
Alice Interactive Mathematics

Neil Strickland
University of Sheffield

n.p.strickland@sheffield.ac.uk

Supplier's contact details

University of Sheffield
[http://dani.shef.ac.uk/
aimsourc](http://dani.shef.ac.uk/aimsourc)

AIM is a system for computer-aided assessment in mathematics and related disciplines, with emphasis on formative assessment. The original version was developed at the University of Gent in Belgium, but details in this article refer to a heavily revised version developed by the author at the University of Sheffield. Documentation, downloads, and archives of the AIM mailing list are available from <http://dani.shef.ac.uk/aimsourc>, and you can try the system as a guest by visiting <http://aim.shef.ac.uk> and giving "guest" as your student ID and password.

There are numerous other systems for CAA in mathematics, but AIM has a number of very distinctive features.

AIM is based on Maple

AIM is mostly written in the Maple programming language, so it can call on the full range of mathematical knowledge built in to Maple.

- If a student gives a correct answer in a form different from that supplied by the teacher, AIM can still determine that it is correct..
- If a student solves a system of equations incorrectly, then AIM can substitute the incorrect answers back in to the equations, and show the student that they do not work out.
- If a student integrates an expression incorrectly, then AIM can differentiate the incorrect answer and show the student that it is not the same as the original function.
- More detailed feedback is available for certain common errors. For example, one can set a standard integration question asking students to integrate $\sin^2(x)$. Without any explicit action by the question setter, AIM will examine the form of the integrand and recognize that students will be tempted to answer $\sin^3(x)/3$, or possibly $\sin^3(x)/(3 \cos(x))$. If a student makes either of these errors, then AIM will automatically generate an explanation of what has gone wrong.
- More generally, questions can be set up to give immediate and detailed feedback depending on mathematical features of the answer offered. For some types of question, there are extensive facilities for this built in to the system; for other types, one has to write elaborate question definitions to achieve this.
- One can set questions asking the student to give an example of a function (or matrix, or vector ...) with some specified properties. This kind of exercise seems pedagogically very valuable, but very labour-intensive to mark by hand. Of course a CAA system can only include questions of this type if it has enough mathematical intelligence to determine whether a suggested example has the required properties. Ideally, if a student's answer is not correct, the system should give a reason why not.

AIM is Open Source

The source code for AIM is freely available from <http://dani.shef.ac.uk/aimsourc>. (Eventually the legal paperwork will be sorted out so it can be formally released under the GNU General Public License.) Recently, a lot of work has gone in to making the code cleanly modular and documenting its structure. There are mathematicians in Belgium, Canada, America and Britain using AIM, many of whom have strong programming skills. The hope is that we will establish an active community of AIM developers, contributing new

facilities, packages to deal intelligently with new types of questions, and so on. The project will be owned and driven by mathematicians (and academics in related areas such as physics and engineering) so it will be highly responsive to the suggestions and concerns of the community.

AIM accepts free text input

Most AIM questions require that students type their answers as free text, using Maple syntax. For example, an answer of $\sin^3(x)/(3 \cos(x))$ would be entered as `sin(x)^3/(3*cos(x))`. It seems clear that questions with this flexibility are pedagogically more valuable than multiple choice or multiple response questions (although these are also available in AIM). On the other hand, of course, the students need to learn the syntax. AIM tries quite hard to help with this:

- Answers that cannot be parsed are discounted without any penalty. Students can ask AIM to parse their answers without marking them, to check that they are interpreted in the intended way.
- If a student enters an answer with mismatched brackets, then AIM will draw a little diagram indicating which brackets match against which other brackets, and which bracket is causing a problem.
- If a student forgets to include stars for multiplication (eg `2x` in place of `2*x`) then AIM will generally work out where the stars should have been, and report back the student's answer with the suggested stars marked in red.
- Similar feedback is given for a variety of other common errors, such as `x^-1` in place of `x^(-1)`, `sinx` in place of `sin(x)`, `sin^(-1)(x)` in place of `arcsin(x)`, and so on.

The syntax checks are heuristic, and occasionally the feedback misses the point or is misleading, but students generally found it helpful.

Despite the help, difficulties with syntax remained a significant cause of student dissatisfaction. Some lecturers will doubtless regard this as a regrettable feature of the basic design, and would prefer some kind of graphical interface for building up mathematical expressions. On the one hand, it should be perfectly possible to integrate a graphical expression editor into AIM, particularly if written as a Java applet. Given the open source model, if there is sufficient demand, then this will probably happen. On the other hand, my personal opinion is rather different. I feel that all mathematics graduates should certainly have the ability to enter complex

formulae into some kind of computer system, and preferably be skilled in using programs such as Maple as a tool. AIM is a good vehicle for introducing these skills, as it gives more detailed and helpful feedback on syntax problems than Maple itself does. Of course, it remains important to improve the automatic syntax feedback, and to study other possible ways of helping the students.

AIM is accessed via the web

AIM runs on a central server, which the students access using a web browser in the usual way. Mathematical formulae are converted to complex HTML tables, which can be viewed by any browser, without use of Java applets or special plugins. The quality of these displays is generally fairly good, although not of the same standard as LaTeX. (The system is designed in such a way that migration to MathML should be painless, if and when that seems appropriate.)

There is a separate web interface for lecturers, which can be used to set and test questions, administer a register of students, review marks and so on.

Writing questions

Currently, all AIM questions are written using a kind of mark-up language incorporating elements of Maple code and elements of LaTeX. (In the future, we envisage having graphical interfaces to set up certain standard types of questions, but moderately advanced users will probably still prefer the mark-up language.) The minimal example is as follows:

```
t> What is 2+2?
a> 4
end>
```

The first line tells AIM to display the question, and the second line specifies the answer.

Here is a more realistic example, involving some LaTeX, and some random parameters. (For simplicity, we have not used AIM's special facilities for integration questions.)

```
local> p, x
h> p := x^4 + randpoly(x, degree=3,
coeffs=rand(-3..3));
t> Evaluate the following integral:
  \[ \int @p@ dx \]
s> [ proc(ans) `aim/Test`(diff(ans, x),
p) end, int(p, x) ]
sb>
t> Just use the fact that
  \[ \int x^n dx = x^{n+1}/(n+1) \]
se>
end>
```

The first line declares that \mathbf{p} and \mathbf{x} are local variables. The second line is written in Maple; it sets \mathbf{p} to be a random monic polynomial of degree 4. The following two lines state the question. They are essentially written in LaTeX, with the additional feature that Maple expressions enclosed in @ signs are evaluated and converted to LaTeX (and then to HTML) before the text is displayed. The next line, starting with \mathbf{s} , is a two element list. The second element $\mathbf{int}(\mathbf{p}, \mathbf{x})$ specifies the correct answer. The first element is a Maple procedure used to check the student's answer, by differentiating it and testing for equality with \mathbf{p} . (Note that this allows, but does not require, a "+c" term.) In more complex examples, the checking procedure could look for common errors, give tailored feedback, assign partial credit, and so on.

In Sheffield we have a bank of around 150 questions for a first year core mathematics course, covering functions, differentiation, integration and matrices; they range from a few lines up to about 120 lines long. One of the longest questions generates systems of linear equations. The number of variables and the number of equations are chosen randomly, and the system also chooses randomly whether the system should have zero, one or infinitely many solutions. The question is generated by working backwards from the solution (if there is one) to ensure that the numbers involved are not too unpleasant.

Other AIM users also have large banks of questions. The majority are at about the same mathematical level as ours, but some cover more advanced material such as vector calculus. AIM could in fact be used for certain kinds of exercises throughout the undergraduate curriculum, including abstract algebra, for example; as yet there are no substantial examples of this, but I hope to write some AIM questions for a Level 3 course on Rings, Modules and Linear Algebra next semester.

The marking system

Most AIM questions give no partial credit. However, students are allowed to repeat questions that they have got wrong. If they get the correct answer after several incorrect attempts, then they are penalised by 10% of the value of the question for each such attempt. This scheme has a number of advantages. Unlike with conventional homework, students are told immediately whether their answer is right or wrong, and often given some information about what is wrong with incorrect answers. The marking scheme gives them a clear incentive to absorb this feedback and think again. It also makes it possible for weak but diligent and persistent students to get good marks.

Experience in Sheffield

A previous version of AIM was used in Sheffield in Autumn 2000. Around 230 students took two tests; they were told that they were required to pass them, but otherwise they were not counted when assigning grades for the course. The exercise seemed reasonably successful, but no formal evaluation was done.

The current version was used in Autumn 2001, in a first year core mathematics course. Around 230 students were involved, many of them being dual degree students. They did a diagnostic test at the beginning of term and four other compulsory tests later on. Four optional practise tests were made available for revision. There will be a final exam in late January counting for 90% of the marks for the course, and the four compulsory tests will count for the remaining 10%. There was a fifteen minute presentation about the system in the first lecture, and help sessions were offered two days a week for the first three weeks of term. Very few students attended these sessions.

After the third compulsory test, the students were asked to fill in a questionnaire. Detailed results are still being analysed, but certain messages come through very clearly.

Things that the students liked:

- The fact that you could try again if answers were incorrect (this was extremely popular).
- Instant feedback
- The ability to do questions from home, in their own time, without any pressure.

Things that they did not like:

- *The lack of marks for method.* This is deeply embedded in the design of the system, and could not reasonably be changed. Some of the issues would be addressed by the system of hints and subquestions that is implemented in the latest versions of AIM developed in Belgium. Unfortunately these developments are currently incompatible with the developments made in Sheffield, but I hope that everything will be merged at some point in 2002.
- *The lack of partial credit for answers that are incorrect only in one small detail (such as a sign)* This was very unpopular and caused a lot of frustration. It would probably be feasible (but not easy) to write code to detect when an answer is slightly incorrect, and to indicate which part of the expression contains the error. Given the way that the marking system works, this would probably

address the real issue . I suspect that if one tried to write code to assign partial credit based on this kind of analysis, it would regularly behave in a way that seemed palpably unfair.

- *The time taken to enter answers.* This is particularly a problem for students who do not have their own PC's. The open access PC rooms in the university are often crowded at popular times, and network congestion means that logging in and starting up a browser can be a very slow process. On the other hand, it is part of the university's declared IT strategy to make more use of web-based teaching, and the relevant service departments seem reasonably committed to alleviating these problems.
- *Difficulties with syntax.* I see this as partly a communication problem: we should have stated more clearly that learning Maple syntax was a deliberate objective of the course, and that students would implicitly be assessed on it. (As the system rejects unparseable answers without penalty, the assessment effect is in fact small.) Nonetheless, further technical work is clearly called for. The system of automated syntax hints needs to be extended, and thoroughly tested on naive users under controlled conditions. There should be an easy way for students to send an email asking for help, with details of the question and the student's answer(s) included automatically.
- *Server crashes.* There is a hardware fault in our server, which caused it to crash quite regularly. Unfortunately, a large amount of time and effort was wasted looking for an explanation in software, which diverted me from more productive work on the system, and doubtless coloured students' overall perception.

Plans

After a period of very intensive work in the summer and autumn of 2001, AIM is currently being developed only sporadically. However, we hope that more active development will resume shortly.

A consortium involving the Universities of Sheffield and Birmingham, and Kings College London, is bidding for FDTL funds to support further development and dissemination of AIM. As part of this, we would hire a programmer to work full time for two or three years on improving and extending the system, and enhancing the framework for accepting code contributed by other developers.

One important improvement will be the development of a streamlined system whereby universities can organize, classify and share their banks of AIM questions, and possibly also import questions that were written for use with other systems.

As well as this development work, the bid (if successful) will support:

- A team of people who will demonstrate AIM in universities around the country, and help interested departments to set up their own servers.
- Maintenance of a central database of AIM questions.
- Workshops on how to write AIM questions, and on pedagogical issues related to computer aided assessment in general.

Your advert could be here!

This newsletter reaches teachers and course developers at all levels and in all areas of mathematics, statistics and OR in all higher education institutions in the UK. It also has an overseas mailing list.

The total circulation at present exceeds 2300.

Workshops, conferences, meetings etc will be given free publicity.

Advertisement rates till Dec 2002 VAT exclusive

	colour	mono	Quarter page (landscape or portrait)	75
Full page back cover	475		one page insertion	350
Full page inside cover	450		extra pages (each)	100
Full page in newsletter	400	300		
Half page	260	150		

Reductions for regular or bulk advertisers. Copy date for the next issue is 10 April 2002.

Further details from the editor and at

<http://ltsn.mathstore.ac.uk/newsletter/advert/adinfo2002.htm>