Enhancing Numerical Analysis Teaching with MATLAB

Carola-Bibiane Schönlieb
Department for Applied Mathematics and Theoretical Physics
University of Cambridge, UK

MATLAB EXPO - October, 8th 2013
The idea . . .

Point of attack:

 Classical setup of the numerical analysis courses in Cambridge does not require the students to have programming skills but concentrates on teaching them concepts and numerical methodologies as well as mathematical analysis.

MATLAB enhancement:

 Illustrate the concepts underlying numerical analysis, as well as demonstrating the power of modern, advanced scientific-computing software, to accompany numerical analysis instruction by computer demonstrations.
NA Curriculum in Cambridge

Undergraduate Teaching
- Part IB (2nd year mathematics),
- Part II (3rd year mathematics).

Graduate Teaching
- Part III (master course, 4th year mathematics).
NA Curriculum in Cambridge

Undergraduate Teaching
- Part IB (2nd year mathematics),
- Part II (3rd year mathematics).

Graduate Teaching
- Part III (master course, 4th year mathematics).
Part IB NA – Learning Outcomes

By the end of this course, you should:

- understand the role of algorithms in numerical analysis;
- understand the role and basic theory (including orthogonal polynomials and the Peano kernel theorem) of polynomial approximation;
- understand multistep and Runge–Kutta methods for ordinary differential equations and the concepts of convergence, order and stability;
- understand the theory of algorithms such as LU and QR factorisation, and be able to apply them, for example to least squares calculations.
By the end of this course, you should:

- understand the **solution of partial differential equations** (Poisson, diffusion, advection, wave equation) by
  - **finite differences** (consistency, convergence, stability),
  - **spectral methods** (Fourier expansion, FFT);

- understand **iterative techniques** for linear equations;

- understand the **calculation of eigenvalues and eigenvectors**;
The team behind

DAMTP faculty:
- Stephen Cowley
- Arieh Iserles
- Alexei Shadrin
- CBS

DAMTP students:
- Ben Champion
- Francesca Balestrieri
- Simon Morris

MathWorks representatives:
- Tanya Morton
- Coorous Mohtadi (MathWorks Cambridge)
Outline of the project

- MATLAB implementations of numerical algorithms embedded in GUIs for straightforward use by students outside class. These are linked to the example sheets that the students have to work on by themselves.
Outline of the project (cont)

- A "Numerical Analysis Webpage", which includes all the course material, i.e., lecture notes, example sheets, MATLAB demos and MATLAB codes. For this we took advantage of the Publish option of MATLAB.
Outline of the project (cont)

- A “**Numerical Analysis Webpage**”, which includes all the course material, i.e., lecture notes, example sheets, MATLAB demos and MATLAB codes. For this we took advantage of the **Publish** option of MATLAB.

---

**Boundary conditions**

It is also possible to change the boundary conditions which the methods use. The general boundary conditions can be "Robin" boundary conditions:

$$a u + b \frac{\partial u}{\partial n} = g$$

on the boundary of the domain. On the case of the domain $[0,1]$ this reduces to:

$$au(0) - bu'(0) = g(0), au(1) + bu'(1) = g(1)$$

From this general form, we can obtain Dirichlet boundary conditions - letting $a = 1, b = 0$ and Neumann boundary conditions - letting $a = 0, b = 1$.

The way in which the code implements boundary conditions is somewhat special. For Dirichlet boundary conditions it is reasonably simple, since we just need to fix the values $u_{N+1} = g(0)$, $u_{M+1} = g(1)$. For Neumann boundary conditions we could use the one-sided approximation of the normal derivative $\frac{\partial u}{\partial n} = -g(0)$, but this would only yield a first order method. In order to sustain a second order method, we need to be more careful about our choice of approximation for $\frac{\partial u}{\partial n}$.

The way we do this involves introducing "Ghost points" which are never actually used or referenced in the code, $u_{-1}$ and $u_{M+2}$. Then we can use a symmetric approximation to the normal derivative:

$$\frac{u_{-1} - u_{-2}}{2\Delta x} = -g(0).$$

This allows us to compute a value for our ghost point $u_{-1} = u_1 + 2\Delta x g(0)$. We can then plug this into the general form of our equation.

Working out Robin conditions involves using the same process as for Dirichlet conditions again.

**Code**

- `stability.m` *(Run this)*
- `stability.fig` *(Required - GUI figure)*

All files as .zip archive: `pde_stability_all.zip`

Published with MATLAB® 7.10
A tour through the MATLAB material

http://www.maths.cam.ac.uk/undergrad/course/na/
Example 1: QR algorithm

Mathematical explanation

NA-webpage & MATLAB GUI

Schönlieb (DAMTP)
Example 2: FFT

Mathematical explanation

Dr C.-B. Schönlieb
Mathematical Tripos Part II: Michaelmas Term 2012
Numerical Analysis – Lecture 4
Algorithm 1.19 (The fast Fourier transform (FFT)) We assume that $n$ is a power of 2, i.e. $n = 2m = 2^k$, and for $g \in \mathbb{C}^{n}$, denote by

$$f^{(0)} = (g_{0}, g_{1}, \ldots, g_{n-1})$$

the even and odd parts of $g$, respectively. Note that $f^{(0)}, f^{(1)} \in \mathbb{C}^{n/2}$.

Suppose that we already know the inverse DFT of both short sequences,

$$f^{(0)} = \mathcal{F}^{-1}(g^{(0)}), \quad f^{(1)} = \mathcal{F}^{-1}(g^{(1)}),$$

It is then possible to assemble $f = \mathcal{F}^{-1}(g)$ in a small number of operations. Since $\mathcal{F}^{-1} f^{(0)} = 1$, we obtain

$$g_{0} = \sum_{k=0}^{n-1} f^{(0)}(k),$$

Therefore, it costs $n$ products to evaluate the first half of $f$, provided that $f^{(0)}$ and $f^{(1)}$ are known. It actually costs nothing to evaluate the second half, since

$$g_{m} = \sum_{k=0}^{n-1} f^{(1)}(k).$$

To execute FFT, we start from vectors of unit length and in each $\ell$th stage, $s = 1, \ldots, 2^\ell$ puts together $2^\ell$ vectors of length $2^\ell$ from vectors of length $2^{\ell-1}$, this costs $2^{\ell-1} \times 2^{\ell}$ products. Altogether, the cost of FFT is $2^{n-1} = 2^{p(\log n)} = p(n)$ products.

For $n = 2^{\ell}$, one may compute the FFT as a product of $\ell$ matrices, each of size $2^{\ell-1}$, provided that $n$ is a power of 2.

Mathlab demos: Check out the online animations for computing the FFT at http://www.maths.cam.ac.uk/undergrad/course/na/ii/fft_gui/fft_gui.php and download the Matlab GUI from there to follow the computation of each single FFT term.

Example 120: Computation of FFT for $n = 4$ in general, and for the vector $g = (1, 1, -1, 1)$ in particular.

<table>
<thead>
<tr>
<th>$g = (1, 1, -1, 1)$</th>
<th>$f^{(0)} = (g_{0}, g_{1})$</th>
<th>$f^{(1)} = (g_{2}, g_{3})$</th>
<th>$f = \mathcal{F}^{-1}(g)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 -1 1</td>
<td>1 1</td>
<td>-1 -1</td>
<td>0 2 -2 0</td>
</tr>
<tr>
<td>$g_{0}$ $g_{1}$ $g_{2}$ $g_{3}$</td>
<td>$f^{(0)}(0)$ $f^{(0)}(1)$</td>
<td>$f^{(1)}(0)$ $f^{(1)}(1)$</td>
<td>$f(0)$ $f(1)$ $f(2)$ $f(3)$</td>
</tr>
</tbody>
</table>

To execute FFT, we start from vectors of unit length and in each $\ell$th stage, $s = 1, \ldots, 2^\ell$ put together $2^\ell$ vectors of length $2^\ell$ from vectors of length $2^{\ell-1}$, this costs $2^{\ell-1} \times 2^{\ell}$ products. Altogether, the cost of FFT is $2^{n-1} = 2^{p(\log n)} = p(n)$ products.
Example 3: Stability of PDE discretisations

Mathematical explanation

NA-webpage & MATLAB GUI
Conclusion after the 1st year of implementation

Was it a success? **Yes**
Conclusion after the 1st year of implementation

Was it a success? **Yes . . . at least partially.**
Conclusion after the 1st year of implementation

Was it a success? Yes ... at least partially.

- Clear enhancement of lectures by live demos
  ⇒
  MATLAB demos link theory to computations; student feedback positive.
Conclusion after the 1st year of implementation

Was it a success? Yes ... at least partially.

- Clear enhancement of lectures by live demos
  =>
  MATLAB demos link theory to computations; student feedback positive.

- Students did not use online material excessively
  =>
  Online interaction is not assessed; integrate the online material more into the course; in particular, through instructing students and supervisors; amending example sheets takes care and time; change format of use (GUI)?
Why MATLAB?

- MATLAB is a great software for a very intuitive way of programming.
MATLAB is a great software for a very intuitive way of programming. One of our developer students only had very basic programming skills when she started the job!
Why MATLAB?

- MATLAB is a great software for a very intuitive way of programming.
- Easy and flexible way of visualising results.
Why MATLAB?

- MATLAB is a great software for a very **intuitive way of programming**.
- Easy and flexible way of **visualising** results.
- It provides many **inbuilt functions** that facilitate computation. GUI’s easy to build.
Why MATLAB?

- MATLAB is a great software for a very **intuitive way of programming**.

- Easy and flexible way of **visualising** results.

- It provides many **inbuilt functions** that facilitate computation. GUI’s easy to build.

- The **publish option** in MATLAB connected the documentation of the GUI’s directly with the NA-webpage.
Effects so far

- Improves teaching experience.
Effects so far

- Improves **teaching experience**.
- Gives **fresh drive** into the otherwise dry NA theory.
Does it work?

Effects so far

- Improves **teaching experience**.
- Gives **fresh drive** into the otherwise dry NA theory.

- **Motivates students** to explore NA further: One of our NA students did a summer internship on a NA topic in Cambridge.
What’s the future?

- Evaluation by student questionnaire.
What’s the future?

- Evaluation by student questionnaire.
- GUI code is unreadable ⇒ Have more understandable MATLAB code in the future to facilitate student learning from provided implementations.
What’s the future?

- Evaluation by student questionnaire.
- GUI code is unreadable ⇒ Have more understandable MATLAB code in the future to facilitate student learning from provided implementations.
- Expand MATLAB material
  - by industrial/application examples (e.g. diffusion for medical imaging applications – image denoising)
- to Part III NA - this needs a more advanced student.
The End

Thank you for your kind attention!

For more details see

http://www.maths.cam.ac.uk/undergrad/course/na/

and

http://www.damtp.cam.ac.uk/user/cbs31/Teaching.html

or write to: cbs31@cam.ac.uk