Multi-disciplinary approach to teaching numerical methods to engineers using Matlab.

S. I. Barry[*] T. Webb[†]

(received 20 October 2005; revised 4 August 2006)

Abstract

Teaching Numerical Methods to first year engineering students is improved by a collaboration between engineers, computer scientists and mathematicians. Our approach has been to teach a problem based course with six, fortnightly assignments, each revolving around an engineering problem solved by a specific numerical method. The collaborative delivery, engineering problem focus, and intensive tutorials all contribute to make this a very successful course that gives the students a sound foundation to take into later years and hopefully their careers.

[*]School Physical, Environmental & Mathematical Sciences, University of New South Wales at ADFA, Canberra, Australia. mailto:s.barry@adfa.edu.au

[†]School of Aeronautical, Civil and Mechanical Engineering, University of New South Wales at ADFA, Canberra, Australia.

See http://anziamj.austms.org.au/V47EMAC2005/Barry for this article, © Austral. Mathematical Soc. 2006. Published August 31, 2006. ISSN 1446-8735

Contents

1 Introduction C217

2 Background history C218

3 How the course is run C220

4 Example problems C222

5 Results and impressions C226

6 Conclusion C229

References C229

1 Introduction

A course in Numerical Methods studies the way we use computers to solve mathematically based problems, a particularly important skill for engineering students. Traditionally this means covering the theory and practice of numerically calculating typical mathematical and engineering problems such as integration, interpolation, regression, root finding, solving systems of linear equations (matrices), and solving differential equations.

There are numerous textbooks which have evolved over the last forty years, most of which are still grounded in the original computational methods and languages. These books spend a great deal of time covering error propagation, the benefits of one integration scheme over another, the derivation of these schemes and so on. While this is useful, especially to computer scientists and expert practitioners in the field, it is less relevant to a practicing engineer and engineering students in their first years of study, where application of in-built routines is more useful. Students are also more motivated when they can see a direct application to their chosen profession.

An example of the book/technology mismatch is simple integration. It takes one page to describe the fundamental principles of numerical integration via Riemann sums. It takes three lines to illustrate the Matlab command `quadl` which will do many integrations numerically. Yet many books with titles *Numerical Methods using* Matlab (for example [4]), aimed at later year students, spend many pages describing competing methods, barely mentioning the simple Matlab command `quadl`. More targeted texts such as [2] emphasise the details of the numerical methods rather than their use in solving engineering problems. However, two recent books [3, 6] deal nicely with the learning of Matlab with chapters on numerical methods with applications.

Our aim in developing the course Engineering Computational Methods was to give first year students some of the basic tools for solving engineering problems, while still giving them some experience in writing more complicated programs and a few of the numerical issued involved in computational methods. We try to teach them the in-built Matlab commands, their uses and limitations, as well as an introduction to programming their own simple routines. The advantage of giving first year students an exposure to Matlab solving of problems, and the underlying numerical methods, is that these students can then use these skills for all their later courses, hence deepening their understanding of other topics. As an underlying principle we wanted the course to be engineering problem focused since having realistic application both motivates [5] students and gives them a deeper learning experience [7].

2 Background history

Some fifteen years ago, the Civil Engineering Program at unsw@adfa was being redesigned and one topic under the spotlight was Numerical Methods which was being delivered as a service to second year engineers by a computer science school. The course was presented from a computer science perspective with emphasis on topics important to those designing numerical methods but of peripheral interest to Civil Engineers. The decision was taken to deliver the course from within the school, by engineering staff with a great deal of professional experience using computing tools in their everyday work, as well as training in the underlying mathematics and computing. What evolved was a project based, tutorial style course based around the Matlab computing language.

In 2003 there was a major restructure of the unsw@adfa Schools and with the subsequent amalgamation of Engineering Schools it was decided to offer a similar course to all engineering students. This required some major changes, notably delivery to a much larger class, and also bringing the Course back from second year to the second half of first year. In the process of gaining approval the question was asked as to whether the Course should be taught by computer science, engineering or mathematics, each having legitimate claims of expertise—and the valuable student credit points on offer.

The decision was taken to run the Course jointly by the three separate disciplines with each putting their own perspective on the course. This presented many challenges such as: modifying the content to be manageable by first year students; catering to different engineering strands (aeronautical, civil, electrical and mechanical); coping logistically with the larger student numbers (130 in 2004) who would require appropriate computing facilities and assistance; streamlining the administration of a course taught by three schools; and, jointly writing a set of lecture notes and problem sheets.

In this article we first discuss how we ran the course before discussing the types of problems we assign to the students. We then give some more detailed descriptions of two of these problems, to illustrate our approach, before discussing the course outcomes.

3 How the course is run

The Course comprises six engineering problems taken from real world experience or related to more advanced engineering courses and was developed by the second author with assistance from the other two lecturers and engineering colleagues. Development of course notes, framing the mathematical and numerical basis and overall management of the course fell to the first author. A third lecturer (Darcy Brooker from School of Information Technology and Electrical Engineering) provided further numerical and computational insight.

Each topic is presented over a two week period, with two lectures, and four hours of supervised computer laboratory sessions. The lectures are jointly delivered by the three lecturers who also conduct the laboratory tutorials with the assistance of tutors.

A typical lecture delivery would start with a description of the engineering problem to be solved by the engineer, followed by descriptions of the mathematical background, the relevant Matlab functions, programming techniques and numerical issues. Most of this would be delivered over two hours at the start of each topic. The deep learning happens in the computer laboratories where the students tackle the assignments under supervision.

Through each topic the student has to come to an understanding of the engineering problem, be able to convert it into an equivalent mathematical formulation, design Matlab code to solve the problem and finally reinterpret it as the solution to the engineering problem. The students submit a written report with attached Matlab code together with its output.

The assignments account for about 50% of the assessment with an exam accounting for the remainder. The three hour, closed book exam consists of small problems designed to reflect the knowledge gained by doing the assignments.

Designing the problems

Some examples of the problems are:

- finding the current in an electrical circuit with many components (linear systems);
- finding the forces in elements of a truss frame (linear systems);
- replacing missing wave height data (interpolation);
- estimating the transmission of solar energy through the atmosphere (integration);
- modelling the pressure trace following an explosion (regression);
- finding the displacement of a pole due to the force of a wave (differential equations);
- finding the wave length of a tsunami for a given depth (solving nonlinear equations).

Each problem was *very carefully* designed so that it reflected a real life engineering problem but had definable outcomes in the form of a report to a fictitious supervisor (thus emphasising that this was an *engineering problem*, the outcome of which had to be communicated clearly to someone). The work had to be doable within the four hour tutorial, with perhaps one extra hour or so for preparing the report. The students were also prompted to explore some mathematical and computing issues so the assignment was not just a simple application of a one-line formula, but an exploration of uses and failings of 'black-box' computing tools.

Designing the course notes

As part of the course, detailed notes were produced, given the absence of our idea of a suitable textbook. These notes are available on the Course web site and were hypertext PDF, so that index and table of contents link directly to the appropriate page. All graphs in the text were produced by Matlab, and when a student clicks on these graphs the appropriate Matlab code appears and can be copied by the student. Hence students can easily use the code as templates or examples for their own work. In addition, a detailed Matlab help page was produced which incorporates our own Matlab guide as well as several on-line guides taken from other universities (with permission) [1].

4 Example problems

To illustrate our approach we outline two problems: finding the zero of a nonlinear function for wavelength; and, solving differential equations for a bending beam.

Zeroing nonlinear functions

This assignment involved finding the wavelength, L, of a tsunami for a given water depth—an apparently trivial task but with enough aspects to make it interesting for the student. The governing equation is

$$f(L) = L - \frac{g\,T^2}{2\pi} \tanh\left(\frac{2\pi d}{L}\right) = 0\,, \tag{1}$$

where g is gravity, $T = 2\,880\,\mathrm{s}$ is the period (calculated from an earlier assignment), and $d \approx 4\,000\,\mathrm{m}$ is the water depth typical for the Indian Ocean.

Typical questions for the students were briefly as follows:

1. Plot $f(L)$ and graphically estimate the zero (L).
2. Use `fzero(@f,`10^6`)`, `fzero(@f,1)`, `fzero(@f,[1,`10^6`])` to find the zero.
3. Approximate the function using $\tanh x \approx x$ to get
$$f_1(L) \approx L - \frac{gT^2}{L} \quad \Rightarrow \quad L \approx \sqrt{gT^2 d}\,. \tag{2}$$
4. Use $\tanh x \approx x - x^3/3$ to obtain
$$L^4 - (g\,T^2 d)L^2 + gT^2 d^3 4\pi^2 \approx 0\,, \tag{3}$$
and use the `roots` command in Matlab to find the zeros of this function.
5. Write a Newton's method code to find the zero.
6. If $d = 4\,000 \pm 50$ what error does one get with L?

We expected the students to comment on what their results meant. For example, the plot of the $f(L)$ along with the approximate versions of $f(L)$ using the asymptotics of tanh are illustrated in Figure 1.

The student should note that the `fzero(@f,`10^6`)` gives the answer accurately, since $L = 10^6$ is a good first guess, but after several iterations. However, `fzero(@f,1)` gives the incorrect result of $L = 0$ since Matlab actually finds the discontinuity of $f(L)$ at $L = 0$, while `fzero(@f,[1,`10^6`])` (giving Matlab the range $x \in [1, 10^6]$) gives the result accurately and quickly.

The student then shows that $L \approx \sqrt{gT^2 d}$ gives an acceptable result with an error of 0.003%. Doing the cubic approximation also gives an accurate answer, but that one has to know enough to discard the inaccurate root at $L = 14\,500$. Doing a little programming to do Newton's method, the student then finds that an accurate answer is obtained with fewer steps than using `fzero` even with a starting value of $L = 1$.

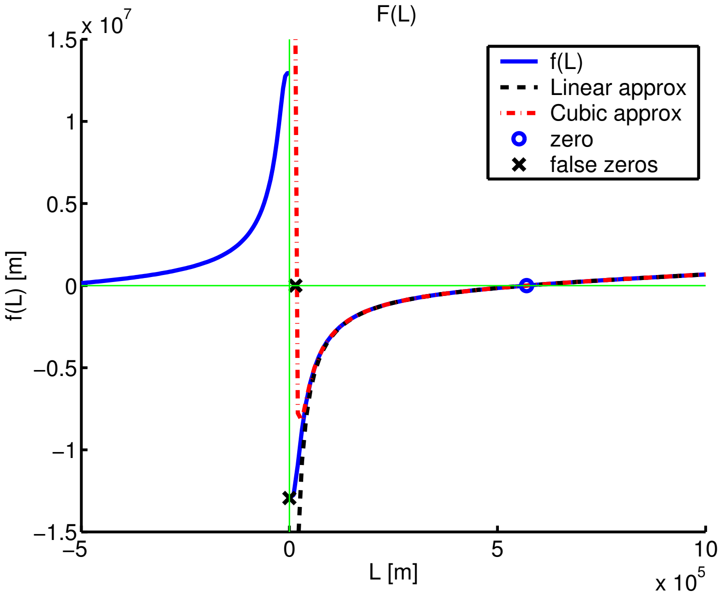

Figure 1: Objective function $f(L)$ with various approximations.

By showing that the error in d gives an error of $L = \pm 0.006\%$ the student should then realise that super-accurate and efficient methods are of little use if the error in the data swamps the method used.

The student finally concludes that $L \approx \sqrt{gT^2 d}$ gives an acceptable solution given the errors in d, but that if more accuracy is required then Newton's method is the fastest, and MATLAB in-built command `fzero` works almost as well, with some guidance. Hence, perhaps the conclusion we most want the student to realise is this: no method is ideal and gives perfect answers every time; one must always consider many options and think carefully about the results.

Differential equations

A second example is finding displacement $y(x)$, for a pole bending under wave drag force, as governed by

$$y'' = \alpha(h-x)^2(1+y'^2)^{3/2}\,, \quad y(0) = y'(0) = 0\,, \tag{4}$$

where α is a constant, x is the height up the pole, and h is the water depth. Here we get the student to solve this using MATLAB `ode45` having first solved the linear approximation

$$y'' = \alpha(h-x)^2 \tag{5}$$

both analytically and numerically. One of the problem solving strategies addressed in this assignment is to first solve a problem with known solution (the linear approximation) before solving more complicated nonlinear problems. In addition we get the student to write a simple Euler solution and comment on the various solutions.

This assignment coincided with the first year mathematics course where converting a second order differential equation to systems of first order differential equations was being discussed. There was thus opportunity to address the problem of compartmentalism by reinforcing and expanding on the material from the mathematics course.

5 Results and impressions

Developing this course was not easy. Apart from the administrative difficulties of a course taught jointly by three lecturers, and three schools, there were the difficulties of reconciling our different approaches. As expected, each of us wanted different emphasis placed on different parts of the course. However, the resultant course was much better than we could have done separately. For example, the engineering examples were more realistic that would have been invented by a mathematician, but the input of mathematics and computer science meant the students appreciated more the background to the solution method and did more rigorous testing of their routines. An additional benefit was that collaboration extended beyond the confines of the course with more interaction between schools on the development and administration of our Engineering Mathematics courses. We also found that each of us learned a great deal, with even our Matlab programming skills improving with the input of ideas from the other lecturers.

We have seen many benefits for the students with this Course. Students learned how to write well structured assignments. We are very strict on presentation and completeness with the solutions. Initial assignments were typically rather incomplete and scrappy, whereas by the end most were producing well polished assignments with well crafted and commented computer code as well as sound engineering interpretation. At the beginning many students were computer phobic. This disappeared by the end and most students were happy and confident in solving problems with Matlab . While the assignments had to be done individually, we saw many pleasing aspects of teamwork in the tutorials. Students were genuinely helping each other and keeping the fine balance between collaboration and plagiarism. Most student quickly became experts in the basics of setting up a Matlab program, using a range of plotting features, writing function subroutines, and sorting out how to use new commands. They also gained skills in the overall approach to solving engineering problems—something that will continue into their later years.

Positive comments were received from the students in staff-student review meetings, and elsewhere, and also from our colleagues. The results of a student survey showed that the majority of the class saw the value in what we were doing although in the first year (2004) they commented that the assignments were too long.

We also had relevant comment from students who had not done the course. In one instance, one of us (Steve Barry) found several second year students working on an engineering assignment that required Matlab . These students had not done the course because 2004 was the first year it was offered. In helping them Steve showed them what was available in the Course. They then enthusiastically used the notes to learn what they needed and were very vocal in their disappointment at *not* being taught this course when they were in first year. They felt it was a crucial course for them and one they wished they had done.

While the Course appears to be successful even in its first year of offering, it has not been without its problems:

1. We had some instances of plagiarism in 2004. We designed the assignments to be doable within the laboratory time frame; however, only about half of the students were completing their work in the laboratory times. In retrospect, we made the assignments in this first year too long, and hence many students felt undue pressure to finish and resorted to plagiarism. In 2005 the assignments were shorter and more focused, and there were no cases of plagiarism. In addition we had a flexible marking scheme whereby a student may count either four, five or six of the assignments towards their final mark. This meant that

the pressure to perform well on each assignment was reduced, and a student could do badly on an individual assignment and still do well overall.

2. Administratively, the timetabling of computer laboratories was very difficult with the constraints of the number of students, the heavy engineering timetable and the equally heavily booked computer labs. This was made easier in 2005 since we advised students at the beginning of the year to purchase their own notebook computers with Matlab—something that about 80% of students did and which the students found very useful, being able to bring their work easily to us for help.

3. There were numerous administrative problems with a joint course of this nature, from mis-scheduling of exams to the simple task of three people editing the one set of notes.

4. Given the problem-based computer nature of the course, it would have been nice to have assignments worth 100% of the mark, or an exam using computers. However, given the large student numbers, and the possibility of plagiarism, a written exam was necessary. In addition, unsw@adfa does not have the facilities for simultaneous computing exams for 130 students. As students move to notebook computers we should be able to address this issue.

While there were many areas we were not able to cover in this course, we felt the students learned the skills to apply their programming and Matlab knowledge to any topic they might later come across. This is being reinforced in computational engineering courses in later years as many assignments are being set for Matlab solution.

6 Conclusion

Our teaching of applicable numerical methods to Engineering students benefits greatly from being jointly taught by engineering, mathematics and computer science academics. The students are more motivated, see a more balanced course and quickly learn how to solve an engineering problem with MATLAB . While initially a difficult course to organise and develop, once set up, it is a course whose benefits to the student and the staff go well beyond the confines of the course itself.

Acknowledgements: We thank Darcy Brooker for computational insights and the input of colleagues and students.

References

[1] S. Barry. MATLAB *guide and online tutorial help.* http://www.pems.adfa.edu.au/~s8704008/MATLAB.dir/MATLAB.html. C222

[2] S. J. Chapra. *Applied Numerical Methods with* MATLAB *for Engineers and Scientists.* McGraw Hill, 2005. http://www.mhhe.com/engcs/general/chapra/ C218

[3] A. Gilat. MATLAB. *An introduction with applications.* Wiley, 2005. http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471694207.html C218

[4] G. Lindfield and J. Penny. *Numerical Methods using* MATLAB. Prentice Hall, USA, 2000. URL C218

[5] V. E. Martinez Luaces and G. E. Guineo Cobs. *Numerical Calculus and Analytical Chemistry*. Proc. 2nd Int. Conf. on Teaching Mathematics, Crete, Wiley, 2002.
http://www.math.uoc.gr/~ictm2/Proceedings/pap289.pdf C218

[6] H. Moore. Matlab *for Engineers*. Pearson, 2007. http://www.pearsoned.co.uk/Bookshop/detail.asp?item=100000000114129 C218

[7] D. Smith. *How people learn ... Mathematics*. Proc. 2nd Int. Conf. on Teaching Mathematics, Crete, Wiley, 2002.
http://www.math.uoc.gr/~ictm2/Proceedings/invSmi.pdf C218