AN INNOVATIVE LEARNING MODEL FOR COMPUTATION IN FIRST YEAR MATHEMATICS

E. J. TONKES§, B. I. LOCH†* and A. W. STACE‡

§ CS Energy, GPO Box 769, Brisbane, QLD, 4001

† Department of Mathematics and Computing, University of Southern Queensland, QLD, 4350

‡ Discipline of Mathematics, University of Queensland, QLD, 4072

*Corresponding author. Email: Birgit.Loch@usq.edu.au

Abstract

MATLAB is a sophisticated software tool for numerical analysis and visualisation. The University of Queensland has adopted Matlab as its official teaching package across large first year mathematics courses. In the past, the package has met severe resistance from students who have not appreciated their computational experience. Several main factors contribute: Firstly, the software is numerical rather than symbolic, providing a departure from the thinking patterns presented in lectures and tutorials. Secondly, many students cannot see a direct connection between the laboratory exercises and core course material from lectures. Thirdly, the students find hurdles to entry as commands often return annoying error messages and don't execute, and programs are difficult to write and debug. Overall, the details of the mathematics are lost in trying to negotiate the software. After considerable effort in tuning, it appears that a sequence of innovations has captured student support and added considerable value to both the computational and traditional learning process.

Keywords: Matlab; first year mathematics education; interactive workbook

AMS Subject Classification: 97U50, 97U70

1 Introduction

Many mathematics educators are aware of the challenge of integrating computational components into a classical first year mathematical curriculum. The diversity of student backgrounds, large class sizes, resource limitations and the fact that the curriculum is already full provides for difficulties. Even worse, teaching efforts can be rejected by students who perceive that the computational components are extraneous add-ons that are irrelevant to the core material. On top of this, the concept of 'computational components' is sufficiently broad in first year mathematics courses to induce debate amongst educators.

This paper reflects upon the efforts of the authors in developing, implementing and assessing a project to further integrate computational aspects into the first year syllabus using Matlab. The courses each have an intake of around 800 students per year (resulting in an annual total of 50 lab classes) and were revised over a three year timespan. Both service all disciplines, being the entry point for engineers, biological and physical sciences and mathematics students. The courses together cover a traditional first year mathematics curriculum: precalculus, calculus of one variable, calculus of many variables, ordinary differential equations and linear algebra.

It has been found [2,4] that student confidence towards mathematics can improve significantly when learning strategies incorporate a software tool. Cretchley et al [1] assessed the success of incorporating Matlab in a large first year class environment to support the learning of Calculus and Linear Algebra. This was achieved through a careful introduction by lecture demonstrations and a guide to the syntax. Dunn and Harman [3] extended this approach by introducing graphical user interfaces (GUIs) to visualise concepts and to make difficult algorithms accessible to students with limited programming abilities.

In this project we have developed a learning model and associated resources to deliver computational modules which (i) reinforce traditional mathematics learning, (ii) teach concepts in numerical mathematics and (iii) engage in problem solving with computers. An interactive workbook coupled with web-based hints and solutions, a structured sequence of Matlab programs and GUIs preceded by introductory demonstrations are instruments to achieve the delivery objectives. Survey results indicate that this model is accepted by the students as an effective instructional approach.

2 The project aims

From their experiences outside academia, the authors contend that mathematicians in industry are ineffective without a solid grasp of computational skills. Matlab is a widespread tool and covers a large cross section of functionalities.

The role of computers in first year mathematics courses varies widely between institutions. Computers are used variously to:

1. perform administrative tasks such as distribution of information or online quizzes
2. convey classical ideas in a more interactive way than a text
3. provide an introduction to numerical mathematics
4. provide an introduction to scientific computation
5. provide an introduction to programming techniques

The broad aim of the project was to more intensively integrate computational aspects into the first year syllabus while maintaining traditional elements in the courses. It was desired to incorporate all of points 2-5 within the curriculum with the intention to weave the topics into the traditional curriculum rather than treating the computational component as an “add-on”.

The inclusion of computers in these first year courses is not new and both Maple and Matlab have been used with varying degrees of success [6]. It is recognised that resistance from students arises from two main sources,

- The initial hurdle in learning the syntax of the software
- Seeing the direct relevance to the rest of the course

The project made concerted efforts to achieve the teaching aims while addressing the common concerns of students.

3 Challenges of teaching with computers

The simple catch-cry of "integrating computers into a mathematics curriculum" is not at all simple to decipher. The design of a course involves allocating a limited amount of time and student efforts amongst the traditional and computational course topics, and invariably conflicts arise. An optimal course design weaves together the curriculum so that student efforts exerted in one area benefit other areas also.

The fact that Matlab is a *numerical* rather than *symbolic* tool provides a very early hurdle for students, and requires a different way of thinking compared with the symbolic manipulation promoted in lectures and traditional tutorials. For example, the "=" sign has the meaning of assignment in the context of the computer, where $x = 1$ has a different meaning to $1 = x$.

Syntax is the main source of problems and frustrations for students [6]. Presenters desiring to demonstrate mathematical concepts using computer technology are stifled by the fact that students cannot appreciate the mathematics while they are hindered by syntax errors [1]. In fact, it has been found that syntax errors are likely to influence certain demographics of students to quit computer labs altogether [2].

It has always been a challenge for instructors to motivate students to appreciate the relevance of computational components to the classical mathematics material. The attitude of the lecturer plays a leading role to instil a sense of importance and indeed [6] claims "It is essential for the lecturer to bring the [Computer Learning Software] into lectures and demonstrate it so that students can see its application and how to implement it successfully and discuss any problems they might have as it is presented". Students need to see the relevance of computational skills for their degree and future careers. Continual references are required to promote the benefit of computational skills - even small gestures like publicising that graphs in the lecture notes are produced by Matlab will assist.

4 Learning model

We have developed a learning model that caters for diversity of background, needs, interests and mathematical and computational skills. This model is capable of bringing a large class of students up to speed quickly in a new language, to the level of solving interesting problems and priming them for further computational experiences.

Throughout the learning process, students work through a structured workbook. This workbook consists of a sequence of modules requiring increasing proficiency in Matlab. Each module addresses one core concept of the course curriculum in the context of computational experiments. The printed workbook contains mathematical ideas, concepts in numerical analysis and Matlab syntax interspersed with a sequence of examples and followed by student exercises. The workbook contains numerous blank spaces (see Appendix) which students fill in to (i) form a set of useful syntax references and (ii) table their results of numerical experiments. As students work through the material, they may query a roaming tutor or seek assistance from web-based hints and solutions linked to the workbook. Some of these used GIF animations to further illustrate difficult concepts. Each module was tested to confirm 45 minutes' duration and the difficulty and accuracy were validated by an external party.

The first two modules are presented by a demonstration tutor via a projector screen in the laboratory. The demonstrator explains how to negotiate the Matlab environment, starting with basic command line operations and leading to the execution of a Matlab program while the students reproduce the demonstrator's steps. This overcomes the majority of initial syntax problems and also builds students' confidence. These demonstrations reinforce the idea that Matlab is an integral part of the course and provide motivation for students' future attendance.

Matlab can be presented in three contexts, command line operations as an advanced calculator, running Matlab code execute algorithms and launching GUIs to illustrate mathematical concepts [3]. In our learning model:

- Students run simple single line commands at the command window to instil confidence and familiarity.

- Students progress to launching GUIs from the Matlab environment with the idea of visualising advanced mathematical concepts or sophisticated algorithms without the need to know the underlying Matlab code. Mathematics is not typically treated as an experimental subject in traditional first year teaching, however, GUIs provide accessible tools to explore.

- Students download and run prewritten programs that emphasise the core course material and are encouraged to adapt and extend the code. Students learn syntax and programming techniques by example. This assists with rapid familiarisation of Matlab code and syntax and students can engage in more complex programming tasks than traditionally possible.

- Finally, students are capable of developing their own code for assessable assignments which do not have the supporting framework of the workbook.

The learning model experimented with students working in pairs both for exercises and assessment.

The learning model was refined to gradually incorporate all of the above over several years. This allowed assessment of the effectiveness of these innovations. In an early stage (pre 2002), labs were held fortnightly, a worksheet listing syntax and problems was distributed and students undertook self-study. In 2002, an introduction to Matlab and exercises were posted on the course website. Exercises had some solutions and labs were held weekly. From 2003, the full learning model was implemented.

5 Specific Implementations

We present some specific examples from the learning model, the motivation and implementation of each mechanism for running Matlab.

Command line operations were used to demonstrate the notion of limits (a traditional topic) by performing a sequence of function evaluations, with the results recorded by the students in their workbooks to elicit a pattern. This experimental method also investigated numerical differentiation (finite differences) and resulted in an exploration of catastrophic cancellation (a computational topic).

The concept of infinite series and convergence always presents a challenge to first year students. To present this topic in a computational light, partial sums were explored by writing and running Matlab programs containing loops. This section covered traditional mathematical concepts, but required relative sophistication in programming techniques to convey the key ideas.

Analogously to [3], GUIs were run to implement algorithms which would be difficult for students to program. The appendix contains a screenshot of the GUI to undertake integration by a Monte Carlo method, which falls outside of traditional calculus curriculum but emphasises an important underlying concept.

Furthermore, GUIs were used to enhance understanding of traditionally difficult mathematical concepts by visualisation, for example, by plotting Taylor polynomials, students observe the way that the Taylor series converges in a radius of convergence. The appendix provides screenshots of a sample GUI to illustrate the interpretation of matrices as transformations in $\mathbf{R}^2$.

6 Outcomes

In 2002 and 2003 very similar lab problems were presented to students, but under different delivery frameworks. Surveys reveal a marked difference in how the computational component was received.

Surveys of 2002 and 2003 students provided the following freeform comments relating to negative experiences in Matlab classes

	2002	2003
Feedback	Proportion	Proportion
Should perform demonstrations in labs rather than self-paced study	24%	0%
Increase the number of tutors, especially around assignment time	23%	2%
General dislike with Matlab	16%	3%

Over the duration of the project, students responded to surveys regarding the computational components in first year courses. All items employed a Likert-style response format with options ranging from 1 (strongly agree) to 5 (strongly disagree). Mean responses which most markedly demonstrate the improvement in the students' experiences are:

	Question	Pre 2002	2002	2003
	Number of responses	398	254	67
1	There was enough introductory material to help me learn the computer packages	3.3	3.5	2.0
2	The computer assignments helped me understand the course	3.7	3.7	2.1
3	There was enough help available with computer problems	2.9	3.4	2.2
4	The computer assignments were the most interesting part of the course	4.2	4.1	2.4
5	I prefer to work alone rather than working in a pair	NA	NA	2.74
6	The interfaces are easy to use	NA	NA	2.32
7	The Matlab code is easy to understand and adapt	NA	NA	2.80

The introductory demonstrations and workbook with hints and solutions infrastructure clearly contributed to a much more positive student attitude to incorporating computers alongside the traditional mathematics curriculum (responses to 1 and 3 respectively).

Responses to question 2 support that the interactive GUI experiments and Matlab programs were successful for students to visualise underlying mathematical ideas. We suspect that in previous years, the syntax problems have hindered students from achieving understanding of the mathematics through the software tool. The responses to questions 6 and 7 illustrate that students have attained a reasonable level of understanding of Matlab, and as expected, find the GUI interaction simpler than writing code.

As a bonus, we have found that there has been improvement in attitude of the student body to computation. Responses to question 4 confirm that the learning model has captured increased student support.

Although students were regularly encouraged to work on one computer in pairs, there was much resistance and students preferred to have a computer to themselves. Similar to the findings in [5], cooperation was not hindered since there was open and frequent discussion between neighbours in problem solving.

A source of feedback that is frequently neglected in assessing the success of teaching innovations is the tutors. Tutors face the frontline of student appreciation or frustrations. Our large first year courses employ a large body of laboratory tutors (50 lab classes per annum) to roam the laboratories and assist with difficulties. Of those tutors who were present throughout the project, very positive feedback was offered. In particular:

- the quality of student questions was improved and students displayed a deeper understanding of the mathematics, rather than just the Matlab syntax
- tutors were not run off their feet by repeatedly answering the same question since the hints and solutions provided that assistance
- student retention was improved (with students even coming to multiple classes).

7 Conclusions

We have demonstrated a learning model that has worked well in practice, from the lecturer’s, tutors’ and students’ perspectives. A non-intimidating introduction through projected demonstrations yielded higher confidence, improved understanding and better attendance levels. A workbook which provided students with web-based assistance in the form of hints and solutions gave students more learning flexibility and tutors the freedom to concentrate on higher level teaching. The learning model employed a progression of Matlab commands, programs and GUIs to deliver programming skills, numerical mathematics knowledge, computational science exposure and illustrations to provide a better understanding of calculus, linear algebra and differential equations.

The learning model can be further improved by deeper integration of lecture and laboratory material. We agree with [6] that the attitude of the lecturer is crucial to the students’ acquiring a positive attitude and rewarding experience from the computational components of the course.

Survey results confirm that it is the totality of our proposed learning model which has improved learning outcomes over historical experiences.

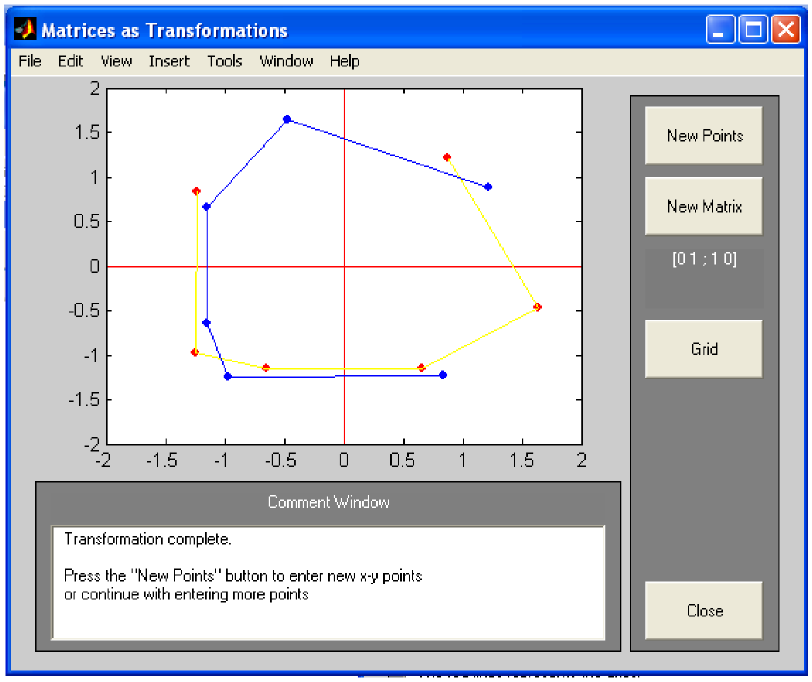
References

1. P. Cretchley, C. Harman, N. Ellerton & G. Fogarty, (1999) Computation, exploration, visualization: reaction to MATLAB in first year mathematics, Proceedings of Delta 99, 81--86.
2. P. Cretchley, C. Harman, N. Ellerton, G. Fogarty, (2000), MATLAB in early undergraduate mathematics: An Investigation into the effects of scientific software on learning,Math. Educ. Res. J., 12, 219-233
3. P. Dunn, C. Harman, (2002) , Calculus demonstrations using Matlab, Int. J. Math. Edu. Sci. Technol., vol. 33, no. 4, 584-596
4. P.Galbraith, (2002). Life wasn't meant to be easy: Separating wheat from chaff in Technology Aided Learning. Proceedings of the 2nd International Conference on the Teaching of .Mathematics (Hersonissos-Greece, 1–6 July 2002) Available at. http://www.math.uoc.gr/~ictm2/Proceedings/invGal.pdf.
5. G. Oates, (1999), The case for collaborative learning in tertiary mathematics, Proceedings of Delta 99, 148—154
6. M. Pemberton (2002), Integrating web-based maple with a first year calculus and linear algebra course, Proceedings of the ICTM, Available at http://www.math.uoc.gr/~ictm2/Proceedings/pap316.pdf

Appendix: Matlab GUIs and workbook extract

The matrix transformation interface is used to visualize the transformation of a vector through matrix vector multiplication in two dimensions. Students click on the axes with the mouse to define the red points and the program displays the transformed blue points on the same set of axes. Students are encouraged to experiment with different points and matrices to help understand matrix transformations.

The interface is ideal for demonstrating group transformations (reflections and rotations) by observing the transformation of a drawing. The interface is also successful for explaining eigenvalues and eigenvectors by illustrating the action of a matrix on the unit circle.

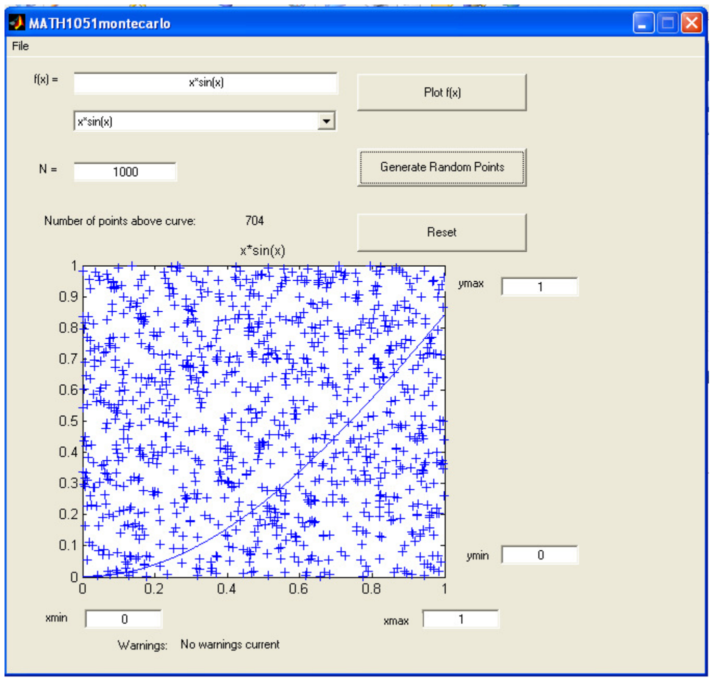

Matrices as Transformations
File Edit View Insert Tools Window Help
New Points
New Matrix
[0 1 ; 1 0]
Grid
Close
Comment Window

Transformation complete.

Press the "New Points" button to enter new x-y points
or continue with entering more points

Figure 1: Screenshot of the matrix transform GUI and sample input and output for visualising matrix transformations.

The Monte Carlo interface introduces students to a statistical integration method which falls outside of the lecture topics. Students enter or select a function that is plotted and the number of random points that will be generated within the plotting range. The ratio of points underneath the curve to the area of the range yields an estimate of the area under the curve (must be calculated outside of the GUI). Students experiment with the number of generated points to quantify the improvement in accuracy.



f(x) = x*sin(x)

x*sin(x)

N = 1000

Number of points above curve: 704

ymax 1

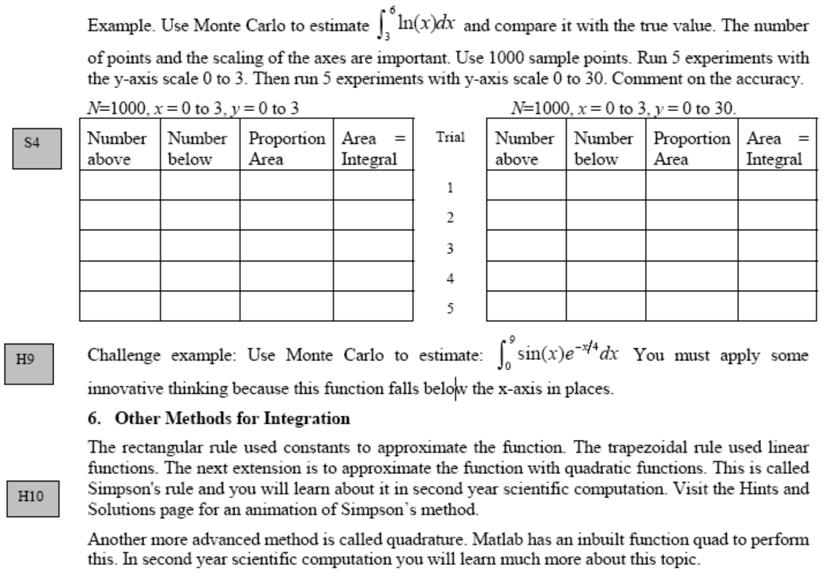
ymin 0

xmin 0

xmax 1

Warnings: No warnings current

Figure 2: : Screenshot of the Monte Carlo GUI and sample output.



Example. Use Monte Carlo to estimate $\int_3^6 \ln(x)dx$ and compare it with the true value. The number of points and the scaling of the axes are important. Use 1000 sample points. Run 5 experiments with the y-axis scale 0 to 3. Then run 5 experiments with y-axis scale 0 to 30. Comment on the accuracy.

S4

N=1000, *x* = 0 to 3, *y* = 0 to 3 | *N*=1000, *x* = 0 to 3, *y* = 0 to 30.

Number above	Number below	Proportion Area	Area = Integral	Trial	Number above	Number below	Proportion Area	Area = Integral
				1				
				2				
				3				
				4				
				5				

H9

Challenge example: Use Monte Carlo to estimate: $\int_0^9 \sin(x)e^{-x/4}dx$ You must apply some innovative thinking because this function falls below the x-axis in places.

6. Other Methods for Integration

H10

The rectangular rule used constants to approximate the function. The trapezoidal rule used linear functions. The next extension is to approximate the function with quadratic functions. This is called Simpson's rule and you will learn about it in second year scientific computation. Visit the Hints and Solutions page for an animation of Simpson's method.

Another more advanced method is called quadrature. Matlab has an inbuilt function quad to perform this. In second year scientific computation you will learn much more about this topic.

Figure 3 Extract from the workbook relating to module on Monte Carlo integration. The "H" or "S" on the left refers to a link for the hint or solution at the webpage.