Research Announcement

A NON-CONFORMING FINITE ELEMENT METHOD FOR A SINGULARLY PERTURBED BOUNDARY VALUE PROBLEM

D. Adam, A. Felgenhauer, H.-G. Roos & M. Stynes

We analyse a new non-conforming Petrov-Galerkin finite element method for solving linear singularly perturbed two-point boundary value problems without turning points. The method is shown to be convergent, uniformly in the perturbation parameter, of order $h^{1/2}$ in a norm slightly stronger than the energy norm. Our proof uses a new abstract convergence theorem for Petrov-Galerkin finite element methods. Full details appear in [1].

Reference

[1] D. Adam, A. Felgenhauer, H.-G. Roos and M. Stynes, *A nonconforming finite element method for a singularly perturbed boundary value problem* (1992) (Preprint 1992–10, Mathematics Department, University College Cork).

Martin Stynes
Department of Mathematics
University College
Cork

Dirk Adam, Andreas Felgenhauer & Hans-Görg Roos
Institute of Numerical Analysis
Technische Universität Dresden
Germany

BENEFITS AND ADVANTAGES OF AN INTEGRATED MATHEMATICS AND COMPUTER SCIENCE DEGREE

T. C. Hurley

Introduction.

Computer science grew out of mathematics - it is surely enough to mention the names and contributions of Babbage, Boole, Hilbert, Von Neumann, Turing. On the other hand, it is clear that mathematics grew out of computation. The two areas are intimately related. Strenuous efforts have been made ever since computers were invented to separate mathematics and computer science. Modern techniques in both areas serve only as a reminder of how much each can be dependent on the other.

Difficulties.

In schools, mathematics is thought of as a 'closed' subject, whereas computer science is thought of as a 'technical' subject. Thus many creative individuals are turned off mathematics and computer science at an early stage.

A pure computer science degree is now considered in some quarters to be a *purely technical* training and not a proper education for a scientist or an engineer. One major employer is quoted as saying: "I only hire mathematicians and engineers–computer science graduates do not know how to *solve problems*".

Engineering and physical science degree programmes insist on a reasonable mathematical background. This is not true for computer science programmes and the graduate here needs this type of background even more than an engineer–consider, as an

example, the situation where many computer science graduates are thrown into work which involves *Field Theory*.

'Programming' has been replaced by 'software engineering', with some emphasis on the 'engineering' aspect. Software engineers need a broad mathematical education. Computer science graduates have no mathematical equipment in which to analyse what they are trying to do.

Hoare, a well-known computer scientist, in his inaugural address made the following assertions:

- computer programs are mathematical expressions;
- a programming language is a mathematical theory;
- programming is a mathematical activity.

Computer science is dominated by the 'life-cycle' which obscures the mathematical dependency. It should be generating new formalism, modelling with formalism, constructing proofs and algorithms. It is stated that software engineers do not construct theories but apply *methods*, thus arguing that it is a *closed* problem and that computer systems now carry out mechanical methods. However the methods come out of a theoretical understanding. Everything in software engineering is seen in terms of the final product and the theoretical context is overlooked.

There are great difficulties in attracting students to mathematics degrees, especially from those who come into Arts or Science degree programmes with no specific subject in mind ('undenominated' programmes). We have to compete with other subjects which are *perceived* to be more vocational, are also perceived as *easier options* and do not have the high fallout rate that honours mathematics has.

Students of mathematics lack *motivation* and have *inadequate preparation*; this contributes to their subsequent subject choices.

On the other hand it is recognized that mathematicians, when they do solve problems, are unable to *communicate* their solution to others.

An integrated mathematics and computer science degree will go some way to rectifying many of these problems.

Common interests?

What areas ought mathematicians be interested in? In the software development process, *validation* and *verification* are now extremely important. "Get the *right system* and get the *system right*". The ideas of *modelling* and *proof* are emphasized again and again.

To quote a person working in the industry: "In the software engineering process, the use of mathematical ideas requires more resources at the initial stages but the **total** resources are less and an *enhanced* product ensues".

To be more specific, the following mathematical ideas should play an important part in the education of a software engineer:

- formal methods;
- modern logics;
- parallel processing;
- modelling, forecasting;
- complexity;
- computability;
- encryption and coding;
- statistics and probability (*networking* is a probabilistic entity);
- neural networks.

Benefits and advantages.

It is nice to make a list, so here goes! (Thanks to many articles and talks from which these have been taken.)
Benefits of an integrated degree (not necessarily in order):

- volatile and exciting area;
- focused programme;
- it still embodies *essence* of mathematics when carefully thought out;
- it facilitates development of PSQs (personal skills and qualities) through group and individual projects;
- vocational;
- accessibility to potentially good students;
- it prepares potential school teachers in both computer science and in mathematics;

- it is a laboratory based programme and can lead to better funding;
- modern applicable area;
- employment prospects for students are greatly increased;
- accessibility to funding agencies and politicians;
- it leads to active applications and involvement by the student who is encouraged to build his/her own system with great satisfaction;
- the emphasis is on *abstraction* and *proof.*
- it is not the 'death of proof' but a deeper understanding of what 'proof' is will ensue from experiment;
- students can see harder continuous mathematics in a new light;
- there is less emphasis on graduates as technicians and more emphasis on science;
- the image of mathematics and of computer science is enormously improved.

Employers' perspective.

Where is mathematics needed in industry? Do employers recognize that mathematics is needed? Again we might ask: "Do employers and colleagues within industry support mathematical activity?". These are difficult questions and the answers are of course dependent on the industry in question. The level of mathematical activity in different industries varies enormously. If we look at the worldwide context then we see that the industries using computer science may be roughly bracketed as follows:

1. Engineering houses, computer manufacturers, research laboratories, scientific users (such as the Meteorological Office).
2. Software houses.
3. Commercial users. These could be subclassified as:
   (a) utilities;
   (b) manufacturers;
   (c) finance institutions;
   (d) retail outlets.

These are listed in order of acceptability of the mathematical tradition. The ones within group 3 in fact have traditionally employed many non-graduates in their computing areas.

Those concerned with rigorous development also include the military and data-security teams within commerce and finance.

These classifications are on an international basis and Ireland has its own particular problems as a small open economy, with manufacturing but little research and development. Our employers also have the narrow view that graduates should be trained for a *specific job* but overlook how things are changing rapidly and that a well-trained mind with the ability to think and solve problems as and when they arise is what is needed.

Where do the graduates go?

There are now figures available from different countries as to where mathematics graduates get employment. The figures for Ireland cannot compare as we have so few honours mathematics graduates in comparison to other countries - e.g. the UK has the order of 4000 mathematics graduates each year, which, pro rata, would amount to about 350 graduates for Ireland. We are nowhere near this number.

The British *Sunday Times* on 8/8/91 reports: "Britain faces an acute shortage of mathematicians". If this is so, where does this leave us? Many of our good mathematics students go into engineering and the school culture is such that those good at mathematics are encouraged to apply there. Would those good at English be encouraged to do a degree in, say, accountancy?

Mathematics is not looked on as a career in itself and nowadays additionally some of the best mathematicians are attracted into business and commerce. If they are good at mathematics and must pursue a career in business, why is it not pointed out that a degree in mathematics and economics is a much better preparation? [1]

[1] See the article by Joel Franklin on *Mathematical methods of economics* in the American Math. Monthly, **90** (1983), 229-244. Among other things in this article it is pointed out that seven of

In the UK, 26% of mathematics graduates[2] go into the computer industry and 52% into finance. Of the mathematics graduates in Ireland that did *not* go on to further study, 82% (1991 figures) went into financial work and computing.[3] Graduates with an *integrated* degree would be much better prepared for careers in these areas.

Information Technology.

One of the buzz-words at the moment is information technology (IT). I, and many others, doubtless, are confused at what exactly IT is. It can mean different things to different persons, depending on whether you are a scientist, engineer, business person, sociologist, industrialist, philosopher or psychologist. The EU has classified IT under five headings. I am grateful to Pat Fitzpatrick for this information.

1. Software engineering or knowledge-based information systems.
2. AI (artificial intelligence).
3. VLSI (very large scale integration).
4. Communications.
5. Human interface.

Of these, only one, the human interface does not require a substantial mathematical background–perhaps. Where does mathematics come into these areas? Pat Fitzpatrick again has some of the answers. *Algorithm design* is fundamental for VLSI, *logic design, LISP* for AI and *coding theory, cryptography, digital signalling* are all areas of importance in communications. The importance of mathematical ideas to software engineering has already been dealt with. Even within the human interface, the ideas behind *security* and *authentication* are now increasingly important.

the previous twelve Nobel prizes in Economics 'involved work that is heavily mathematical'.

[2] Interestingly, the career booklet for mathematical graduates contains computer science as a subsection.

[3] Statistics on graduate careers only give destination for the year after first graduation.

Computer Algebra[4] systems.

It is my opinion that it is inconceivable that a graduate in mathematics, and I would argue now a graduate in computer science, would nowadays not have had experience of computer algebra systems. It is like saying that a chemistry graduate never did any laboratory work! A graduate in many years time will continue to use mathematics if he/she realizes that some of the hard calculations and theoretical background, which he/she has probably forgotten, can be done by machine.

Computer algebra systems are likely to become as useful to scientists and engineers as word-processors and data-bases have become to all.

Cohen remarks: "A mediocre mathematician with a computer might be able to simulate the creative powers of a top notch mathematician with pen and paper". How much more could a top-notch mathematician produce?

Packages like MAPLE, MATHEMATICA, REDUCE, CAYLEY (to be replaced by MAGMA soon), GAP, AXIOM should be, in fact **must be**, integrated within our courses. From our experience it is much easier to integrate these within an integrated mathematics and computing degree, where an element of laboratory and experimental work already exists.

Generally speaking a package like MAPLE, REDUCE or MATHEMATICA would go well with analysis-type courses or applied/mathematical physics courses, and one of CAYLEY (soon to be MAGMA), GAP or maybe AXIOM[5] should be integrated within abstract algebra courses such as group theory, field theory, coding theory or even homological algebra. Also MATRIX, for

[4] Computer Algebra is also often referred to as *symbolic manipulation.*

[5] AXIOM is a package which has great potential, especially when you need to build your own abstract system, but is not available for many machines yet, is expensive and difficult to use.

linear algebra and linear programming, is an excellent CAL (computer assisted learning) package which goes down well with students. In addition, MACTUTOR, for MACs, is an excellent all-purpose CAL package.

Use of computer algebra does however tend to increase the workload on the instructor and many mathematics departments and colleges are as yet unwilling to recognize this fact and indeed do not recognize the importance of considering mathematics as a laboratory-based subject.

Leibnitz stated: ".... it is unworthy of excellent men to lose many hours like slaves in the labour of calculation, which could safely be relegated to someone else if the machine were used".

Syllabus.

There are various suggestions as to what should be included in a syllabus for a joint degree. Not all areas can be covered and choices must be made. What is important is that a good *scientific* training in *both* mathematics and computer science should be an essential part of any programme. Some might argue that a language and/or business skills and/or placement should be part of the programme but incorporating two major subjects does not leave much time for anything else, even within a four-year course. Projects in all years will develop PSQs.

There can be **core** courses and options to suit individual tastes. It is important to note however that this an *integrated* programme and that although some of the mathematics and computer science courses are independent, the programme should be drawn up by reference to common areas of interest and dependency. It is not a matter of simply combining the subjects as in a traditional two-subject degrees.

To spell out a full syllabus would be pointless here but some suggestions on related areas that could be included are given below–fundamental courses are in *addition* to these.

Within mathematics, the importance of *discrete mathematics* to computer science is fundamental and has now been recognized as such but this is only part of the picture. Discrete mathematics should include algorithms, recursive function theory, Boolean

algebra, logic and circuit design. Analysis courses are also an important element and must be compulsory in the early years. A good idea which works well at UCG is to have courses on metric spaces *and* fractal geometry. Other ideas to think about: have a course in number theory *and* cryptography; include coding theory with field theory, and semigroups and machines with a group theory course. Category theory: "Categories themselves are the models of an essentially algebraic theory and nearly all derived concepts are finitary and algorithmic in nature" (John Gray in *Computational Category Theory*). This is all good mathematics.

Numerical analysis cuts across both areas and could be included either within the mathematics core or within the computer science core areas.

In computer science, courses on programming, operating systems, networking and communications, data bases, architecture, modelling, algorithms, computability, complexity, graphics, parallel processing, artificial intelligence and logical aspects of computing, would appear to be fundamental. Consideration should also be given to courses on automated reasoning and neural networks which would fit in well with the mathematics. In general, the computer science element in a joint programme should be directed more to *software* considerations.

Everyone has their own favourite language but at the risk of upsetting some persons I would suggest that C and C++ are the most useful (*and* mathematically oriented) languages now used in science and industry. This also fits in well with operating systems as many of these were originally written in C.

Conclusion.

The programme suggested is what is needed and will train *both* mathematicians and computer scientists for worthwhile careers. It will also satisfy the needs of industry and the commercial world.

Both subjects have much to learn from one another. The debate will continue!

T. C. Hurley
Department of Mathematics
University College
Galway

THE IRISH INTERVARSITY COMPETITION IN MATHEMATICS

Timothy Murphy

What, in your opinion, is the next term in the sequence

$$3, 5, 1, 15, 11, 10?$$

This will be a cinch. Easy one to start with. Thanks, Des. Knew he was a good sort.

Let's see. Probably squares minus 1 ... that will explain the 15, anyway. No, doesn't seem to be that. Maybe it's not quite as simple as I thought ...

The Irish Intervarsity in Mathematics came out of the fertile brain of Des MacHale. It was a natural extension of the Superbrain Competition that Des has been running in UCC since 1984. (The question above was the opening problem in the 1987 Superbrain paper.)

I've got it! Should have thought of that earlier. It's obvious. Just a common-or-garden code, 1 for A, 2 for B, and so on. Let's see... CEAOKJ.... What language is this? Maybe the Viking name for Cork? Well, it was worth a try, anyway.

The Intervarsity was first held in Cork, in 1990. It moved to TCD for the next 2 years; and UCG hosted the event this year.

The competition was won by TCD in 1990 and 1991, and by UCC in 1992 and 1993. (UCC and UCD tied in 1993; but the prize went to UCC for the best individual result.)

Although the competition is mainly a team event, there is also an individual winner each year. Paul Harrington of TCD won in 1990; Aiden O'Reilly of Maynooth in 1991; Cian Dorr of UCC in 1992; and Peter Hegarty of UCC in 1993.