Enhancing the computing side of the Mathematics student journey

James H Davenport
Dept. Computer Science
University of Bath
Bath BA2 7AY
J.H.Davenport@bath.ac.uk
http://staff.bath.ac.uk/masjhd

Alastair Spence
Dept. Mathematical Sciences
University of Bath
Bath BA2 7AY
A.Spence@bath.ac.uk
http://staff.bath.ac.uk/masas

David Wilson
Dept. Computer Science
University of Bath
Bath BA2 7AY
D.J.Wilson@bath.ac.uk
http://www.cs.bath.ac.uk/~djw42/

Abstract
Most Mathematics students learn some computing, and many students need to use it in their studies, often in Numerical Analysis or Statistics. This knowledge and skill may be self-taught, taught by the Mathematics Department or taught by the Computer Science Department. It is the contention of this paper that none of these is right, and an integrated interdisciplinary course is to be preferred, yields better intellectual results, and much enhances the Mathematics student's appreciation, not just of Computing, but also of Mathematics. Having established a relevant course, the issue is to make it *seem* relevant to the students. This, we believe, is best done through a mixture of relevant examples and illustrative mini-lectures.

Keywords
Mathematics, Computing, relevance, interdisciplinarity

1. Introduction
The interplay between programming and learning Mathematics goes back at least to the Logo project (Feurzeig *et al.*, 1969). We, and in our opinion most of the Mathematics community, believe that students *should* be taught some programing, both for its utility elsewhere in the curriculum (typically Numerical Analysis and Statistics, but other courses can be enhanced with computer-based example) and its utility in later life: very few mathematicians will use their Mathematics in industry without a computer. Traditionally, one of three approaches has been used.

1. Muddle through — "they must have learnt some of this at school", and the Numerical Analysis (or Statistics) lecturer can fill in the gaps. Unfortunately, the current (as opposed to planned, and the plan will take time to feed through: English universities will only start seeing the effects of bulk adoption of GCSE Computing in 2018, and Key Stage 3 Computing in 2020) English school curriculum does not teach programming, and Computing A-level has less than 5% the uptake of Mathematics, and only 27% of the uptake of Further Mathematics.
2. Teach it in Mathematics, generally in Numerical Analysis. This generally meets the aims of the specific course, but not the wider aims. If the Statistics courses also use some programming, the students are apt to be confused, and perceive incoherence.

3. Have it taught as "service teaching" by Computer Science. This generally meets the wider aims, but is disconnected from the rest of the curriculum and, worse, is perceived as such by the students.

We have elsewhere (Davenport *et al.*, 2014) explained how, since the 2009/10 year, Bath has broken free of this trichotomy, with a tailor-made interdisciplinary course "XX10190 Programming and Discrete Mathematics" — the XX prefix indicating visibly that it is interdisciplinary. This course is based on the principle that "The course is greater than the sum of its parts", or more formally as the course aim:

> The course should **be**, and be seen to be, relevant to the rest of the Mathematics curriculum, and not just as "a useful skill for later on".

Hence a great deal of attention went into connecting this course to the rest of the curriculum: see section 4.

While this achieves the aims of giving students a good general grounding in programming (including testing and documentation) which is perceived, at least in hindsight, by the students to be fairly relevant to the rest of the course, there is more that we do to enhance the student experience. This paper describes the relevance and these further steps.

2. Context at Bath

The Department of Mathematical Sciences offers a range of degrees, both BSc (three years study) and MMath (four years study). About 1/3 of the students also do an Industrial Placement between years 2 and 3 of study. The intake has grown at a 6% compound rate for many years, and is now (2013/4) 300. A major degree scheme review led to a new syllabus starting in 2009, of which XX10190 was among the major innovations.

Prior to the introduction of XX10190, Bath had followed the third route above, with the Department of Computer Science teaching a relatively standard programming course, first in C then from 2001 in Java, in the second semester of Year 1. Despite the best efforts of the lecturer concerned, there were frequent complaints of "why are we doing this", "what has this to do with the rest of the course" etc.

3. Syllabus of XX10190

The main syllabus follows from the course aim given above. From the point of view of the mathematicians, the choice of programming language was easy: the languages used later on were MatLab and R, with the MatLab programming being distinctly more challenging. Hence the relevant language was clearly MatLab. From a Computer Science point of view, no-one had ever heard of teaching MatLab as a primary programming language. While there are many MatLab texts, almost all of them focus on the graphics capability and the wide range of toolboxes available, rather than the programming aspects.

The key concepts of recursion and induction are taught side-by-side: the first example, literally in week 1, is the Fibonacci numbers, which are defined by induction, programmed by recursion, and in the next four weeks have theorems on growth proved by induction, and have these related to the *O*-complexity of the programs produced earlier. The tutors remark that part of their work consists in explaining recursion to the student who understands induction, or *vice versa.* Induction is used to prove non-trivial theorems about the complexity of sorting and searching algorithms, such as the following two (proved in the corresponding weeks 7 and 8, and more technically difficult than most induction exercises).

Theorem [7.6] For any $n > 4$, mergsesort, using the base cases from Lemma 7.1--7.5, takes at most $n\ \mathrm{ceil}(\log_2 n) - \frac{9}{8}n + 1$ data comparisons on n objects.

Theorem [8.1] The number of comparisons needed by [the median-finding, or more generally *t*-th finding] algorithm on n objects, for any *t,* is at most $15n - 163$, for $n > 32$. The second result is usually quoted as "finding the median is $O(n)$", but to prove it one has to go through the details of 8.1.

3.1 Course Structure

The Programming and Mathematics components are each taught in one lecture a week, with a third plenary hour being used by both lecturers for a "Problems Class", discussing work the students have done, or are doing. Each student also gets one hour a week in a programming laboratory, where a group of 12-14 students has a dedicated tutor, and an hour of Mathematics tutorial, typically in a group of 16. The groups are centrally assigned, with an effort made to ensure ethnic mixing and gender balance.

The programming laboratories take place in a 75 seater laboratory holding five groups, with a mixture of experienced and less experienced tutors, and the Programming lecturer also attends for the first few weeks. The students settle down remarkably quickly, and the mix of tutors means that Programming problems very rarely have to be escalated to the lecturer[1]. The collegiate atmosphere in the laboratory sessions also means that a substantial amount of peer learning takes place, or if two students are stuck, they have more courage to call for the tutor. Since not all students start from an equal background in Computing, to put it mildly, the tutors run four or five "catch-up" classes half-way through semester 1, where students, either self-identified or identified by their tutors, can get extra help with the programming aspects. These are generally successful in recovering those who are struggling before the assessed coursework comes out, and are much appreciated.

The tutors are generally research students (from both Mathematical Sciences and Computer Science Departments) or final-year MMath students[2].

4. Relevance to other courses

All students doing Mathematical Sciences degrees do XX10190, and hence the XX10190 team knows their entire programme: Analysis, Algebra, Mathematical Methods and Probability/Statistics. However, there are students on other degree schemes (Computer Science and Mathematics, Mathematics and Physics) whose syllabus includes (the first three of) these other courses, so those cannot rely on XX10190.

4.1 Statistics

The Statistics course is in the second semester of year 1 (after Probability in the first semester). For many years, it has been taught as a fairly practical course, using the `R' statistical system. Prior to XX10190, this always caused difficulties for the lecturer and tutors. Since it was at the same time as the pre-XX10190 Programming course, the teachers could not rely on anything taught there

These days, the Statistics lecturer just points to a one-page "R for MatLab users" that we have written, and the difficulty seems to have completely gone away.

---

[1] The problems escalated to the lecturer tend to be administrative: "student not properly registered" or "allocation to groups doesn't work for this student"

[2] Who have all done XX10190 themselves, now that the course has been running for three years.

4.2 Algebra

Since MatLab is the "Matrix Laboratory", matrices come up naturally throughout the course. Adjacency matrices of graphs are introduced early, and sparse matrices are used as an example of data structures. The students are also told that the core of Google is indeed a very large adjacency matrix, and this is reinforced in second year linear algebra.
The linear algebra component of the algebra stream is taught as being over an arbitrary field. Nevertheless, most examples are over **Q** or **R**, and the student could be forgiven for thinking that these are all the examples that matter, or indeed that the "arbitrary field" idea is "spurious generality". The Discrete Mathematics component of XX10190 includes a chapter on Coding Theory, which is very much linear algebra over $\boldsymbol{F}_2$.

4.3 Analysis

Week 6 of XX10190, which is while the Analysis course is doing sequences and series, introduces the $O/\Omega/\Theta$ notation (so Ω is defined as $f(n) = \Omega\big(g(n)\big) \Leftrightarrow \exists b, B > 0, \forall n > b |f(n)| \geq B|g(n)|$), and uses it heavily in the analysis of algorithms. Hence orders of growth are already familiar objects by the time they are met in formal Analysis.

4.4 Cryptography

By the second semester, students have met (and used) the Symbolic Toolbox of MatLab (essentially the muPAD computer algebra system) and in particular its ability to manipulate arbitrary-sized integers. Hence, after the Diffie-Hellman and RSA cryptosystems have been introduced (and in fact this is familiar to many students) with small numbers, the students can be given another set of examples, with 60-digit or more integers, to appreciate the real power of these systems. In fact, we use the same examples sheet as before, with only the numbers changed, to make it clear that the algorithms are the same. Of course, guesswork will no longer find the solutions!

5. Other Relevance

Although the teaching team felt that the "why are we doing this" questions had become much less strident following the introduction of XX10190, they were still there.

5.1 Vignettes

In 2013 we introduced the concept of vignettes of research, where people other than the lecturers talk about their research lying on the boundary of Mathematics and Computing. Sometimes these are tutors describing their own research, and this goes down well as these tutors are normally more senior ones, known to the whole laboratory group, and are often those who also take "catch-up" classes. One particularly topical one was a research student speaking about her research on ash cloud modelling.
Others vignettes are given by academic staff, and we reproduce (figure 1) an illustration from one, showing how weather forecasting has improved over time, due to a combination of higher Computing power and better Mathematics, to the point where today's 3-day forecast is as accurate as the 1-day forecast in 1985.

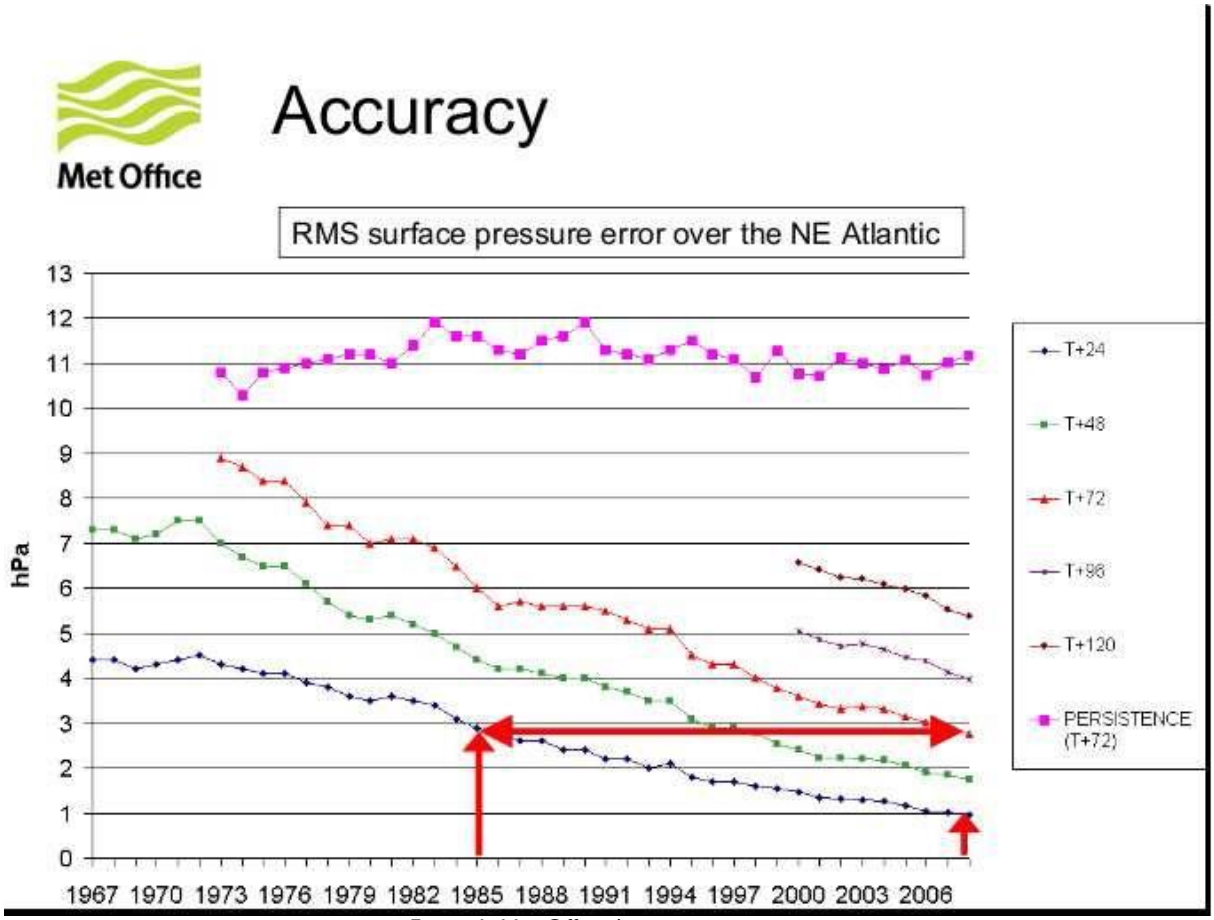

Figure 1; Met Office Accuracy over time
Courtesy of the Met Office and Dr, Melina Freitag.

5.2 Relevant Examples

By week 6, the students have seen both insertion sort and merge sort, and have had it proved that the average complexity is $O(n^2)$ and $O(n \log n)$ respectively. Lest they think this is the end of the story, exercise 6 asks them to write a program to sort n objects that recurses and merges if $n \geq N$, and uses insertion sort if $n < N$. They also have to find a suitable value of N, and are often surprised when N turn out to be 400. They are then shown Figure 2, work done by a Bath student in an industrial setting (British Aerospace) showing similar poly-logarithmic behaviour, and which inspired the first author to teach poly-algorithms.

6. Conclusion

It is not easy to make programming relevant to Mathematics students, but it *is* possible, and the increase in student competence, satisfaction and engagement is well worth it. The competence is visible: 2nd and 3rd year Numerical Analysis lecturers commented on this unprompted, and the Statistics lecturers have observed that "The 'Teaching R' problem has gone away". Satisfaction and engagement are harder for us to measure, as several other changes took place at the same time as the introduction of XX10190, and the student feedback format has changed from paper to electronic, with a significant drop in response rate. No student feedback has suggested that this was a change for the worse.

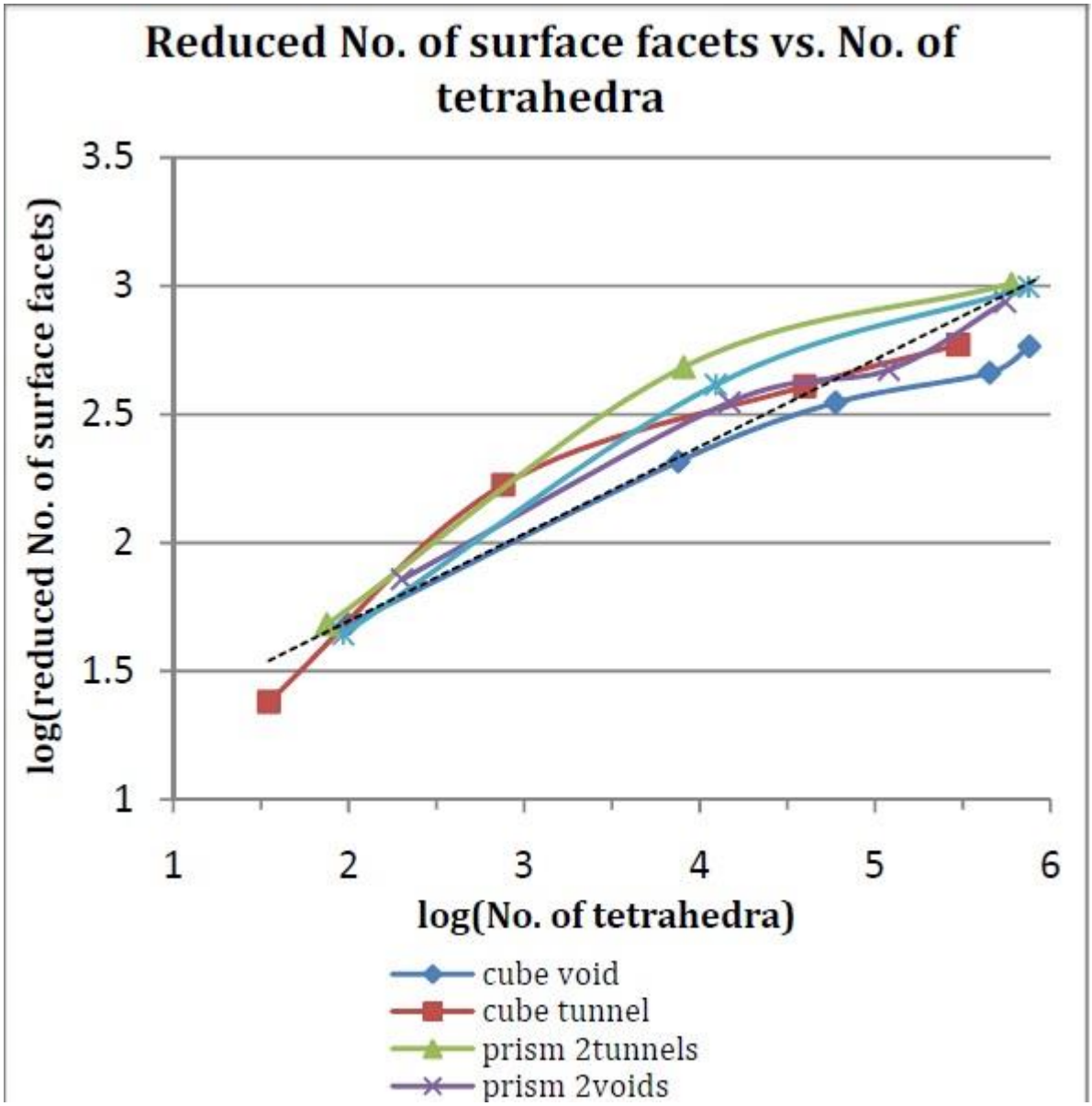

Figure 2: extract from Holstein (2010)

References

Chapman, S.J. (2009) *Essentials of MATLAB Programming*. London etc.: Cengage.

Davenport, J.H., Wilson, D.J., Graham, I.G., Sankaran, G.K., Spence, A., Blake, J.C.H. & Kynaston, S.J., (2014) Interdisciplinary Teaching of Computing to Mathematics Students: XX10190 Programming and Discrete Mathematics. Submitted to *MSOR Connections.* http://opus.bath.ac.uk/37841/

Feurzeig, W., Papert, S., Bloom, M., Grant, R. & Solomon, C., Final Report on the first fifteen months of the LOGO Project. Technical Report 1889 Bolt Beranek & Newman Inc., 30/11/ 1969.

Holstein, A.S. (2010) *Computational Topology: Identifying Topological Features of a 3D Mesh.* MSc. Modern Applications of Mathematics Dissertation, University of Bath, 2010.