A computational mathematics education for students of mechanical engineering

Mikael Enelund[1] and Stig Larsson[2]

[1]Department of Applied Mechanics, Chalmers University of Technology, SE-412 96 Göteborg, Sweden
[2]Department of Mathematical Sciences, Chalmers University of Technology, SE-412 96 Göteborg, Sweden

ABSTRACT: In this article, the authors present a new model for computationally oriented mathematics education. This education combines traditional symbolic mathematics with computational mathematics and programming in the Matlab environment. Engineering applications are explored in computational exercises that are taught jointly with the courses in mechanics and thermodynamics at Chalmers University of Technology in Göteborg, Sweden.

INTRODUCTION

The Mechanical Engineering program at Chalmers University of Technology has taken part in the development of the CDIO [1] model of engineering education since 2000. For example, in the courses in mechanics and strength of materials, a common methodology of mathematical modeling and abstract thinking is emphasized. Important goals are: to be able to set up mathematical models, to formulate the models in the form of equations, to simulate the phenomena by solving the equations on the computer, and to analyze the simulations in order to assess the correctness of models and solutions, as well as to improve the learning of basic phenomena and concepts. The interaction between courses is also emphasized.

At the same time, new mathematics courses for the engineering education have been developed at Chalmers and implemented in the Chemical Engineering and Bioengineering programs since 1999. These courses emphasize mathematical modeling, simulation, the use of modern computational tools, and interaction with courses in chemistry and chemical engineering. This is achieved by taking a computational (constructive) approach to the teaching of mathematics.

A basic idea of the reformed mathematics courses is a full integration of the computational (numerical) aspects of mathematics (including programming in the Matlab environment), and the analytical (symbolical) aspects. The traditional separation of these aspects, where the analytical mathematics is usually presented in the first year and the computational mathematics in a later course, is not adequate.

We believe that the computational aspects of mathematics should be presented from the start. This permits the discussion of, for example, nonlinear algebraic and differential equations, and hence also the introduction of realistic applications from applied subjects such as mechanics and thermodynamics. This creates motivation for studying the analytic mathematics and raises the level of both the mathematics and the applied courses. The students write their own equation solvers in Matlab based on the algorithms presented in the lectures, which focus on related analytic concepts such as convergence, limit, linearization, and derivative.

In this paper we present our ongoing work on developing such mathematics courses for the Mechanical Engineering program at Chalmers. If they are accepted, the new courses will be launched in the academic year 2007-08.

More specifically, our work involves:

- Developing course materials for computational mathematics to supplement the traditional textbooks that are used;
- Developing computer-oriented projects and exercises that will be used simultaneously in the mathematics courses and the courses in mechanics and thermodynamics;
- Developing a new introductory course in Matlab programming.

Mathematics is a fundamental subject in the engineering education and our goal is not a "mathematics for mechanics" toolbox-based course. However, both subjects can benefit from interactions and exchange of examples.

COURSES

We list the mathematics courses in the first year of the Mechanical Engineering program together with the accompanying engineering courses. The year is divided into four periods (quarters of eight weeks).

Period 1.

- Programming in Matlab (4.5 ECTS)
- Introduction to mathematics (7.5 ECTS)
- Introduction to mechanical engineering

Period 2.

- Analysis and linear algebra A (7.5 ECTS)
- Thermodynamics (7.5 ECTS)
- Introduction to mechanical engineering (7.5 ECTS), continued

Period 3.

- Analysis and linear algebra B (7.5 ECTS)
- Mechanics and solid mechanics I (7.5 ECTS)

Period 4.

- Analysis and linear algebra C (7.5 ECTS)
- Mechanics and solid mechanics II (7.5 ECTS)

The third year also contains two mathematics courses: *Mathematical statistics* and *Transforms and differential equations*. These are not considered at the moment and are therefore not included in the work reported here.

The following is a short overview of the contents of the mathematics courses and their connections with the accompanying engineering courses.

Period 1. *Programming in Matlab*. Introduction to Matlab. General programming concepts and techniques. *Introduction to mathematics*. Number systems, elementary functions, derivative, integral, graphs, geometry in space, vector, elimination method. Common project: function gallery.

Period 2. *Analysis and linear algebra A*. Nonlinear algebraic equations. Derivative. Integral. Ordinary differential equations. Matrix algebra, linear systems of equations, linear independence. Common projects with *Thermodynamics*: steady state heat equation, equilibrium equations.

Period 3. *Analysis and linear algebra B*. Eigenvalue problem for matrices and differential equations. Linearization and stability for systems of differential equations. *Mechanics*. Statics. Principal stresses. Differential equations for bars and axles.

Period 4. *Analysis and linear algebra C*. Analysis in several variables. Introduction to partial differential equations, boundary value problems. Finite element method. *Mechanics*. Elasticity, plane problems. Beams and plates. Stability. Fracture mechanics.

TEACHING

The teaching is organized as follows.

- Lectures.
- Exercises, both analytical and computational.
- Computational assignments common to both the mathematics course and the accompanying engineering course.

From the viewpoint of the mechanics and thermodynamics courses the purpose of the common computational assignments is:

- To allow more complex applications compared with the traditional analytical methods or handbook methods.
- To illustrate phenomena such as deflection curves, stability, stress concentrations, stress distribution and fracture, to make parameter studies.
- To give an introduction to working with realistic, complex, engineering problems.

From the mathematical viewpoint the purpose is:

- To give a better understanding of mathematical concepts.
- To motivate the study of mathematics.

An overall purpose is to strengthen the ability to apply a modern way of working based on modeling, simulation, analysis; an approach which is generally applicable in all engineering subjects, not just mechanics. Several courses in the second and third years involve computational simulation, *e.g.*, *Mechanics, Mechatronics, Machine design, Machine design project, Manufacturing, Finite element methods (elective), Control theory* and *Fluid dynamics*. The teachers of these courses will be informed about the changes of the mathematics education so that they take proper advantage of the new skills and knowledge. This will be followed up in the course evaluations as well as in the yearly program evaluation.

MATLAB

Matlab (Matrix Laboratory) is a software for numerical matrix computations. It can be used at many levels, from a simple calculator to a rather advanced interactive programming environment. It contains advanced tools for graphics and for creating graphical user interfaces, as well as "toolboxes" for many problem areas of science and engineering. We have chosen to use Matlab as our programming environment. We do not use any software for symbolic calculation such as Mathematica or Maple. We are aware that this may be viewed as an inappropriate promotion of commercial software to the students, in particular, because the license cost has increased dramatically recently. However, the lack of competitors with the same flexibility, as well as the long tradition among researchers and lecturers, motivates the choice of Matlab as our software.

We use Matlab in three ways: (i) for illustrating mathematical concepts using ready-made interactive programs; (ii) for letting students write their own software for implementing the numerical algorithms that are used in the courses; (iii) for presenting results in the form of graphs, plots and simulations.

The motivation for (i) and (iii) is obvious. The second one may require some comments. Even though Matlab contains tools for most of the computations that we need to do, we let students write their own programs, from the simple bisection algorithm to the rather advanced finite element method. We believe that there are pedagogical advantages with this. Clearly it gives the students skills in programming and implementation of numerical algorithms. But it also forces understanding of the algorithms and the mathematics behind them; Matlab is more unforgiving against logical errors and sloppy typing than any teacher. Our constructive approach to mathematics based on numerical algorithms gives the computer exercises a natural and strong connection with the other forms of teaching. Finally, we hope that the students will gain self-confidence from using their own software, based on mathematics that they understand, to model advanced engineering systems, instead of running "black box" simulations with ready-made programs.

GENERAL VERSUS SPECIAL EQUATION

The use of numerical algorithms makes it possible for us to discuss equations in their most general forms and to use them in mathematical models in engineering and science. In contrast, a discussion of the solvability of general ordinary differential equations is outside the scope of a traditional first year mathematics course, and numerical methods are often treated in a later course. Since there is little that can be said, or done, about the general case, all emphasis is put on special cases that can be solved symbolically. We try to reverse this order of priority.

A *general algebraic equation* is of the form $f(x)=0$. A special case is $x^2+ax+b=0$, which is solved symbolically by the formula

$$x = -a/2 \pm (a^2/4-b)^{1/2} .$$

Note that this is not an "exact solution" because it is no more accurate than the accuracy in computing the square root, which done by solving $f(x) = x^2 - 2 = 0$ numerically. However, it provides a formula, which we can manipulate by analytical techniques.

A *general ordinary differential equation* is of the form $u'(t) = f(t,u(t))$. A special case is $u' = au$, which is solved symbolically by the formula

$$u(t) = A\exp(at).$$

Again this is not an "exact solution"; it expresses the solution in terms of a well-known special function, which can only be evaluated numerically.

The possibility to solve general equations numerically does not diminish the importance of special solutions such as $\cos(\omega t)$ or $\exp(at)$. On the contrary, they help us interpret the numerical results and to understand complex behavior in terms of simple well-known cases. But there is no reason to pursue symbolical computation to its extreme; it is the simple cases that are useful.

A CONSTRUCTIVE APPROACH

The most important feature of the reformed mathematics courses is that we try to take a *constructive* approach based on numerical algorithms. For example, after we have studied the bisection algorithm, we note that it proves the intermediate value theorem, as explained below. We believe that this constructive/computational approach has the following advantages:

- it makes the mathematics more understandable;
- it makes it possible to discuss general equations, not just simplified special cases;
- it makes it possible to do applications early in the curriculum;
- it makes it possible to reach advanced applications at the end of the curriculum;
- it allows open-ended project work.

This approach is based on the textbook [2] and has been implemented since 1999 in the Chemical Engineering and Bioengineering programs at Chalmers University [3]. However, the book [2] has proved to be somewhat too difficult for the students and we plan to use traditional textbooks complemented by lecture notes.

We illustrate the approach by describing the bisection algorithm and the intermediate value theorem here. We identify the real numbers $\mathbf{R}$ with the set of all decimal expansions. If we have a decimal expansion, then we can form an approximating sequence x_n by truncating it after n decimals. On the other hand, a convergent approximating sequence provides a decimal expansion. (This is made rigorous by means of the mathematical concept of Cauchy sequence.) Thus, a real number cannot (in general) be specified exactly; however it can be evaluated up to *any desired accuracy*.

The use of numerical algorithms to construct new objects is fundamental to our program. The students first encounter such a constructive argument in the classical construction of $\sqrt{2}$ by solving the algebraic equation $f(x) = x^2 - 2 = 0$ by means of the bisection algorithm. Each student is instructed to write a Matlab program implementing this algorithm. The result of the computation with the program is presented in a table with x_n denoting the n^{th} entry of the table. The students observe that the decimals are fixed as we go down the table, which indicates that they form a decimal expansion. Indeed, we prove that, by construction, $|x_n - x_m| \le K\,2^{-n}$ if $m \ge n$, which shows that the algorithm always generates a decimal expansion. We conclude that we have constructed a new real number (decimal expansion) $x = 1.41421356\ldots$

The students are then instructed to prove that x solves the equation in the sense that the residual tends to zero, $f(x_n) \to 0$. In other words, $f(x)=0$. Finally, we note that f is strictly monotone for $x>0$, so that the equation $f(x)=0$ cannot have two positive solutions; the solution is unique. Therefore, any other construction would give the same result. This new number is so important that we give it a name: $\sqrt{2}$.

Note the following steps in the constructive argument:

- an algorithm that generates a decimal expansion;
- a proof that the limit solves the equation;
- a proof that the solution is unique.

The bisection algorithm proves *the intermediate value theorem*: A continuous function $f : [a,b] \to \mathbf{R}$ attains all values between $f(a)$ and $f(b)$. Proof: If y is an intermediate value, then solve the equation $f(x) - y = 0$ by means of the bisection algorithm.

An analogous treatment is also made to *systems of algebraic equations* (algorithm: Newton's method), *systems of ordinary differential equations* (algorithm: Euler's method), and *partial differential equations* (algorithm: the finite element method).

EXAMPLES OF APPLICATIONS

Here we present two examples of joint computer assignments from mathematics and mechanics. The first one is placed in period 3 while the second one is placed in period 4. The assignments will be tutored by lecturers and assistants from both mathematics and mechanics.

The first assignment is a static analysis of a plane truss by the displacement-based matrix method. The truss is shown in Figure 1. The task is to determine a value of the area parameter A so that the magnitude of the normal stress is less than half the yield stress everywhere in the truss. Further, using this information the normal stresses in all members and the joint deformations are to be calculated. The deformed truss should be displayed and a figure showing the relative stress in each member should be produced. The student needs to handle matrices with dimensions up to 20×20 and a Matlab code must be written. Examples of aims are: (1) To provide an in-depth understanding of fundamental principles used for static analysis and corresponding computational procedures. (2) To serve as an introduction to the application of the finite element technique in structural mechanics. (3) To train the student in

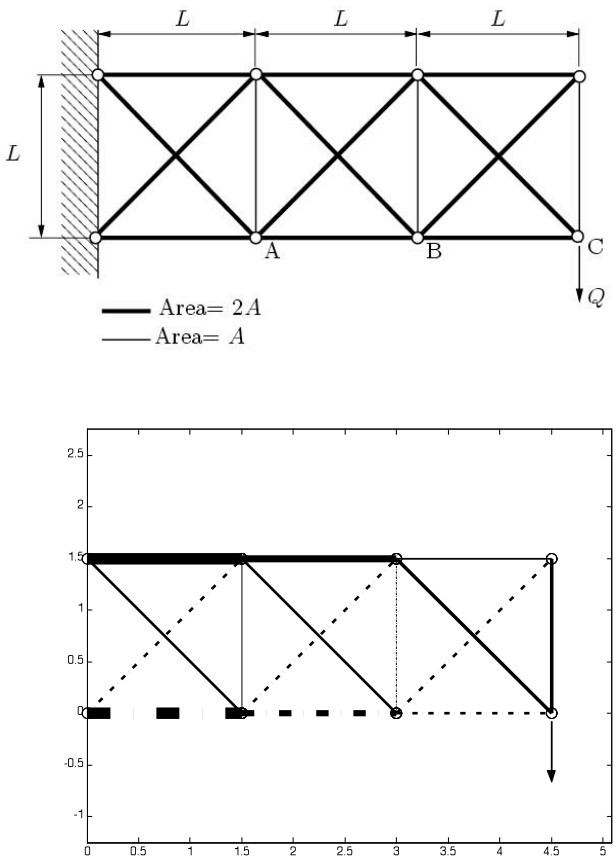

Figure 1: Top: plane truss to be analysed. Bottom: the line width indicates the relative stress in the members (bars), a continuous line indicates tension while a dash-dotted line indicates pressure.

(Matlab) programming from problem definition to working code. (4) To use graphical tools for presentation of results and for better understanding. (5) To understand the mathematical treatment of large linear systems of equations.

The second application considers a two-dimensional stress analysis of a thin plate with three holes subjected to uniform stress at the vertical boundaries, see Figure 2. Plane stress conditions are assumed. The analysis is carried out using the finite element method and the PDE-toolbox in Matlab. The task is to calculate the stress concentration factor K_t, which is defined as $K_t = \sigma_{max} / \sigma_{nom}$. By varying the distance b, the students should be able to decide whether the stress concentrations near the holes are correlated or not. Further, the calculated K_t should be compared with tabulated values from handbooks. To reduce the number of elements, symmetries should be used and only a quarter of the plate needs to be considered. This means that special attention needs to be put on the choice of boundary conditions.

There are several aims with this assignment, for example: (1) By visualizing the stress distribution, the students can develop an intuition about stress distributions and how the stress is increased due to abrupt changes in geometry. (2) Motivate the need to study the governing equations of elasticity. (3) It serves as an introduction to the finite element method. (4) It provides an introduction to error estimation and adaptive mesh refinement in the finite element method.

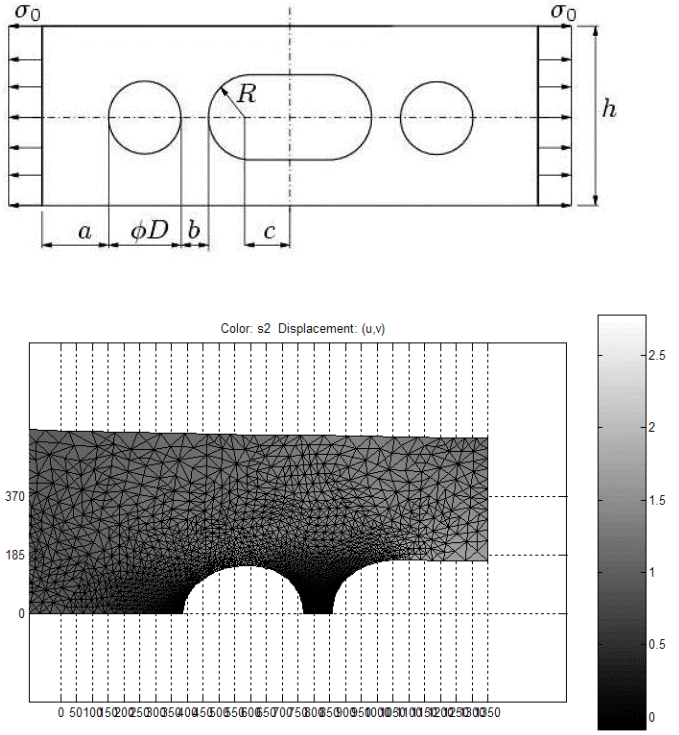

Figure 2: Top: plate with three holes. Bottom: the shading represents the stress distribution in a quarter of the plate. The largest principal stress is presented. The applied stress is σ_0=1. Note the higher stress near the holes and the mesh refinement around the holes. Note also the deformed geometry.

DISCUSSION

From the small scale that this has already been implemented in the mechanics courses, as well as our experience from the Chemical Engineering program [4], it is clear that the students appreciate the ability to work with realistic models and the possibility to gain insight and understanding of the behavior of the systems studied. For instance, the students have used these tools in the 2nd year design project and the quality of the analysis has been significantly raised. Further, we strongly believe that the proposed education also has the potential to increase the interest for the underlying mathematics. Clearly, the proposed approach strengthens the connection between the applications and mathematics. This is very important for the engineering education, having in mind that mathematics is the fundamental tool for most of the engineering students. This kind of mathematics makes it possible to study the complete problem: from modeling and solution to simulation of the system and comparison with physical reality. This is, as far as we understand, one of the corner stones of the CDIO curriculum. However, our experience also shows that it is important to emphasize symbolic hand calculation and basic programming concepts in the teaching, so that these are not lost in the excitement over the possibility of doing simulations.

REFERENCES

1. http://www.cdio.org
2. Eriksson, K., Estep, D., and Johnson, C., *Applied Mathematics – Body and Soul*, Springer (2003).
3. http://www.math.chalmers.se/cm/education/courses
4. Öhrström, L., Svensson, G., Larsson, S., Christie, M., Niklasson, C., The pedagogical implications of using Matlab in integrated chemistry and mathematics courses, *Int. J. Engng. Education* 21, 683-691 (2005).