komunikácie
COMMUNICATIONS

Michal Kaukič *

MATHEMATICS IN APPLIED INFORMATICS EDUCATION – NEW CHOICES AND CHALLENGES

In this paper, new ways of teaching Mathematics, which have been opened by the widespread of computers and notebooks and also by adequate software tools, are discussed. Based on our experience, we advocate the use of Open Source software as the mainstream tools in Mathematics and Informatics education. We give the more concrete examples of tools, used for teaching of Numerical Analysis topics, which can be successfully explored also for majority of other subjects in Applied Informatics and Mathematics. Some recommendations for the choices of suitable software and for further coordination in this field are given.

1. The traditional view of teaching Mathematics in Engineering environment

In Engineering education, there was traditionally a big difference between the way of teaching Mathematics and other, specialised subjects e.g. from mechanical, electrical, civil engineering or informatics.

The mathematicians like to expose the essence of Mathematics in the strictly logical form of reasoning using axioms, definitions an theorems, eventually with rigorous (or informal, but more intuitive) proofs whenever it can contribute to better understanding of explained concepts.

The engineers are more interested in the solution of real-world problems, using selected mathematical apparatus for formulation of (simplified) computationally tractable model and for solution of that model problem with the aid of some of available (or suitably modified) algorithms, usually only with certain (prescribed or estimated) accuracy.

Standing on this viewpoint, the practical engineers perceive many of traditional topics of algebra or calculus as more or less unnecessary or simply useless for engineering practice. On the other hand, mathematicians must follow the logical continuity and dependencies between introduced concepts. There is no way to fully understand the concept of Fourier transform without the basic knowledge concerning the definite integrals. In linear algebra, the basic concept is that of matrix. Without proper theoretical grounding (linear independence, determinants, inverse matrix, eigenvalues and eigenvectors, etc.) the students cannot fully understand and explore many of practical algorithms for solution of linear systems, optimisation algorithms such as simplex method, spectral analysis of linear processes, etc.

Two principal questions in this field are about teaching material – (what to teach?) and teaching methods – (how to teach?). In our opinion, the latter question is far more significant. In present, the concrete information we give to students is not so important as the message it cares about ideas, leading to some intellectual or practical achievements. *We should communicate ideas not the plain information.* Our students should be able to make individual progress in the process of complementing their existing skills and knowledge with new tools and information as demanded by needs of their professional career.

Thus, the common point for both mathematics and engineering educators is to prepare students to *independent, logical, clear and continuous thinking.* It is not sufficient to give the students many (maybe, highly practical) isolated pieces of information in the cookbook style. We should bring to the students the feeling of beautiful integral view of *The Forest of Science,* not to walk with them in the darkness of labyrinth of individual trees. If such walk is necessary, then it should be not too long and lead to a peak with far, clear views.

We (the communities of mathematicians and engineers) should choose the adequate tools to be in state of resonance and to support each another, not to make the useless remarks about (supposed) weak points of either "engineering" or "mathematical" mode of thinking. In this paper, we will report our experience with computer assisted teaching of some topics in Numerical analysis and some negative habits observed in students' behaviour.

2. Dangers coming from mindless use of computers

With the wide accessibility of computers, there is the growing principal danger of overestimating their role in education and in the engineering practice. In the naive students' perception the computers with highly intelligent software can be used as substitute for the (painful) process of creative thinking. But also on the teacher's side there are some overly optimistic expectations from using tools like LMS (Learning Management Systems, e-learning).

* **Michal Kaukič**
Department of Mathematical Methods, Faculty of Management Science and Informatics, University of Zilina, Slovakia
E-mail: mike@frcatel.fri.utc.sk

The common sense, of course, tells us that nothing can replace the individual creative thinking and that the personality of teacher will always play the key role in the process of knowledge, intuition and skill transfer to the students.

Let us bring the quotation from SEFI (European Society for Engineering Education) document [1]:

There are some signs that the involvement of computers in the teaching of undergraduate engineering mathematics is beginning to gain momentum. However, the experience of the last thirty years warns that care must be taken. When the pocket calculator arrived we were told that there would be two advantages: first, the students would be relieved of hours of tedious calculation, leaving them free to concentrate on concepts and understanding; second, it was more likely that the calculations would be performed correctly. The reality is somewhat different. The most trivial of calculations is often subcontracted to the machine, the students have little or no feel for what they are actually doing or to what precision they should quote their answers. What they can do is to obtain obviously unrealistic results more quickly and to more significant figures. There has also been a tendency to replace analytical reasoning by trial-and-error methods.

This is the clear analysis of dangers, caused by unqualified use of computers. We should bear this in mind and to create the environment where computers are helpful but where critical, logical thinking is all the time at the first plane. In the next section we will show an example of such environment, we now use for the teaching of Numerical analysis.

3. Pylab – the environment for experimenting and visualization of Numerical analysis concepts

Effective teaching of Numerical Analysis is closely related to the problem of using computers as the laboratory, experimental equipment, in the sense which is well known from other natural sciences. Numerical analysis must take into account the real world imperfections, which can be happily ignored by theoreticians. Thus, in the field of Numerical analysis there is a big room for experiments. We believe that numerous computer-based experiments are essential for successful teaching of Numerical Analysis.

The choice of appropriate software tools is of the fundamental importance. This choice, once made, determines the effectiveness and long-term maintainability of educational and research activities for many future years.

On the first look, we will probably find three well-known commercial software systems for mathematical teaching and research: MATLAB, Mathematica, Maple.[1)]

Besides of certain advantages, commercial software systems have also many drawbacks (for more detailed discussion, see [2]). Our search for alternative software leads us to several Open Source systems with MATLAB-like capabilities. The most MATLAB-compatible among them was Octave [3]. In the course of using MATLAB and later Octave we constantly perceived the apparent deficiencies of underlying simple programming language. Moreover, programming in MATLAB can be done in very bad style by inexperienced students not encouraging them to learn new ways of thinking about problems. The MATLAB language simply is not general enough to be useful outside of its primary domain - matrix and vector manipulations.

Although there is no universal system suitable for all areas of mathematical education, we can try to minimise the number of different tools used. The choice should be made so that students can reuse our tools in possibly widest area of their future professional career (having in mind preferably the students of Informatics).

This brings us to the idea of using some of general purpose programming languages. The language of our choice should allow to express mathematical ideas with minimum programming effort and in "nearly mathematical" notation.

In present, we use the Pylab environment, comprised of interpreted language Python [4], complemented by user-friendly interactive environment (IPython shell, [5]), modules for linear algebra and numerical analysis (see [6], [7]) and also module Matplotlib [8] for excellent quality two-dimensional graphics. We will show the typical use of this environment on the following example.

Example 3.1 *Find positive solutions (i.e. with positive components) of the system of nonlinear equations*

$$\sin(xy^2) - \cos(x^2 - y) + 0.2 = 0$$

$$x^3 + y^3 - 3xy = 0$$

First, we can give the nice graphical interpretation for this example. The solutions will be exactly the common points of zero contour levels of surfaces $z_1(x, y)$, $z_2(x, y)$, given by left sides of equations, i.e.

$$z_1(x, y) = \sin(xy^2) - \cos(x^2 - y) + 0.2,$$

$$z_2(x, y) = x^3 + y^3 - 3xy$$

The corresponding plot will be generated by the commands (exactly the same, as it would be in MATLAB):

```
xx=linspace (-2, 2, 120);   yy=linspace (-2, 2, 120)
X, Y=meshgrid (xx, yy)
```

[1)] Maple, MATLAB, Mathematica are registered trademarks of their respective owners (Waterloo Maple Inc., The MathWorks, Inc., Wolfram Research, Inc.). Other trademarks mentioned in this article are registered trademarks of their respective owners too.

```
Z1=sin(X*Y*Y)-cos(X*X-Y)+0.2;     Z2=X**3+Y**3-3*X*Y
contour (x, Y, Z1, [0.0])
contour (x, Y, Z2, [0.0],colors='g')
```

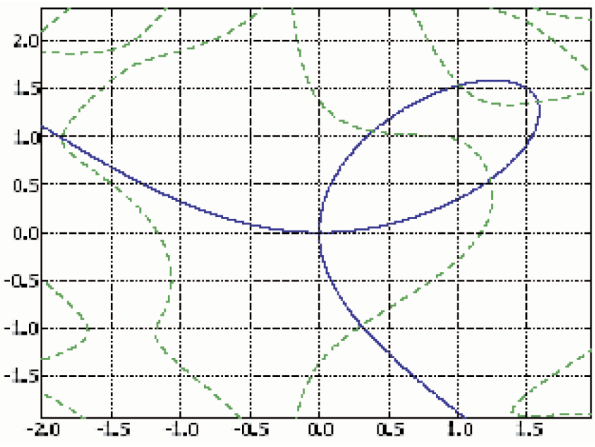

Fig. 1 Solution of system nonlinear equations

From this figure we can see that there are exactly four positive solutions. We can zoom the figure window and to make the good initial guesses for solutions:

```
(0.38, 1.06), (1.23, 0.55), (1.02, 1.54), (1.58, 1.36)
```

For the numerical solution of nonlinear systems we can use the function `fsolve`. The first argument of this function is the vector function, describing the given nonlinear system – the input is in our case the tuple (x, y) and the function returns the tuple of values ($z_1(x, y)$, $z_2(x, y)$). The code for this function is as follows (we can save it to the file `nlfct.py`):

```
def nlfct(xy):
x,y = xy # xy has two components, separate them
z1=sin(x*y*y)-cos(x*x-y)+0.2
z2=x**3+y**3-3*x*y
return (z1, z2)
```

The second argument for fsolve is the initial approximation of solution. Thus, we can get the first solution in interactive IPython environment by commands:

```
run nlfct
from scipy.optimize import fsolve
r=fsolve(nlfct, (1.23, 0.55))
# The answer is: r=(1.2328735, 0.5521787)
```

As always, we cannot blindly trust the results, we get from computer. But we can easily verify that the command `nlfct(r)`, which computes the values of functions $z_1(r)$, $z_2(r)$ returns very small values (approximately $-4.552 \cdot 10^{-15}$, $-3.997 \cdot 10^{-14}$. The same is true for remaining three solutions, but we will not repeat the calculations here.

We can see that Pylab is, indeed, the very high level language – no type declarations of variables and functions, no troubles with memory management. There are many ready-made useful functions for numerical analysis (data interpolation and approximation, linear algebra functions – eigenvectors and eigenvalues, determinant, inverse matrix, solution of linear equations, also in least-squares sense, numerical integration, optimisation with bound constraints, solution of differential and differential-algebraic equations and many others). Pylab has also many functions for statistics and probability theory. There is also a signal processing toolbox analogical to that of MATLAB.

Using additional Python modules, we can easily add e.g. the SQL DBMS interface for working with commercial (Oracle) or Open Source (PostgreSQL, mysql, SqLite) SQL databases, there are other modules for discrete and continuous simulations, linear and, more general, convex programming.

Now, we can summarise some experience we gained in teaching Numerical Analysis with Pylab. The very positive thing is the possibility of rapid prototyping and interactive debugging of little scripts. We have prepared the material about basic programming in Pylab (see [9]), which was sufficient for students to start working in the interactive environment. Unfortunately, many students have strange and non-productive habits of programming.

IPython environment has excellent command completion, but students prefer the (erroneous) lengthy typing. Or, there is no need to position the cursor on the end of line before entering the command, but they have constantly made this action. IPython saves command line history, so any previous commands can be conveniently re-entered or edited, but students like to type the same thing many times, instead of using this feature.

The biggest misbehaviour from students' side is the programming without thinking. They start typing something, having little understanding of the given task and the methods they will use for solution. They put emphasis on programming routine, but not on "The Art of Computer Programming" as seen by the classical works of D. Knuth [10]. We have the strong opinion that for beginners the clear and simple programming language should be used. Neither Pascal, nor C/C++ or Java can fulfil these basic requirements. We propose to try (at University level) to unify the programming tools for basic courses on algorithmization and programming using Python-based environment.

The considerable amount of work should be done for analysis of existing tools and environments (especially non-commercial, OpenSource alternatives), the choice of basic software for applied informatics education and the customisation of chosen software for the specific needs of faculty and university. We think there should be first a broad discussion at inter-faculty level and afterwards the work group responsible for the implementation of decisions coming from this discussion should be formed. The process of implementation will, of course, take several years to be reasonably complete.

4. Conclusion

We hope that the reader got some feeling of what can be done with newly available Open Source software tools in the field of mathematical education at University level. The cultivation of mind in Mathematics and in Computer Science can be done in close cooperation. This can lead to the synergy effect of better understanding of the common points and the role of clear, logical, independent thinking in all applied informatics activities. This is especially important for making right key decisions, which will be crucial for the future of the fields of Applied informatics and Mathematics at our University and in our country.

We saw the example of successful use of user-friendly environment, based on Open Source tools in teaching of Numerical Analysis. This environment, in our opinion, has the great potential of becoming the tool of choice for nearly all educational topics in traditional and modern branches of Mathematics (algebra, calculus, probability and statistics, graph theory, mathematical programming and optimisation, cryptography, numerical computations, computer graphics, computational geometry, etc.) and also for many important concepts of Applied Informatics (the basic principles of algorithmization, data structures, discrete and continuous simulation, stochastic processes, signal processing, parallel computations, etc.).

For optimal results in applied informatics education (bachelor, undergraduate, graduate and doctoral study) we cannot use the multitude of software tools without any coordination. There should be clearly defined "central path" software, which will be used continually during all the time of study. We are convinced that the right choice for this durable software tool cannot be based on traditional, low level languages like Pascal, C/C++. The good candidate should be very high level language with clear, simple syntax and high expressive power – at present, the language of our choice is Python with suitable modules. Using it has also the additional benefit of learning a simple, useful general-purpose language, which students can use later in their career (important not only for students of Computer Science but also for general engineering students and mathematicians, too).

References

[1] SEFI Math. working group: *Mathematics for the European Engineer,* SEFI HQ, 2002
[2] KAUKIČ, M.: *Open Source software resources for numerical analysis teaching,* International Journal for Mathematics Teaching and Learning, CIMT, Univ. of Exeter, Exeter, Oct. 2005, ISSN 1473-0111, electronical form, 9 pages
[3] EATON, J. W.: *Octave Manual, available on http://www.octave.org/docs.html*
[4] VAN ROSSUM, G.: *Python Documentation, on http://www.python.org/doc*
[5] PEREZ, F.: *IPython – An enhanced Interactive Python,* available from http://ipython.scipy.org
[6] ASHER, D., DUBOIS, P. F., HINSEN, K., HUGUNIN, J., OLIPHANT, T.: *Numerical Python,* available on *http://numpy.sourceforge.net*
[7] SciPy Devolopers: *SciPy,* available on *http://www.scipy.org*
[8] HUNTER, J. D.: *Matplotlib*, available on *http://matplotlib.sourceforge.net*
[9] KAUKIČ, M.: *Essentials of Pylab Programming, (in Slovak),* draft of textbook in preparation, can be downloaded from *http://frcatel.fri.utc.sk/mp2.pdf*
[10] KNUTH, D.: *The Art of Computer Programming,* vol. 1, Addison-Wesley, 1969.