Global J. of Engng. Educ., Vol.9, No.1
Published in Australia

Challenges in Computer Mathematics in Engineering Education*

Dieter Schott
Hochschule Wismar – University of Technology, Business and Design
Phillipp-Müller-Straße 14, D-23966 Wismar, Germany

Identifying the position that mathematical software systems, with their components (numerical and symbolic computation, graphic representation and experimentation), should hold in the education of engineering students is discussed in this article. Its convincing advantages are more sophisticated and practically significant examples, a release from trivial calculations and transformations, and the possibilities for experiments and the inclusion of computer science parts. However, there are also some risks, such as neglecting mathematical knowledge, blind confidence in computer outputs, missing the *feel* for the correctness or misinterpretation of computer results and the inability to interactively interfere in the case of unsatisfactory results. The case of selecting *MATLAB* used to illustrate typical examples is presented and discussed in this article. The dilemma is that although the mathematical knowledge of many beginner students nowadays is significantly sub-standard, the reasonable integration of mathematical software systems in the general concept of engineering education demands a solid knowledge of mathematical basics. Therefore, modern mathematics in engineering education has to include computer mathematics, but need not be reduced to it. Identifying the right composition of both parts is crucial; indeed, meeting the optimum level is an art.

INTRODUCTION

The value of mathematics is controversial, although its achievements and potencies are huge. The bad image of mathematics in public, and the lacking motivation of many young people to acquire a good level of knowledge in this science, will generate, sooner or later, a very critical situation in education if no re-evaluation is forthcoming.

The *Gottlob Frege Centre for Engineering Science and Design* (GFC), which is based at Hochschule Wismar - University of Technology, Business and Design (HSW), Wismar, Germany, has acted since its foundation in 2000 to propagate mathematics as a key qualification in engineering education and to increase its starting level at universities [1-3]. It should be noted that the GFC is also a satellite centre of the UNESCO International Centre for Engineering Education (UICEE), which is located at Monash University, Melbourne, Australia.

Present public activities are being developed as a response to depressing international comparative studies, such as TIMMS or PISA [4]. Such activities react in steadily changing chaotic small steps within the framework of party interests. Each regional minister in Germany is hunting for a *new cow*, further destabilising the structure and increasing the uncertainty of the educational landscape. Hence, the GFC's contribution to bringing up-to-date training in mathematics is weakened by these conditions.

Undoubtedly, the discussion about the value of computer mathematics in education is being strongly influenced by the role of given mathematics in society. Conservative thinkers hate the use of computers in mathematical training, arguing that classical mathematical techniques would be neglected. Yet pseudo-modern pragmatic thinkers love it because the software does the work and the user seems to need no mathematical skills at all. Both are not right.

*A revised and expanded version of a lead paper presented at the *8th Baltic Region Seminar on Engineering Education*, held in Kaunas, Lithuania, from 2 to 4 September 2004. This paper was awarded the UICEE diamond award (joint first grade with one other paper) by popular vote of Seminar participants for the most significant contribution to the field of engineering education.

MATHEMATICS AND SOFTWARE TOOLS

It is interesting to study the role of tools in mathematics. *New mathematical tools influence the methodology and the contents of mathematical training*. Previously, engineers used known mathematical theories to solve their sophisticated problems. Therefore, they had to absolve hard mathematical training.

Nevertheless, such tools, like slide rules or pocket calculators were used to reduce mechanical headwork. Each time, there were discussions whether these tools were helpful for mathematical training or not. But new technologies, which increased the computation power and reduced boring mechanical work, have always succeeded. Responsible training should be orientated *to hold corresponding computation skills principally present but to transfer them to mathematical tools for efficiency*. Consequently in mathematical training, *understanding the principle of a method becomes more important than corresponding calculation and transformation skills.* Yet tools also influence the solution methods utilised. As such, it is natural to treat mainly the principles of methods applied in computers on a reasonable level.

New mathematical tools offer new perspectives, but also new risks. It is fascinating to see which results can be reached by mathematical computer software. But, who ensures that the results are correct? Namely, there are many sources of error (eg input errors, model errors, approximation errors, rounding errors, error propagation, misinterpretation of results, etc).

If strong mathematical tools are utilised in engineering practice without a principal understanding of computation processes and without any verification of the results, the consequences can be tragic. In each case, it is important that the user of tools is able to estimate or test whether the results correspond to the general experiences or lie within the limits known from the theory. So as to avoid or to find errors, it is advisable to also check each step.

For example, when calculating any probability, the value has to be in the interval [0,1]. When calculating the value of a well-known function (eg sine or exponential), it has to lie in the well-known range of the specific function.

Another example may involve having to determine the height of a tower in physics using freefalling bodies. Then one can imagine the real height. Should the difference between the expected and the calculated height be too great, then one has to check if something is wrong.

Yet another example is when there are ill-posed problems and solution methods that are unstable under certain conditions. These phenomena can cause useless results. If a non-standard problem is solved by computer software, the user should check if such dangerous effects could also occur. At minimum, the result should be verified via experiments or other means.

Nowadays, there are powerful software tools that have only to be fed with the physical description of a model before providing the solution in a short space of time, including colourful graphical representations. *Greenhorns* then become very enthusiastic because mathematics will not stress their great mood. Moreover, since the performance of such tools is fantastic and will increase further and further, the solution of scientific problems seems to be very easy. The successful blind application of these tools in various standard situations will strengthen such perceptions without affecting the dangerous illusion. Sometimes, there is a painful awakening. There are horrible scenarios that have already become reality: buildings can collapse by errors in static computations, machines can explode by the use of unstable numerical methods for design, and people can die as a consequence. In such cases, first-class examining commissions should be appointed and famous lawyers then enter the scene to work for years, thereby consuming a lot of money. Press campaigns may then be started to determine the reasons for the faults. Finally, someone has to take the blame. However, the author remains sceptical if there would be much interest in demanding improvements in mathematical education in order to reduce the risks generated by the ill use of mathematical software and to save money and people's lives.

Software tools are *black boxes* for the user that contain complicated expert systems. One can start with mathematical models that are solved by professional, analytical or mostly numerical methods. Thus, it is hidden that the kernels of these software systems consist of high performance mathematics. However, there are no *perfect* systems; there are no universal solution methods. The limits of such methods have to be known because, in certain situations, they can fail. In the case of numerical methods, error propagation can generate totally false results. Furthermore, the software realisation may have some weaknesses because software products become increasingly complex and there is considerable competitive time pressure to release the software as early as possible to the international market.

Most developed software tools start with physical models that are internally transformed into mathematical models. Therefore, the user has to know something

about the mathematical and physical backgrounds in order to avoid mistakes or even catastrophes. *In critical cases, results have to be verified by applying different solution methods and/or practical experiments*.

The conservative answer to the mentioned risks is to go back to classical teaching and to ban computers from mathematical teaching. Yet this is highly counterproductive, since computer tools increase performance immensely. Certain important practical problems are only solvable with computers.

At some point in their careers, graduates will have to work with mathematical software and will need to acquire skills in their use and learn a reasonable relation to this tool. Nobody assumes that the number of hours for mathematical training will essentially increase, since demands in other subjects grow and new subjects, like business and social skills, also occur. Therefore, the time for the training of computational skills has to be reduced in order to gain time to acquire qualitative knowledge. *Mathematical education has to consider the optimal mixture of software mastery and mathematical background knowledge* [5-7]. An article by Schott contains some instructive examples to illustrate the statements [7].

SOLUTION OF LINEAR EQUATION SYSTEMS

Let us start with linear systems of equations in order to explain some challenges concerning the use of mathematical software. This equation type occurs again and again in mathematics and in practical applications [5][8]. One example is also mentioned in the next section [8]. Key facts should be known when solving such a system of m linear equations with n unknowns, in matrix-vector-notation written as A x = b. These facts are as follows:

- The system may have no solution, a unique solution or infinitely many solutions (described by using a number of parameters according to the rank deficiency n – rank A).
- In the *classical case*, there are as many unknowns as equations that are linearly independent (m = n and det A ≠ 0); this leads to a unique solution.
- The *general solution* of a consistent system A x = b is composed by a special solution and a linear combination of (linearly independent) basic solutions of the corresponding homogeneous system, A x = 0.
- *Generalised solutions* exist for an inconsistent system (in the sense of least-squares-approximations). They coincide with the common solutions for a consistent system.
- The least-squares-solutions of the system are the solutions of a transformed consistent linear system (system of Gaussian normal equations); therefore, the structure of these pseudo-solutions is the same as stated before.
- Errors arising from measurements, models or numerical computations on a computer can distort the results or the structure of the linear system. This is to be considered especially if the system is *ill-posed*. Then *regularisation* techniques replacing the system by a well-posed neighbouring system are used to reduce the errors.

Some concepts and theory are necessary for a correct solution. The theory is quite simple compared with the theory of non-linear systems of equations. Nevertheless, students often have trouble remembering mathematical facts. They have also trouble solving systems matrices greater than 3 x 3 without computers, producing false results as a consequence of mistakes or ignoring the solution structure in the case of infinitely many solutions. So a computer system can be very helpful. However, the following example shows that there can occur other difficulties.

Example: Solve $Ax = b$, where

$$A = \begin{pmatrix} 14 & -4 & 10 & -6 \\ -4 & 5 & -5 & 6 \\ 10 & -5 & 11 & -4 \\ -6 & 6 & -4 & 10 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

The numerical computer solution is detailed below. The following signs and structure are used to record computer calculations:

```
>> user input              % user comments
   computer output
```

The *standard solution method* (Gaussian algorithm with column pivot strategy to reduce numerical errors) is as follows:

```
>> x1 = A \ b          % backslash as left division by A
   x1 = 1.0e+015*      % huge pre-factor 10^15
        –0.9007        % vector coordinates
         2.2518
         1.3511
        –1.3511
```

Although there is no computer warning, the huge pre-factor should lead to a critical check.

The *inversion solution method* using the explicit

representation x = A-[1] b , including the matrix inverse A-[1] (computed by Gaussian algorithm) is as follows:

```
>> x2 = inv(A) * b        % inv(A) means the inverse of A
   x2 = 1.0e+015 *        % huge pre-factor 10^15
         –0.9007          % vector coordinates
          2.2518
          1.3511
         –1.3511
```

There seems to be no difference to the standard solution method. The separate computation of the inverse supplies a further hint to doubt the correctness of the result:

```
>> IA = inv(A)
```

Warning: the matrix is close to singular or badly scaled. The results may be inaccurate. RCOND = 4.168547e-018.

$$IA = 1.0e+015 * \begin{pmatrix} 0.6005 & -1.5012 & -0.9007 & 0.9007 \\ -1.5012 & 3.7530 & 2.2518 & -2.2518 \\ -0.9007 & 2.2518 & 1.3511 & -1.3511 \\ 0.9007 & -2.2518 & -1.3511 & 1.3511 \end{pmatrix}$$

The extremely small number RCOND shows that the problem is ill-posed. The rule of Cramer states that the solution of the linear system can be expressed with a matrix determinant in the denominator. Hence, it can be guessed that the huge numbers are caused by a determinant value that is nearly zero. The check essentially confirms the guess:

```
>> DA = det(A)            % determinant of A
   DA = 0
```

It should be noted that this is in numerical mode. Thus, the result means that the determinant is zero or nearly zero. However, only in the second case is there a unique solution. There are further possibilities to check the result, namely:

- Measuring the equation defect of the results (in Euclidean norm) which has to be zero for a solution:

```
>> d1 = norm(A*x1–b)
   d1 = 3.4641
>> d2 = norm(A*x2–b)
   d2 = 2
```

  There are two findings. First, the great differences from zero show that the results are useless. Second, although the output form of both results is the same, there should be slight differences caused by the use of different numerical computation methods.
- Rank investigations which can also be done before computation of a solution:

```
>> rA =rank(A)
   rA = 3
>> rAb = rank([A b])
   rAb = 4
```

  The different ranks show that there is *no solution*. The difference n – rank A = 4 – 3 = 1 means that we have a one-parametric general least-squares-solution.

The *pseudo-inversion solution method* generates the special pseudo-solution x = A[+] b , where A[+] is the pseudo-inverse of A. Further, the null space N(A) of A containing the solutions of the homogeneous system has to be determined in order to obtain the general least-squares-solution. The procedure is as follows:

```
>> xs = pinv(A) * b    % pseudo-solution of minimal norm
   xs =
         0.1419
         0.1174
         0.0545
         0.1556
>> NA = null(A)        % special null space basis
   NA =
        –0.2917
         0.7293
         0.4376
        –0.4376
```

The least-squares-solutions manifold is now:

$$x = \begin{pmatrix} 0.1419 \\ 0.1174 \\ 0.0545 \\ 0.1556 \end{pmatrix} + \lambda \begin{pmatrix} -0.2917 \\ 0.7293 \\ 0.4376 \\ -0.4376 \end{pmatrix}.$$

The parameter is denoted here by λ. The pseudo-solutions produce the *minimal defect* in the linear system of equations in the least-squares-sense. Geometrically, the defect vectors then have the shortest length. The following is obtained:

```
>> ds = norm(A*xs–b)
   ds = 0.4376
```

Hence, the defect norm here is quite smaller than in the case of the useless results given before.

IMAGE RECONSTRUCTION

If a planar body slide is penetrated by X-rays, the radiation intensity I is weakened depending on the local densities of the body material. Since the densities correspond to the body structure, this process can be utilised to detect this structure by intensity measurements. This method is often called *image reconstruction*, or for human bodies, *computer tomography*. Following the physical law of weakening under some simplifying assumptions, we obtain a linear integral equation of the form:

$$\int_L f(x)\,dx = g(L)\,,$$

where the straight lines L are the paths of X-rays, the function f(x) is the unknown density of the slide in the points x and the function g(L) is obtained from the intensities of the rays before and after penetrating the slide. If a square region Q containing the slide is divided in such a way into n small squares (pixels) Q_j that the densities in Q_j can be supposed to be constant, then for a sample of m rays L_i the following system of linear equations arises:

$$\sum_{j=1}^{n} a_{ij} \cdot f_j = g_i \quad (i = 1,...,m)\,,$$

where:

$$a_{ij} = l(L_i \cap Q_j), \quad f(x) = f_j \ (x \in Q_j), \quad g_i = g(L_i)$$

are the intersection length of the straight lines L_i in the squares Q_j, the unknown density values and the intensity functions, respectively. This system of m equations with n unknowns can be consistent or inconsistent, depending on the ray geometry and the pixel division. Also, the system is often ill-posed. So the whole theory mentioned in the preceding section is necessary in order to obtain discrete solutions f_j. By interpolation, continuous or even smooth solution functions f(x) can be generated. It is a true challenge for students to develop a complex *MATLAB* software package realising the reconstruction of objects in a satisfactory way.

Image reconstruction is an exciting example to demonstrate the mathematical solution of practical problems [8]. The mathematical model is derived (linear integral equation) starting from the physical model. The continuous model is approximated by a discrete model (system of linear equations) in order to solve the problem numerically on a computer. Some known solution methods can be tested so as to choose the best.

The reconstruction f(x) can be visualised by a surface in the space or by a planar level plot. Given phantom objects can be used to study the quality of reconstruction. For unknown objects, the verification procedure will include error analysis and the check of being consistent with known facts and theory.

THE PENDULUM PROBLEM

A further interesting and simple problem, relevant to engineering education, is the motion of a pendulum. Starting with a physical model, the student arrives at a simple mathematical model, assuming that the mass of the pendulum is concentrated in a point at the end of an infinitely thin thread.

In this example, the motion of a (mathematical) pendulum satisfies approximately the model:

$$\varphi''(t) + c \sin\varphi(t) = 0, \quad \varphi(0) = \varphi_0, \quad \varphi'(0) = \varphi_0'$$

(with given φ_0 and φ_0') if the air resistance (friction) is neglected. Here $\varphi(t)$ is the angle swing at the moment t, with c being a constant depending on the Earth's acceleration g and the length l of the pendulum.

It is important to understand the following:

- The model is a *differential equation of second order* complemented by two *initial conditions* (a so-called initial-value-problem).
- The differential equation is *non-linear*, caused by the non-linear sine function applied to $\varphi(t)$. However, the equation can be considered to be linear if there are only small swings of the pendulum ($\sin\varphi \approx \varphi$ if $|\varphi|$ is small).
- In the linear case, the solution can be given exactly. It is a periodic trigonometric function. In the non-linear case, numerical methods deliver approximate solutions.
- The two initial conditions are necessary in order to make the solution unique.

Naive *discretisation* of the problem using times t_i (i=0,...,n) with initial time $t_0 = 0$, small constant *time steps* h > 0 and first and second order difference quotients, instead of first and second order differential quotients, leads to a simple non-linear system of equations that can be solved by forward substitution of the stepwise calculated discrete unknown angle swings denoted by $\Phi_i \approx \varphi(t_i)$ (i=0,...,n):

$$\Phi_0 = \varphi_0, \quad \Phi_1 = \Phi_0 + h\varphi_0',$$
$$\Phi_{i+1} = 2\Phi_i - \Phi_{i-1} - ch^2 \sin\Phi_i \quad (i = 1,...,n-1).$$

The result yields discrete points (t_i, Φ_i) that can be interpolated by a continuous function $\Phi(t) \approx \varphi(t)$ in the interval [0,nh].

It is easy to produce a *MATLAB* file that

calculates and visualises the approximate swing curve of the pendulum. Experiments show that the solution of the given pendulum model is described accurately enough only if the time step h is sufficiently small compared with the length l of the pendulum. But choosing h too small will increase the errors again by numerical effects. This phenomenon is called *semi-convergence*. Hence, a good choice of h needs to be identified. Professional software will use combined methods of a Runge-Kutta type with variable time step control. In *MATLAB*, the standard functions ODE23 or ODE45 can be utilised if the differential equation is described in a corresponding file. But if the differential equation problem is hard enough, these methods can also produce useless results.

In some sense, the model for the pendulum is not adequate. So the influence of friction (nearly proportional to $\varphi'(t)$) and the impact of external forces f(t) can be considered as follows:

$$\varphi''(t) + b\varphi'(t) + c\sin\varphi(t) = f(t), \quad \varphi(0) = \varphi_0, \quad \varphi'(0) = \varphi_0'.$$

With small changes, the *MATLAB* files can be used again in order to solve the problem. It is easy to produce instructive graphical representations in *MATLAB*.

INFINITE SERIES

So far, numerical calculations have been used. It is now shown that *symbolic calculations* can also produce some difficulties. This is illustrated in the case of infinite series, as follows:

$$\sum_{k=0}^{\infty} a_k = a_0 + a_1 + ... + a_n + ...$$

- The series can be given the finite value *a* if the sequence s_n of partial sums is convergent to *a*:

$$\lim_{n\to\infty} |\, s_n - a \,| = 0, \quad s_n = \sum_{k=0}^{n} a_k .$$

This series is called *convergent*.

- If the partial sums increase or decrease beyond all limits, the series will get the value $+\infty$ and $-\infty$, respectively. Such a series is called *definitely divergent*.
- Otherwise, the series is called *indefinitely divergent*. But using a mean value method you can give some of such series a generalised finite value *a* if the mean values t_n of the partial sums converge to *a*:

$$\lim_{n\to\infty} |\, t_n - a \,| = 0, \quad t_n = \frac{1}{n}\sum_{k=0}^{n-1} s_k .$$

These series are said to be *generalised convergent*.

By symbolic calculation we can obtain the following results:

$$\sum_{k=1}^{\infty}\frac{1}{k} = +\infty, \quad \sum_{k=1}^{\infty}\frac{1}{k^2} = \frac{\pi^2}{6}, \quad \sum_{k=1}^{\infty}\frac{1}{k^n} = \zeta(n).$$

The first series is the so-called *harmonic series*, which is known to be definitely divergent. It should be noted that this result cannot be obtained through numerical computer calculations, which terminates with a finite value in each case, depending on the computing accuracy. The third series supplies values of the famous Riemann zeta-function. It is worth noticing that the first two series are special cases of the third. The comparison leads to the following:

$$\zeta(1) = +\infty, \quad \zeta(2) = \frac{\pi^2}{6}.$$

The first result shows that the Riemann function has at 1 a pole (point of infinity). These results are obtained through the following commands:

```
>> syms k n;                    % symbolic variables
>> s1 = symsum(1/k,1,inf)       % symbolic sum
   s1 = inf                     % inf means infinity
>> s2 = symsum(1/k^2,1,inf)
   s2 = 1/6*pi^2
>> s3 = symsum(1/k^n,k,1,inf)
   s3 = zeta(n)                 % Riemann zeta-function
```

For the first two sums, the running variable k need not to be indicated separately in the commands after the expression of members because there is no danger of misunderstanding. However, for the third sum, there are two variables, k and n. Therefore, the command contains the specification of k as a running variable. If this specification is ignored, an error message could be expected. But, surprisingly, the following result is given:

```
>> s4 = symsum(1/k^n,1,inf)
   s4 = 1/k-1)
```

A user with some basic knowledge will then recognise that this is the value of the convergent geometric series:

$$\sum_{n=1}^{\infty}\frac{1}{k^n} = \sum_{n=1}^{\infty}\left(\frac{1}{k}\right)^n = \frac{1}{k} + \frac{1}{k^2} + \frac{1}{k^3} + \frac{1}{k^4} + ... \quad (k = 2,3,...).$$

In the case of several variables, *MATLAB* chooses the running variable by taking the variable name with the highest order position in the alphabet. Nevertheless, the question arises for which real k the result is true.

The answer is | k | > 1, which follows on from the theory but not from the expert system *MATLAB*. Almost all members of the given geometric series are greater than the members of the series:

$$\sum_{k=1}^{\infty}\frac{1}{k^k}=1+\frac{1}{4}+\frac{1}{9}+\frac{1}{256}+...$$

It follows on from the theory that the latter series has to be convergent. So the user hopes to obtain the value from *MATLAB*. However, *MATLAB* returns the input since it has no result:

```
>> s5 = symsum(1/k^k,1,inf)
   s5 = sum(1/(k^k), k = 1 .. inf)
```

Even the simple series below:

$$\sum_{k=0}^{\infty}(-1)^k=1-1+1-1+...$$

can cause some confusion. The series has no value and is indefinitely divergent since the partial sums alternate from 1 to 0. However, the series has the generalised value ½, since the mean values of the partial sums are:

$$t_{2n}=\frac{n+1}{2n+1},\quad t_{2n+1}=\frac{1}{2}.$$

This sequence has the limit ½. Now, *MATLAB* is used:

```
>> s6 = symsum((-1)^i,0,inf)
   s6 = 1/2
```

Students usually know only elementary convergence theory from the lectures or textbooks (without generalised values and convergence). If they utilise *MATLAB*, they will perhaps come into conflict regarding whether to believe the professors or the *MATLAB* experts. Students will probably see no conflict. If so, they will take the *MATLAB* result as the value in the classical, instead of the generalised, sense. If students use the variable i instead of k in the foregoing *MATLAB* commands concerning series, they will obtain partially wrong results. The cause for this is that i is predefined as an imaginary unit in the context of complex numbers. Hence, ignorance can be dangerous.

SUMMARY

The use of mathematical software increases the demands upon teachers and students. The student can make a lot of errors using computer software. The software system involves a certain degree of imperfection. Therefore, the user must be able to check the results (or to delegate the check in critical cases to experts). Computer mathematics needs both knowledge of mathematical software systems (syntax and semantics of problem language, potential of commands, and dialogue management) and mathematical knowledge (correctness and interpretation of results). This is a dilemma because it has been observed that beginner students suffer from a decrease in mathematical competence plus severe deficiencies in elementary mathematics [2][3]. This means that educators are busied reducing deficiencies instead of teaching up-to-date mathematical methods. The level in mathematics has been reduced. The gap between entry-level students' knowledge and the requisite level for higher mathematics needed in other engineering disciplines (electrical engineering, technical mechanics, mechanical engineering, communication or automation engineering) is becoming great. As a consequence, these subjects are also developing to be a horror for students. The alternative is to reduce mathematical demands in these subjects, which leads to vocational training instead of scientific education. This perspective is not desirable. The following handicaps can be stated:

The theory, models and methods used in mathematical software are, for the most part, not (or only superficially) treated, or else not understood in the mathematical training of engineering students at universities. Examples of this include Fourier series, Fourier transformation, Eigen-value problems, finite-element-methods (FEM), generalised functions and variation principles. *Some important physical phenomena are not taught nowadays at universities because the mathematical basics are missing.*

It is the author's opinion that these conditions are not tolerable.

CONCLUSIONS

The following conclusions have been reached:

- Mathematical training for engineers has to be realised on a high level, combining suitable mathematical theory and methods with the use of mathematical software tools [5]. Changes of contents and training in mathematics are necessary. The student must know both the new perspectives and the new risks of using mathematical software tools.
- An offensive campaign is necessary to explain the key position of mathematics for further development [2].
- Education experts in the political field, the broad public and students have to be informed about the consequences of bad mathematical knowledge for the future [3].
- The author has found that this previous step has

to be complemented by resolute pressure to force reasonable decisions.

- The motivation of students is becoming increasingly important in order to achieve educators' aims.

REFERENCES

1. Grünwald, N. and Schott, D., Gottlob Frege Centre for Engineering Science and Design (GFC). *Global J. of Engng. Educ.*, 8, **1**, 53-64 (2004).
2. Grünwald, N., Kossow, A. and Schott, D., Mathematics – key-qualification in engineering education. *Proc. 3rd Annual Conf. on Engng. Educ.*, Hobart, Australia, 37-41 (2000).
3. Grünwald, N. and Schott, D., World Mathematical Year 2000: challenges in revolutionising mathematical teaching in engineering education under complicated societal conditions. *Global J. of Engng. Educ.*, 4, **3**, 235-243 (2000).
4. http://www.pisa.oecd.org
5. Schott, D., *Ingenieurmathematik mit MATLAB, Algebra und Analysis für Ingenieure*. München: Fachbuchverlag Leipzig im Carl Hanser Verlag (2004).
6. http://www.et.hs-wismar.de/~schott/IngMat
7. Schott, D., Fluch und Segen der Computermathematik, *Global J. of Engng. Educ.*, 8, **3**, 319-326 (2004).
8. Schott, D., Image reconstruction – an interesting project for basic science education. *Proc. 6th Baltic Region Seminar on Engng. Educ.*, Wismar/Warnemünde, Germany, 13-17 (2002).
9. Schott, D., Challenges in computer mathematics in engineering education. *Proc.* 8*th Baltic Region Seminar on Engng. Educ.*, Kaunas, Lithuania (2004).

BIOGRAPHY

Dieter Schott is a Professor of Numerical Mathematics and Technical Mechanics in the Department of Electrical Engineering and Computer Science at Hochschule Wismar, University of Technology, Business and Design, Wismar, Germany. He graduated from the University of Rostock, Germany, as a mathematician in 1972. He received there a Doctorate in 1976 and the Habilitated Doctor's degree in 1982 in the field of mathematics.

Later, Prof. Schott worked at the Universities of Güstrow and Rostock, where he was engaged in the education of scientists, teachers and engineers. His numerous publications are mainly related to the field of numerical analysis.

Prof. Schott is very interested in new teaching methods within the field of mathematics, including project work, computer mathematics and e-learning.

He has supervised foreign students in the design of mathematical multimedia teaching units and project studies. He is the author of a textbook concerning engineering mathematics using MATLAB.

Prof. Schott is also a Co-Director of the *Gottlob Frege Centre for Engineering Science and Design*, a satellite centre of the UICEE.

Prof. Schott has also published widely as an author in various UICEE journals and proceedings. Further, he has acted as a referee for several UICEE publications.

In February 2002, he received the UICEE Silver Badge of Honour for *distinguished contributions to engineering education and outstanding achievements in the globalisation of engineering education.*