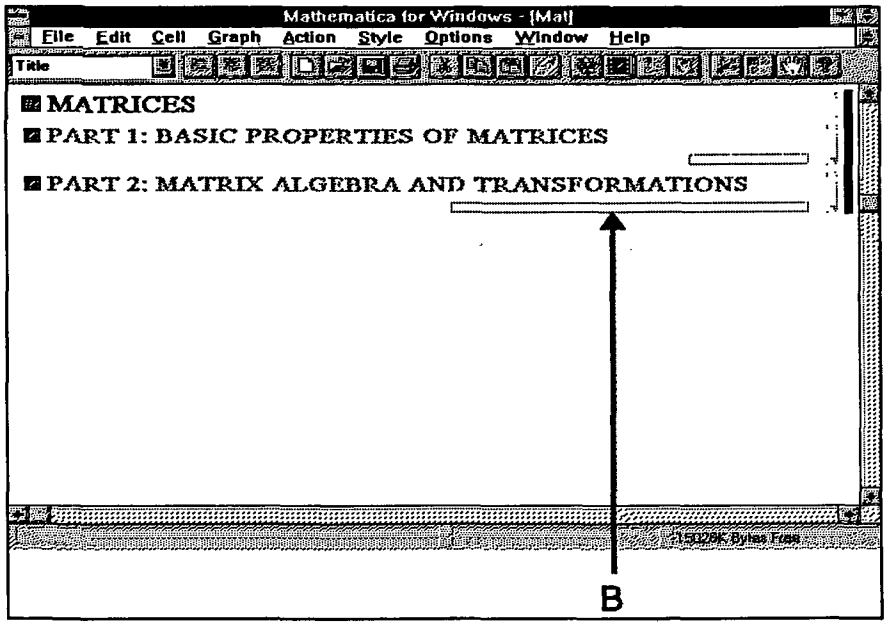

Figure 2: The Matrix Notebook after double-clicking on the cell marked 'A' in Figure 1

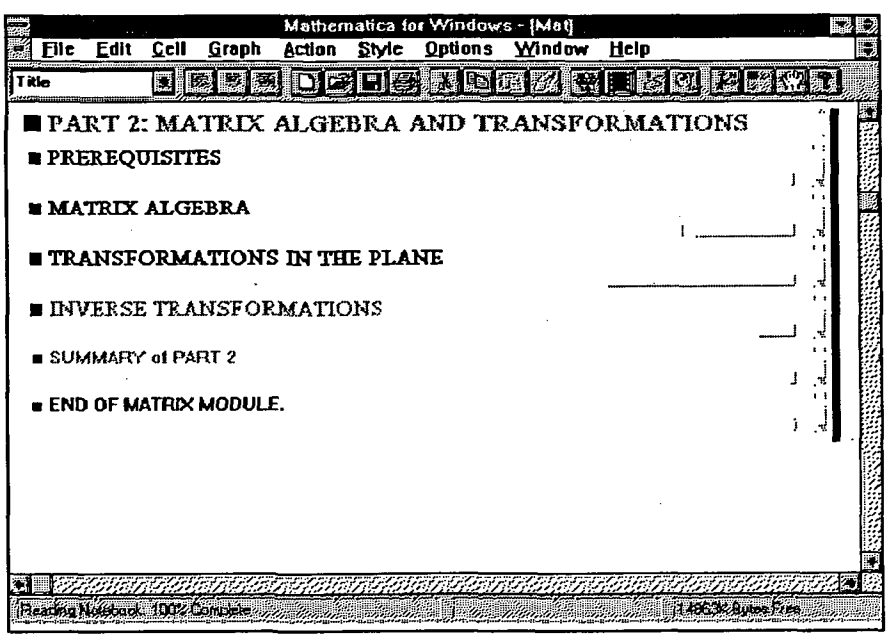

Figure 3: The Matrix Notebook after double-clicking on the cell marked 'B' in Figure 2

Difficulties within Notebooks

In producing the Matrix module, we encountered the occasional difficulty within the *Notebooks* environment. For example, once opened, groups remain thus even though the material contained therein may no longer be required. Thus the *Notebooks* become more linear in appearance as progress is made through the module unless the user explicitly closes the 'finished' cells. Similarly, when the user is interacting with *Mathematica* via its *dialogue box*, the autoscroll in the *Notebooks* main window is disabled, thus forcing users explicitly to search for the responses to their input. In version 2.2 of *Mathematica*, there is no control over either the size or location of the dialogue box; this, coupled with the aforementioned loss of autoscroll, constitutes a non-trivial problem in terms of ease of navigation by uninitiated users. Finally, the absence of mathematical text and the inability to change fonts *within a cell* proved somewhat restrictive; much effort is required to circumvent this rather artificial restraint.

Mathematica's absence of hotwords or 'intuition-driven' buttons etc. enforce a certain degree of linearity on the style of the module. We hope that future versions of *Mathematica* will accommodate the necessary features to support a more tree-structured module design: if they do not, then *Toolbook* may yet be employed – upon arrival of *MathLink* – to front-end the interaction, which is a further reason supporting the shielding of students from *Mathematica*'s programming language.

It is important to remark that the LU team has enjoyed an excellent rapport with Wolfram Research and their in-house Technical Support Group who, in the light of our findings, were quick to remedy anomalies in the 2.2 β version of *Mathematica*. We are confident that our aforementioned comments will also be considered seriously, and that, through interaction with CAL groups such as ours, Wolfram Research will undertake relevant developments in future versions of *Mathematica*.

The Matrix Module

Content

In order to facilitate navigation through the module, it was 'naturally' split into two sections (see Figure 2): *Basic Properties of Matrices*, for complete beginners, and *Matrix Algebra and Transformations* for users possessing a basic knowledge of principles and notation. Each such section was further subdivided into more detailed sections (see Figure 3). Note that the (horizontal) size of any group-box in Figures 1 to 9 is indicative of the amount of material contained within that group. Each general title contained a combination of tutorial notes, supplementary exercises and, where relevant, animated graphics. For illustrative purposes, we shall concentrate further on the section entitled *Transformations in the Plane*; a typical section of text is depicted in Figure 4.

Interaction and questions

In the absence of hotwords and buttons in the *Notebooks* environment, interaction had to be 'invited', as shown in Figure 4. This is far from ideal, as it essentially means that users must be aware of how to make *Mathematica* recognize their input; while such information can be placed in a supposedly prominent location on the screen, there is no guarantee that the user will assimilate it. We would prefer to see a more intuition-based front-end for this purpose.

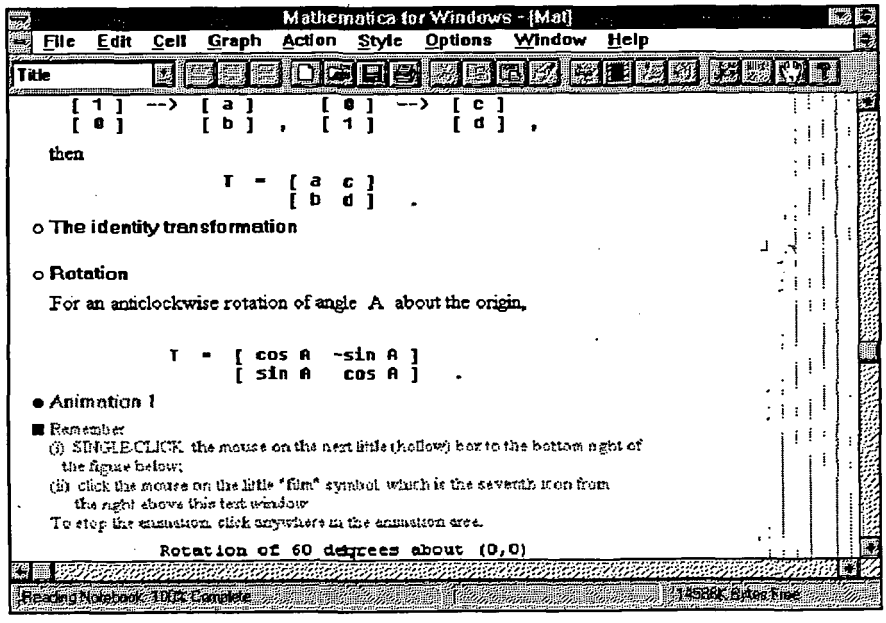

Figure 4: Textual content and 'invitation' into interaction

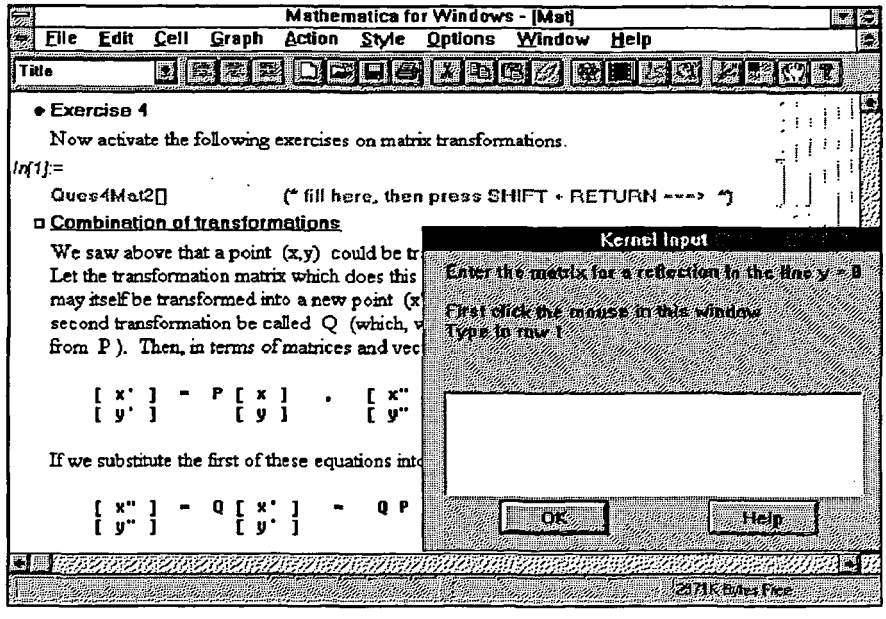

Figure 5: The dialogue box with associated question text

On initiating a question, a dialogue box appears as in Figure 5. Unfortunately, the tutor's dialogue is restricted to only two or three lines of text; couple this with the absence of mathematical fonts and the wish to shield users from *Mathematica*'s language, and it soon becomes apparent that a great deal of programming effort is required to obtain seemingly straightforward information.

As a means of discouraging the memorizing of previously-attempted examples, random numbers are used in all the interactive exercises. Students are encouraged to perform calculations using pen and paper and, in the later sections of the module, open-ended examples are proffered (just as they might be by a lecturer!).

In producing this (and other) modules, we sought to achieve an approach whereby the user was not merely told that their responses were 'right' or 'wrong'. In the former case, this is not too much of a problem, but in the latter, somewhat more is required if the CAL user is to gain a true understanding of the principles involved. In this respect, *Mathematica*'s inherent high-level functions proved invaluable in assisting us to locate various types of user-errors.

Error trapping

When prompting users for input, error-trapping code had to be written in order to circumvent the restrictions imposed by the pedagogically artificial *Mathematica* format. We stress that this effort is very much justified since it is common knowledge that once an 'automated' package fails to perform correctly, user-disenchantment and frustration quickly follow.

In order to allow the users to enter solutions in an intuitive manner – i.e. not tied to *Mathematica* input-format – considerable effort was expended in ensuring that the packages could recognize a large array of possible input formats. For example, our module accepts any of the following possible inputs for *sin*(–40): sin[40], sin(40), sin -40, sin-40, -sin40, – sin 40, Sin[40], Sin(40), Sin -40, – Sin 40, Sin-40, -Sin40. To interpret all of these (perfectly feasible) inputs as one and the same quantity, some sophisticated aspects of *Mathematica*'s programming language were employed. For the reasons described above, we do not consider that if an expression is mathematically correct but syntactically wrong with respect to *Mathematica*, users should be penalized (and therefore disenchanted) for their input style.

Fortunately, *Mathematica* supports an extremely powerful programming language which facilitates the implementation of interactive questions that are robust, syntactically flexible and corrective in nature. In the experience of the authors of this paper, *Mathematica*'s inherent language is more flexible than that of similar packages (see, for example, Char, 1991).

In general, the user is provided with two chances to enter the correct response, which is then parsed using a custom-written *Mathematica* package, to determine whether or not it is essentially correct. Minor errors are highlighted but not penalized. If possible, the specific nature of more serious errors is identified, and appropriate corrective action is then suggested; otherwise, more general information on the required approach is supplied. As an example, Figure 6 shows such a message in response to an incorrect user input.

This parsing and subsequent interpretation of user input is, in our view, a crucial element in the philosophy of CAL. Many CAL users – including the authors of this paper – are frequently impressed by (static or animated) examples involving sophisticated three-dimensional multi-coloured graphics (see, for example, Maeder, 1991)); to most users, these are the flagship of algebraic manipulators. However, from a pedagogical point of view such a

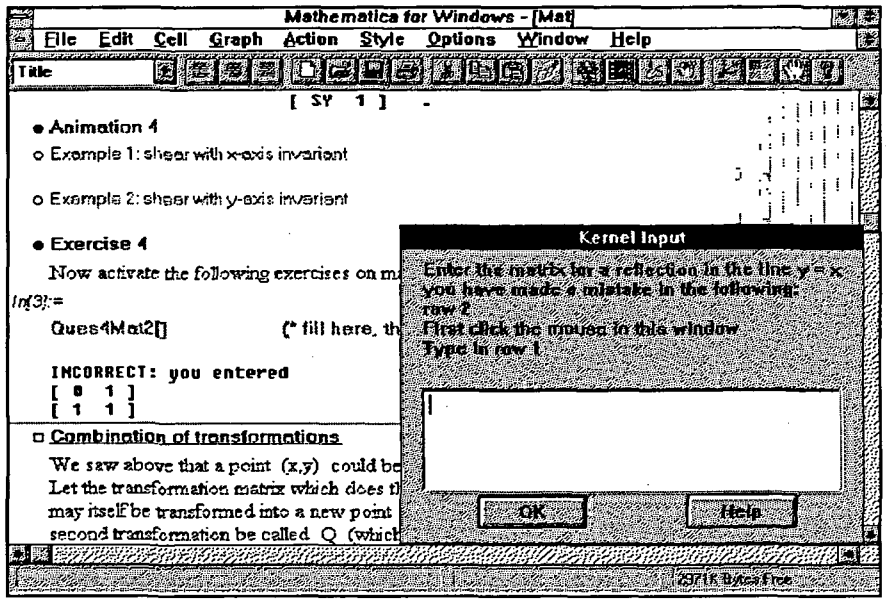

Figure 6: The dialogue box with error-correction prompt

product, visually impressive though it is, may prove far easier to achieve than the auto-simulation of truly sympathetic, discerning, corrective interaction. We believe that our approach goes some way towards achieving these aspects.

Animated graphics

An important feature of the *Notebooks* is the ability to incorporate animated graphics within the same environment as mathematical output and tutorial notes; it is this feature of *Mathematica Notebooks* which proved so attractive for the purposes of the present project. In *Notebooks*, an animation is obtained by grouping together a sequence of cells each of which contains a PostScript image produced by *Mathematica* graphics functions. If the group of cells is then closed in the manner described above in the section headed *Mathematica Notebooks*, only the first image in the sequence remains visible. An animation icon may then be used to initiate the animation of the cells so grouped. As an example, Figures 7 and 8 show animations illustrating the application of rotation matrix to a prescribed polygon: in this paper, the two images are necessarily shown 'in series' whereas in the module, the polygon flips cyclically from one position to the other with the axes remaining fixed.

The matrix module contains a substantial number of animations illustrating various types of transformation. Geometric interpretations, such as the general non-commutativity of matrix multiplication, are ideally illustrated via this medium. From an interactive point of view, the user may provide the parameters for a given transformation, thus permitting a degree of investigative experimentation; on this note, *Notebooks* may still be used to initiate open-ended investigations, as they constitute a powerful tool for the required analysis.

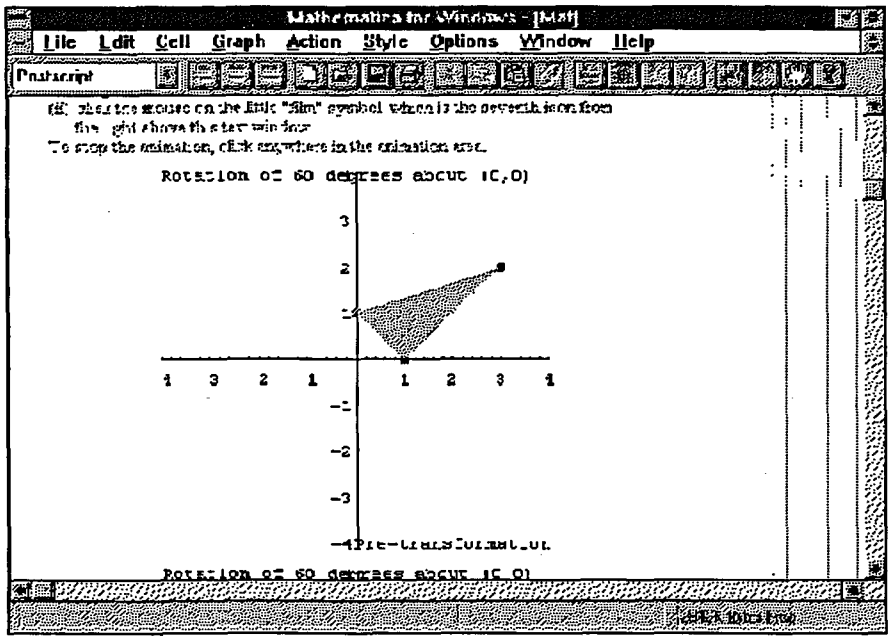

Figure 7: Animation cell showing pre-transformed polygon

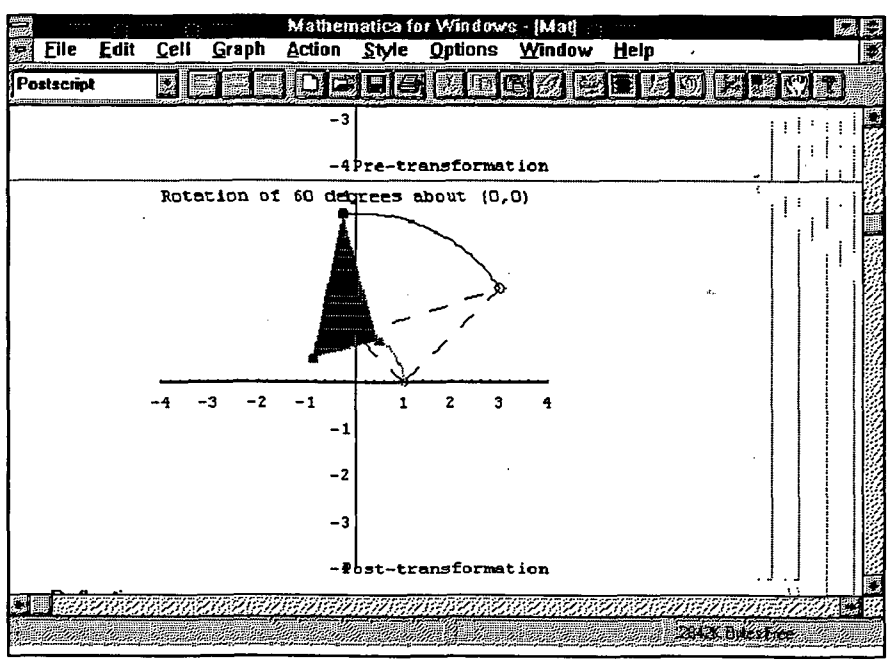

Figure 8: Animation cell showing post-transformed polygon

Automated performance indicators

An initial requirement of the project was that the modules produced therein should impinge minimally on existing staff resources. In order to ensure that student progress could be monitored, an automatic performance-recording procedure was implemented. Results of the user's attempts at the (or, indeed, any) module are automatically written to a file to which there is only staff access. For each user, only a single entry is maintained: this is updated in a cumulative sense insofar as it monitors attempts at a particular question from a number of different sessions. The user-record indicates both the percentage ratings on a given question and the number of attempts at a given question (this is to distinguish those users who, for example, score 1/1, from those who score 19/20).

In order to implement this automated logging of performance, an alphabetically-sorted list of entries is maintained. Any interactive question ends with an update of the appropriate entry. The update for a question may take one of two forms: amendment of existing information for that particular question, or insertion of performance information for that question into the relevant part of the user's entry (if that part has not previously been created). It should be noted that each entry is an alphabetically sorted list comprised of sub-entries for each attempted question. Since there exists only a single line in the file for each user, and furthermore that line consists of information for the attempted questions, the file is particularly concise. This file is encrypted and its contents accessed by only the (human) course tutor, using a custom-written, simple-to-use *Mathematica* package which (see Figure 9) displays graphically the required information.

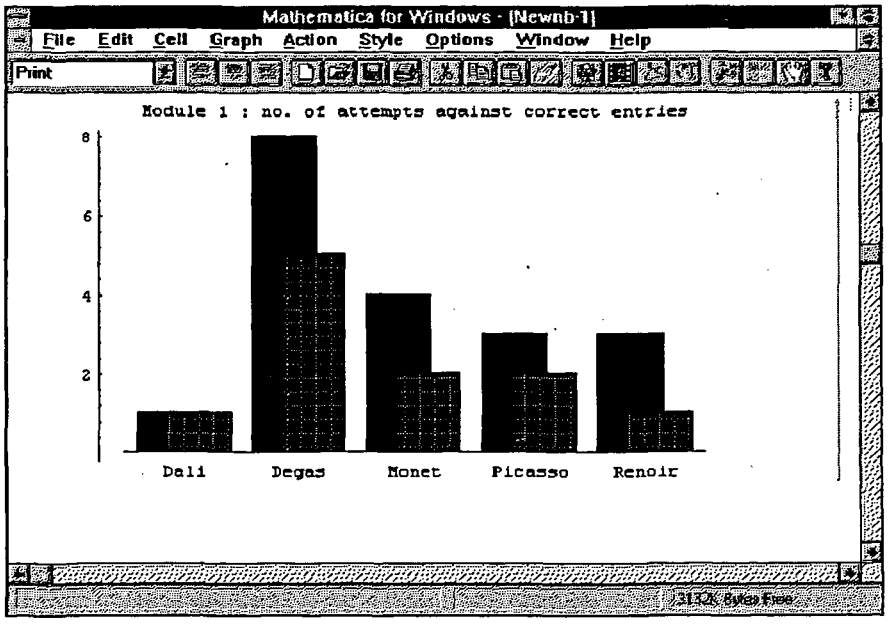

Figure 9: Tutor-viewable performance graphics

Evaluation

This paper would be incomplete without some indication of the performance of the module 'in the field'. At the end of May 1993, the module described above was tested by a group of sixth-formers from the Further Maths group at Harrogate Grammar School. None of the testers had used *Mathematica* previously, and about half were familiar with Windows. Nearly all had used some form of PC before.

Feedback on the module was extremely encouraging: 90-100% (of the testers) were positive about the content and presentation; 60-70% liked the style and mode of feedback; 60% found *Notebooks* clear and navigable. On this last point, the navigability is a factor which, in essence, lies in the hands of Wolfram Research. Overall, 20% of the users still preferred a text-book approach to that of CAL.

Many of the criticisms raised related to the inherent properties of the *Mathematica* front-end (briefly discussed above in the section headed Difficulties within Notebooks): position of dialogue box, loss of autoscrolling during interaction, absence of standard 'mathematical' notation, etc. These had already been predicted as they had essentially been the subject of correspondence between Wolfram Research and the authors of this paper. It is hoped that future versions of *Mathematica* will accommodate features which will alleviate or eradicate these particular problems.

Future developments

Further modules

The module described in this paper has essentially constituted a pilot investigation, the results of which can now be implemented in the other modules which will complete the first inter-site suite. As mentioned in our introduction, these are: vectors, trigonometry, complex numbers, differential calculus, integral calculus and first-order differential equations. In the light of current exterior interest in the joint project, both sites are at present clarifying copyright arrangements with a view to distributing freely the modules so far produced (any interested parties may contact any of the authors of this paper).

Alternative front ends

An essential difficulty of the *Notebooks* style is the inherent 'linearity'. Navigation through *Notebooks* is possible via user-search rather than by 'hotwords' or buttons; in our view, this is somewhat limiting. A separate, button-driven front-end, such as *Toolbook*, would seem more suitable for driving the *Mathematica* maths engine, but *Toolbook* also suffers from limitations in mathematical text presentation. One can circumvent this problem using equation editors and graphics packages, but that is far from ideal. In any case, not until *MathLink* becomes available from Wolfram Research can *Mathematica*'s kernel be easily linked to an alternative front end.

Acknowledgements

The authors of this paper wish to express their gratitude to the following people involved in the project: Professors M.I.G, Bloor and J. Brindley at Leeds University, the TLTP team at Imperial College, and Mr P. Heyes of Harrogate Grammar School.

References

Brown D., Porta, H. and Uhl, J. (1991), *Calculus and Mathematica: Part I*, Reading MA, Addison Wesley.

Burbulla, D.C.M. and Dodson, C.T.J (1992), *Self Tutor for Computer Calculus using Mathematica*, London, Prentice Hall.

The Calculus Reader (1993), Project CALC Manual, College Mathematics Department, D.C. Health Co., Lexington, USA, Autumn 1993.

Char, B.W. *et al.* (1991), *Maple V Library Reference Manual*, Berlin, Springer Verlag.

Gray, T.W. and Glynn, J. (1991), *Exploring Mathematics with Mathematica*, Reading MA, Addison Wesley.

Maeder, R. (1991), *Programming in Mathematica*, Reading MA, Addison Wesley.

Wolfram, S. (1991) *Mathematica: a System for Doing Mathematics by Computer*, 2nd edn., Reading MA, Addison Wesley.