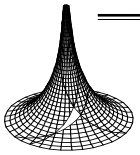


6/2 (2008), 395–408
tmcs@math.klte.hu
http://tmcs.math.klte.hu

Teaching
Mathematics and
Computer Science

Application of computer algebra systems in automatic assessment of math skills

Przemyslaw Kajetanowicz and Jedrzej Wierzejewski

Abstract. Mathematics is one of those areas of education, where the student's progress is measured almost solely by testing his or her ability of problem solving. It has been two years now that the authors develop and use Web-based math courses where the assessment of student's progress is fully automatic. More than 150 types of problems in linear algebra and calculus have been implemented in the form of Java-driven tests. Those tests that involve symbolic computations are linked with Mathematica computational kernel through the Jlink mechanism. An individual test features random generation of an unlimited number of problems of a given type with difficulty level being controlled at design time. Each test incorporates the evaluation of the student's solution. Various methods of grading can be set at design time, depending on the particular purpose that a test is used for (self-assessment or administrative exam). Each test is equipped with the correct solution presentation on demand. In those problems that involve a considerable amount of computational effort (e.g. Gauss elimination), additional special tools are offered in a test window so that the student can concentrate on the method of solution rather than on arithmetic computations. (Another obvious benefit is that the student is thus protected from the risk of frustrating computational errors). Individual tests can be combined into comprehensive exams whose parameters can be set up at design time (e.g., number of problems, difficulty level, grading system, time allowed for solution). The results of an exam can be automatically stored in a database with all authentication and security requirements satisfied.

Key words and phrases: CAS, teaching mathematics online, automatic assessment.

ZDM Subject Classification: U70, N80, R50.

Introduction

Typically, when a portion of math material is being taught, the student is first familiarized with given math concepts or methods followed (or accompanied) by suitable examples, then come math problems that the student is supposed to work out by themselves. In really few other areas does problem-solving play such an important role in the student's progress as in mathematics, and in few other areas is the student's active attitude towards independent work as crucial. For that reason, automatic testing in math have been drawing the attention of math instructors' and e-learning content developers for many years.

There exist some truly amazing solutions in that field. One has to mention exclusively math-dedicated ALEKS (see [2]). The creators of ALEKS seem to have successfully implemented the idea of full control of the student's progress, based on a logical system of graph representation of knowledge state. Guided by ALEKS, the student is simply not admitted to study a portion of material without prior demonstration that he or she has satisfied the necessary prerequisites.

Most of learning management systems (LMS) support assessment tools. For example, the well-known WebCT provides what is called *online quiz* – a utility that assists the teacher with creating questions of various types (single-choice, multiple-choice, formulae-based etc). From the point of view of math instruction, however, there are obvious limits of usability of such general-purpose tools.

The solution of an individual math problem usually breaks down into two or more steps (except, of course, simplistic problems involving the application of a single correctly identified formula or rule). When grading a math exam, the teacher typically devises a grading procedure for each individual problem in such a way that each part of the solution is assigned a given fraction of total score for the graded problem. In that way, the grading procedure reflects the solution process: the student that arrives at a partial solution earns the corresponding partial credit. It is desirable that automatic assessment should take such partial solution into account.

From the student's point of view, there are other important elements in the learning process, like the availability of the correct solution, for example. Traditional textbooks typically provide correct answers to all (or a part of) exercises. Hints or full solutions are limited to selected problems only. Electronic assessment content should offer extensive measures of such kind of help.

In this paper, the authors present their experience in creating and successfully implementing highly interactive math content equipped with automatic knowledge assessment. As of this writing, it is the fifth semester that the authors teach math courses, in which the whole administrative grading system is based on automatic e-exams.

In the sequel, the authors first briefly review their e-course in algebra with analytical geometry. The functionality of the automatic tests and exams is then discussed. In particular, the mechanism of automatic grading with partial credit is presented. Finally, the current work on applying CAS in automatic assessment of math problems involving symbolic calculation is discussed.

E-course in algebra – review

During the first 9 months of 2005, the authors designed and created an e-course in algebra with analytical geometry. The course employed and extended the idea of automatic knowledge assessment first used in an experimental lesson on quadratic function ([4]), in which the presence of highly interactive exercises and tests was made the authors' main objective. The course has been repeatedly taught in different versions (and continuously enhanced) with very good results both in terms of students' results and student's feedback.

The instruction itself was based on blended teaching. Traditional class meetings were held in accordance with the course curriculum (2 hours of lectures and 1 hour of recitation per week). The whole electronic part was available to students through an e-learning platform. The idea was to supplement rather (than substitute) the traditional teaching process with the e-course. In that way, the student both had access to electronic materials and exercises and to a "live" teacher. The presence of rich electronic, interactive material, considerably changed the style of lecturing. Standard, routine problems (e.g., computing the determinants) was almost entirely left to students for own practice through automatic tests. Accordingly, more in-class time could be assigned to advanced issues and applications.

For the sake of completeness, let us repeat the most important features of the algebra e-course (refer to [5] for details concerning the functionality):

- The math content of the course covers complex numbers, polynomials and rational functions, matrices and linear systems, analytical geometry in space, and conic sections. The content is logically divided into sharable content

objects (SCO), so the material can be easily aggregated into different organizations, depending on specific syllabi requirements. In everyday practice, it is IMS or SCORM packages that are being generated from individual SCO's, and then imported to an LMS (see [1] for details on SCORM philosophy).

- The material is offered in the form of Web pages supplemented with highly interactive Java-driven exercises and simulators.
- Over 150 types of algebra problems are supported in the form of interactive tests, varying from simple, drill-type exercises to quite sophisticated graphing problems or problems where the user is supposed to demonstrate mastery of a given method.
- Individual problems can be combined into comprehensive exams that can be used both for practice and as an element of the official grading procedures.
- The flexibility of automatic tests makes it possible to have the administrative grading system totally based on automatic assessment.

How automatic testing works

The basic idea behind the automation of math assessment that the authors employed in order to transform a math problem into an interactive computer-driven entity, was the notion of a *problem type*. By this the authors mean a math problem stated with a certain amount of generality in terms of specific problem data. These are a few examples of problem types:

- Find the roots of the given polynomial.
- Draw the subset of a complex plane defined by the inequality of the form $|z - z_0| < r_2$.
- Write the equation of the conic section, given its graph.
- Bring the given system of linear equations into the echelon form. Then determine the number of its solutions. Provide the solution, if it's unique. Provide any single solution, if there are infinitely many solutions.

An individual problem type gives rise to (theoretically) infinitely many problems of that type. As said above, the system has been designed in such a way that a given problem either can run as a single exercise or be combined with other problems to form a comprehensive exam.

Functionality of exercises

In single-exercise case, each individual problem type works as a Java applet that is capable of repeatedly offering the student a (theoretically) unbounded sequence of problems of that type. This happens each time that the student presses the **Next test** button in the applet window. The student work then goes along these lines:

- The student is required to solve the problem by hand and then enter the result(s) in the applet window through specially designed user interface elements: edit fields, spin controls, check boxes or radio buttons. The designer can additionally impose a time constraint on every test, forcing the student to provide the solution within prescribed amount of time.
- Some problem types involve graphing. In that case, the student is provided with dedicated graphing tools in the applet window, including virtual ruler and virtual eraser. The solution is graded based on the number of correct graphical objects that the student creates.
- Some problem types require the student to demonstrate the ability to perform certain transformations. Specifically, this applies to operations on rows or columns of matrices (e.g., the problem of bringing a matrix to lower triangular form). In that case, the student is provided with dedicated tools as well. In practice, the student specifies the operation to be performed, and the computational part is left to the program. In that way the student is released from the risk of accidental computational errors.
- After the student has provided the solution in the applet window and pressed the **Submit** button, the student's result is verified for correctness and graded. Depending of the configuration of a specific exercise, the solution is graded as "incorrect – correct", "incorrect – partially correct – correct", or is assigned a numerical score. We discuss this in greater detail later.
- At any time, the student has access to the correct solution through the **Solution** button in the applet window. Of course, requesting the correct solution before the student's own solution has been submitted results in the loss of the grading opportunity for the current problem. The solution can be presented "all-at-once" or in steps. Of course, the solution always addresses the actual problem that the student was given, not a problem type.

Additionally, the student has access to such utilities, as virtual calculator and online help on using the user interface elements.

Figure 1 illustrates the look of the applet window that offers exercises on solving linear systems by Jordan–Gauss elimination. Note the presence of additional tools for performing row/column operations.

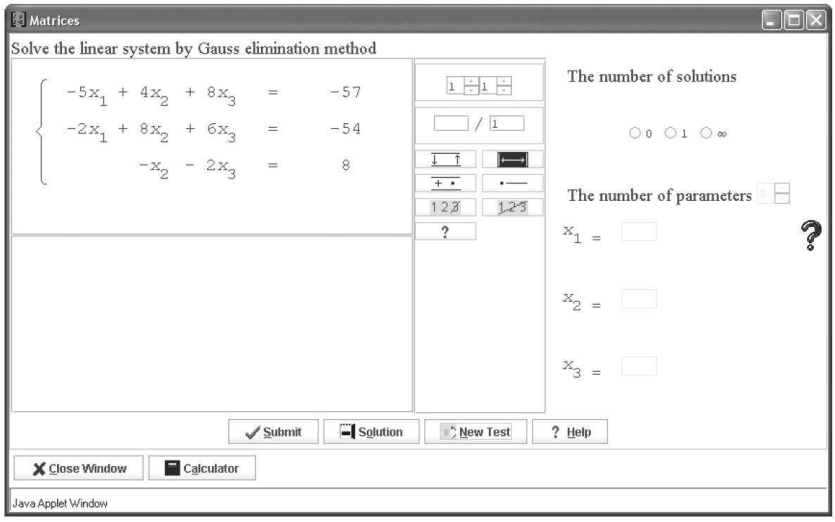


Figure 1. A single-problem applet window with matrix operations tools

Functionality of exams

Individual problem types can be combined to produce more comprehensive exams. An exam can consist of a variable number of different problems, and each problem can be assigned a different score. A specific set of exam problems is generated based on a designer's settings in a configuration file (without the necessity of intervention in the source code). Since the data for each problem is randomly generated, no two different students receive identical problems. Moreover, the designer can either predefine the exact type of each problem on the exam, or to specify the probability with which problems of given types will appear. In that way, the distribution of problem types within an exam is further diversified.

From the student's perspective, an exam again works as an applet window, in which a collection of problems is given in the form of tabs (see Figure 2). To work with a selected problem, the student just clicks a corresponding tab.

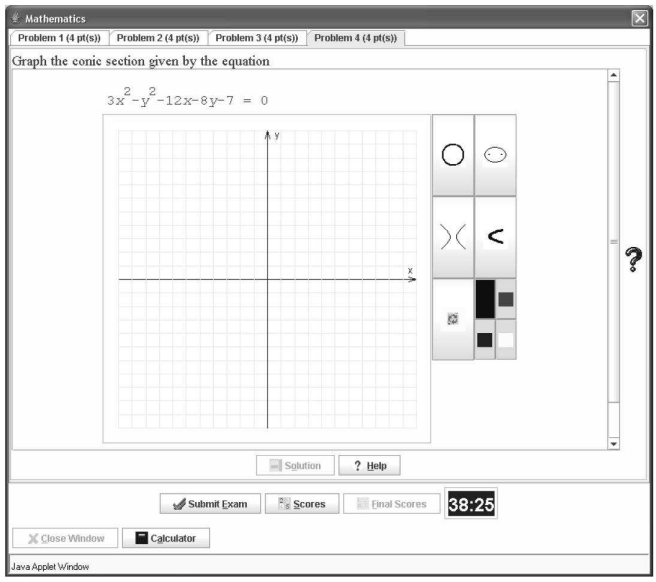


Figure 2. An exam applet window (a graphing problem displayed)

Typically, the student works out the exam problems in any order that he or she wishes. The exam ends when the student presses the **Submit** button or when the student runs out of the imposed time limit.

The total score for the exam is available to the student as soon as the exam ends. Also, the student has immediate access to step-by-step solutions of all the exam problems.

The exams are used for two purposes. They can serve as practice utility, enabling the student to carry out self-evaluations. In that case, the student can launch an exam as many times as he or she wishes, getting different problem sets each time. The other option is to configure the exam in such a way that the student's results are written to a database, with all necessary security and encryption conditions satisfied. In that case, the student can launch the exam once only, and within a prescribed time period.

The authors have been successfully carrying out e-exams as a part of administrative grading in several algebra courses, with good results. Refer to [5] for details.

Configuring exercises and exams

Obviously, there exist problems of substantially varying difficulty levels within the same problem type. To give the designer as much flexibility as possible, each problem can be configured through a text file that the Java applet reads each time the problem is generated. The designer thus has control over a number of parameters for each problem type.

For example, the problem of factoring a polynomial can be made easier or more difficult by setting the minimal or maximal number and multiplicity of roots, the presence or absence of complex roots etc. Likewise, the problem of solving a linear system can be configured according to the desired number of equations and/or unknowns, the number of solutions, or the range of coefficients. It is possible, for example, to have the latter problem set in such a way that the systems with no solutions, one solution or infinitely many solutions appear with given probabilities.

Another teacher's option is to control the "amount of care" that the student is given on the part of software. Especially greeted by students were two utilities: the completeness check and the initial correctness check.

The completeness check is a feature that warns the student in case that the solution is incomplete (e.g., some of edit fields are left blank). Typically, the student is given as many chances to complete the missing entries as he or she wishes.

The initial correctness check can be set by the teacher to give the student a chance to correct the wrong solution. When the student submits the exam for evaluation, all the solutions are initially verified for their correctness. The student is then informed about those problems whose solutions are not fully correct. The number of tries to correct the wrong solution depends on the teacher. In practice exams, the number of tries granted by the teacher is usually 2 or 3, while in administrative exams one try is usually granted.

Flexibility of grading system

As mentioned in the introduction, the solution process of a math problem often consists of steps, each step resulting in a partial result to be used in the sequel of the process. Also, the solution itself can have the form of a number, a collection of numbers, or a symbolic expression.

In the process of preparing an exam (whether traditional or electronic), the teacher decides on a maximal score for each problem in the exam problem set. Depending on the teacher's choice, scores can be identical for every problem, or can vary from problem to problem. The teacher's usual aim is that the distribution of scores best reflects the corresponding distribution of problems difficulty level.

Automatic grading has been designed in such a way that the teacher is given as much freedom as possible. The authors give a brief outline here, referring the reader to [6] for detailed description and examples.
Generally, three grading types can be associated with an individual problem.

Two-state grading: "correct" or "incorrect" is one of two possible grades.

Three-state grading: a student receives the grade equal to 0, 1/2 or 1, with 1/2 reflecting partial credit for a partial solution.

Score grading: a specified maximum number of points can be earned for the solution.

If the problem acts as an individual exercise, then the teacher can choose any of the three grading types described above. If the problem is a part of an exam, then the only teacher's option is the score grading. In that case, the teacher has to specify a maximum score M that the student can receive for the solution of the problem. The teacher's specification has the form of an entry in a configuration file.

When a student submits a problem solution for evaluation, the following process takes place:

1. A hard-coded *initial grading function* (strictly associated with a given problem type) is called. The function verifies the correctness of individual elements of the solution entered by the student. (Note that depending on a problem type, the solution can be provided in the form a single number or can have a more complex form). The function then returns a real number p from the interval $[0, 100]$, based on the function's internal algorithm. The number is referred to as *raw score*. The raw score is intended to reflect the "percentage of correctness".

2. Depending on the teacher's specification of grading, the actual raw score is transformed into the grade for the problem in question.

    a. In two-state grading, the solution is graded as "correct" if and only if the raw score equals 100. Otherwise the solution is graded as "incorrect".

b. In three-state grading, the solution is graded as "1", "1/2" or "0" depending on the subinterval of $[0, 100]$ that the raw score falls into. If the raw score equals 100, then the solution is graded as "1". If the raw score p satisfies $50 \leq p < 100$, then the solution is graded as "1/2". Otherwise the solution is graded as "0".

c. In the score grading, the following algorithm is applied. Let M denote the maximum score that the teacher has set for the solution. If p is the raw score and E stands for the greatest integer function, then the actual credit that the student earns is computed from the formula

$$p = E\left(\frac{p \cdot M}{100}\right)$$

For example, if the initial grading function has returned $p = 40$ and the teacher has set $M = 5$, then the actual score that a student gets for the solution is $P = 2$.

Employing CAS

The above-described automatic assessment mechanisms work very well for the problems where symbolic computation is practically limited to algebra. Obviously it does take a lot of work to encode appropriate algorithms (e.g. for matrix or polynomial transformations), let alone the development of sophisticated graphing tools, yet no CAS is absolutely necessary.

On the other hand, when it comes to calculus problems, then a dedicated symbolic computational kernel is required. As of this writing, the work is going in two directions. One approach is to combine the well-tested Java solutions with the power of webMathematica. The other is to use Mathematica for generating large data sets for specific problem types that are further used in Java-driven tests.

webMathematica approach

The Jlink utility of Mathematica (and, in particular, webMathematica) makes it possible to communicate with the Mathematica kernel through Java classes. The following solution has been developed that both makes extensive use of existing mechanisms and utilizes the power of CAS.

- The data for an individual problem is generated by the Java applet. The problem is presented to the student.
- The applet accepts the student's entry(ies). (The student is first instructed how to enter symbolic expressions in a string form). The applet then sends the student's input to Mathematica for initial evaluation.
- Mathematica evaluates the student's input into a math expression in traditional math form and returns the result in the form of a graphics file (GIF). The student is prompted for the confirmation that the returned expression coincides with the student's intended input.
- After the student's confirmation, the applet sends both the problem data and the student's input to Mathematica for actual evaluation of correctness.
- Mathematica compares the student's solution with the correct result and reports back to the applet.
- The applet applies appropriate grading.
- Additionally, Mathematica is used to generate and send back other elements, e.g. graphs or expressions used for the presentation of the correct solution.

Summarizing, the Java applet performs all the work, except two elements: establishing the correctness and producing certain elements of presented solution.

The described combination of Java and Mathematica is actually invisible to the teacher in terms of configuring individual questions, designing grading systems etc.

Quiz-type approach

The above-described method has one downside. Namely, it requires a 100%-reliability on the part of a Mathematica server. In the traditional, solely Java-based assessment mechanisms, the whole work is done on the client's side (with the exception of database operations when an administrative exam is being taken): once a Web page with the applet has been successfully downloaded by the browser, no service is requested from the server. In contrast to that, communication between the applet and the Mathematica server is initiated each time that the student requests another problem or submits the solution. Without ruling out the described combination of Java and webMathematica, a parallel approach has been developed and successfully used for considerably many types of problems in calculus.

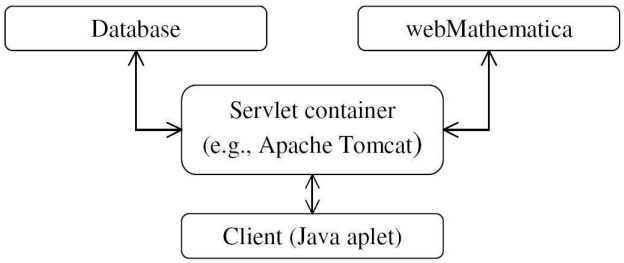
The method mimics (in a way) the online quiz utilities seen in various LMS's. Specifically, all the data necessary for the problem formulation, the correct results corresponding to respective data sets as well as elements of the correct solution to be presented, are encoded in an xml file. When a student requests a problem of a given type, a Java applet looks up the xml file and acts accordingly. The power of CAS is used here to generate necessary data sets together with corresponding correct answers.

An obvious advantage of that approach is that CAS is only needed on the problem design stage. No need for CAS emerges at runtime, thus eliminating delays and decreasing network traffic associated with the student's work. The advantage is at a price, however: an xml file that controls a single problem type can be very large in size, especially when the problem involves graphics.

Summarizing, it seems that a combined approach is optimal. For some problem types the use of a distant CAS kernel is inevitable. For other, an xml-driven solution serves its purpose well.

Technical implementation of the system (outline)

The implementation of the automatic assessment system makes use of Java technology combined with database connectivity (JDBC) and (in the case when webMathematica services are needed) with the JLink feature of webMathematica. The diagram below illustrates the general conceptual framework of the implementation.



An individual Java applet corresponds to an individual problem type, the notion that was introduced earlier in the paper.

A Java class `AbstractApplet` controls the whole process of selecting specific data for the problem, delivering the problem to the student in a user interface, grading the student's input, and presenting the correct solution on demand.

In the `AbstractApplet` class, the following 3 methods have to be implemented (the actual implementation is strictly related to the specific problem type in question):

- `abstract void newProblem` ( ) – controls the process of selecting the problem data (e.g. the expression to be differentiated), and the process of correct solution presentation.
- `abstract void showProblem` ( ) – prepares the presentation of the problem, and implements all the necessary elements of the user interface; note that depending on the problem type, the user interface can vary from simple number-input collection of fields to very sophisticated user interface with specialized tools for drawing etc.
- `abstract void gradeProblem` ( ) – compares the student's input with the correct answer(s) and grades the solution according to the grading algorithm described earlier in the paper.

We wish to emphasize that the above-described methods are the only methods that actually need to be implemented while designing a new problem type. The framework is designed in such a way that it is more a problem for a mathematician than for a programmer to implement the three methods. The complexity of implementation depends exclusively on the mathematical complexity of the problem type itself.

Conclusion

It has to be mentioned that as many problem types as have been already supported, automatic tests and exams have their obvious limitations. For example, the authors do not see a sensibly effective way to automate problems that consist in proving math statements. The automatic grading also has its downsides, especially when a single computational student's error results in a substantial loss of a part of credit. (On the other hand, the afore-mentioned initial correctness check utility has practically eliminated students' complaints regarding that issue).

Having said that, the authors have gathered enough evidence of effectiveness when teaching with the use of automatic assessment to strongly believe that further expansion is worthwhile. The grounds for that belief vary from substantially better students' results to students' enthusiastic feedback. Towards the end of every semester, the students are given an anonymous survey, in which they give their evaluations of that electronic form of learning. One of the most emphasized

facts on the students' part is the opportunity of learning anytime, anywhere, with continuous access to assessment utilities.

References

[1] ADL Technical Team, *Sharable Content Object Reference Model (SCORM)*, Version 1.2, Advanced Distributed Learning, 2001, `www.adlnet.org`.

[2] E. Cosyn, J - P. Doignon, J - C. Falmagne and N. Thiery, *The Assessment of Knowledge*, in: *Theory and Practice*, 2004, `www.aleks.com/about/Science_Behind_ALEKS.pdf`.

[3] J. Engelbrecht and A. Harding, Teaching Undergraduate Mathematics On the Internet, *Educational Studies in Mathematics* **58** (2005), 235–252.

[4] P. Kajetanowicz and J. Wierzejewski, *E-lesson on Quadratic Function. A Step Towards an Online Remedial Math Course*, in: *Proceedings*, 5th International Conference Virtual University, Bratislava, Slovakia, December 2004.

[5] P. Kajetanowicz and J. Wierzejewski, *E-learning in College Mathematics – an Online Course in Algebra with Automatic Knowledge Assessment*, in: *Proceedings*, 6th International Conference Virtual University, Bratislava, Slovakia, 2005.

[6] P. Kajetanowicz and J. Wierzejewski, *Automatic assessment of math skills*, in: *Proceedings*, 7th International Conference Virtual University, Bratislava, Slovakia, 2006.

PRZEMYSLAW KAJETANOWICZ and JEDRZEJ WIERZEJEWSKI
INSTITUTE OF MATHEMATICS AND COMPUTER SCIENCE
WROCLAW UNIVERSITY OF TECHNOLOGY
POLAND

E-mail: `przemyslaw.kajetanowicz@pwr.wroc.pl`

E-mail: `jedrzej.wierzejewski@pwr.wroc.pl`

(Received September, 2007)