Testing and Assessing Mathematical Skills by a Script Based System

Thomas Risse

Testing and Assessing Mathematical Skills by a Script Based System

Thomas Risse

Institute of Informatics and Automation, Hochschule Bremen, Germany

Key words: *Test and Assessment, Mathematics, Computer Algebra System, Learning Platform*

Abstract:

Especially in the mathematical education there seems to be a need for strict drill and practice exercises. Such training cannot be met by traditional means. Therefore, a computer program is needed which generates parametrisable questions/problems which the students have to answer/solve and which at the same time assesses the answers/solutions of the students.
We designed such a system as an interface between the learning platform ILIAS and MATLAB/Maple as the symbolic mathematics engine. We demonstrate how by scripting a wide variety of parametrisable problems can be generated and how MATLAB/Maple can be used to checks the correctness of the solutions of the students. Examples illustrate the potential and limitations of this approach.

1 Necessity of Automatically Assessed Mathematical Tests

Students starting for example engineering degree courses show more and more severe deficits in their mathematical skills over the years – a phenomenon monitored by many observers who worry increasingly about the prognosis of the general learning outcome of these weak students.

In the mathematical education there seems to be a special need for strict drill and practice exercises which we consider necessary to enable students to work their way from simple calculus (with most of the time unrealistic problems) to the more demanding tasks like modelling and simulating real problems. Such exercises have to be marked, and there has to be reasonable feedback to the students in case they fail a task.

The high demand for such training cannot be met by traditional means. Therefore, a computer program, more precisely a carrier system is needed which generates parametrisable questions and problems which the students have to answer and to solve – a computer program which at the same time assesses the answers and solutions of the students, not only by a right/wrong decision but preferably also with some sensible feedback. Of course, there are commercial solutions, e.g. [7], but also for economical reasons we strive for an open source solution.

We designed such a system as an interface between the open source learning platform ILIAS and the computer algebra system MATLAB/Maple (to be replaced by Octave/GiNaC later on) as the symbolic mathematics engine. We demonstrate how by scripting a wide variety of para-

metrisable problems can be generated and how MATLAB/Maple checks the correctness of solutions delivered by the students.
To exploit the power of the symbolic mathematics engine offers the following advantages:

- There is practically no limit to the areas which are addressed by questions and problems, algebra, calculus, statistics, etc. comprising literally all areas of scientific computing.
- All parameters like constants and variable names can be generated at random.
- Initialising the seed of the pseudo random number generator depending on the date of some test or examination makes it even possible to generate the same questions and problems for a group of students who are scheduled to be under examination on a certain date,

Students formulate their answers and solutions using the prevalent syntax for mathematical expressions which is established in MATLAB/Maple as well as in more or less every programming language.

We will illustrate the features with relevant examples.

2 Interfacing

In order not to reinvent the wheel, we used the relevant features of our university-wide learning platform ILIAS on one hand and of the computer algebra system MATLAB/Maple available in our department which is available at the Department of Electrical and Electronic Engineering and Computer Science of Hochschule Bremen on the other hand.

2.1 The Computer Algebra System MATLAB/Maple

MATLAB is a numerical computing environment and programming language. Although it specializes in numerical computing, the optional Symbolic Math Toolbox of MATLAB interfaces with the Maple symbolic engine making MATLAB/Maple a full computer algebra system like Mathematica, MuPad etc.

2.2 The Learning Platform ILIAS

ILIAS (Integrated Learning, Information and Work (Arbeit) System) [4] is an Open Source e-Learning Management System for development and deployment of web-based learning content. The university-wide learning platform AULIS of Hochschule Bremen is based on ILIAS. Specifically, ILIAS provides authoring tools for creating tests, administration of existing questions in pools, evaluation and grading routines. ILIAS stores, manages, displays and statistically evaluates test results. Thereby, ILIAS supports all standard types of test and assessment, like single/multiple response, fill-in-blank, matching, ordering, and image maps can be used in online self-tests and online examinations.
In addition to the standard test types, ILIAS offers to implement questions by JAVA applets [16]. This feature is crucial for the implementation of script based test and assessment schemes for mathematics as we will see below.

2.3 Interfacing between ILIAS and MATLAB/Maple

In JAVA we developed the interface program IliasMatlab which uses the library JMatLink [9] to access MATLAB/Maple via Java. (Another option is described in [5]).
It seems near at hand to implement generation and assessment of tests in MATLAB. Thus, each test is developed as one MATLAB m-file which without argument generates the test and with argument assesses the answer to the generated test. Hence, the m-file returns either a string consisting of the text of the test or the result of the assessment of the answer to the test. Below we sketch the generic structure of such m-files in Pidgin MATLAB

```
function resultstr = Testname(answer)
    if (nargin=0)                       % if no argument is given then generate test
                                        % generate symbol names, parameters or
        resultstr = text_of_test;       % coefficients at random
                                        % and then, construct the text of the test!
    else                                % if argument is given then assess test
        if (answer is mathematically correct)
            if (answer has the correct form)
                resultstr = 'ok';
            else
                resultstr = 'false';    % or some other appropriate feedback
            end
        else
            resultstr = 'false';        % or some other appropriate feedback
        end
    end
end
```

Of course, this generic structure does not show features like maintaining the workspace corresponding to both a test and a student running the test, like exception handling, like localisation, etc. Also, to generate appropriate feedback to answers which are not mathematically or not formally correct is rather difficult because it involves analysis of the answer in order to figure the intended meaning and to diagnose what went wrong.

2.4 Test Generation and Assessment

The following steps show how a certain test is generated and how the answer to it is assessed.

1. A student selects a certain test, either as self-test or in an examination.
2. Via IliasMatlab the m-file corresponding to the selected test generates the text of the test which is presented in ILIAS.
3. Via IliasMatlab the students answer to the test is passed as an argument to the m-file corresponding to the selected test which in turn returns an 'ok' or some other feedback when assessing the answer.
4. Via IliasMatlab this feedback is presented to the student by ILIAS. ILIAS marks the test as passed if the feedback is the simple 'ok' and as failed otherwise.

That way, we can take advantage of all features ILIAS provides to perform, manage and evaluate tests and examinations.

2.5 The User-Interface (between User and ILIAS / liasMatlab / MATLAB/Maple)

The last interface to be considered is the user interface.
All algebraic expressions are to be entered just like in most programming languages (which for sure is adequate for students at least in computer science), i.e. products have to be entered as *factor1*factor2*, fractions as *numerator/denominator*, powers and roots as *base^exponent*. Likewise, all elementary functions are available represented exactly as in many programming languages, e.g. exponential function exp, natural logarithm log, trigonometric functions sin, cos, tan, cot with their inverse functions asin, acos, atan, acot and the hyperbolic functions sinh, cosh, tanh, coth with their inverse functions asinh, acosh, atanh, acoth. In MATLAB/Maple there is also the numeric constant pi.

3 Scripting

We tested the potential of our approach by implementing the tests given in [15], just one of the very many books providing first-year-students with question/answer type problems in basic mathematics, i.e. simple calculus, basic linear algebra, elementary geometry etc.

3.1 Potential

Due to the power of the symbolic computation in MATLAB/Maple, the student's skills to perform most computations in algebra and calculus can adequately be tested and assessed by IliasMatlab and the type of scripts described above. Certainly, skills for e.g. first-year-students include

- manipulation of algebraic expressions like expanding and factoring
- arithmetic of fractions including operations like expanding and cancelling
- arithmetic of powers and roots
- computation of logarithmic expressions
- solving (linear) equations in one variable or several variables
- solving quadratic equations in one variable

And of course, all problems which students can only solve if they have the basic skills listed above can be packaged into math story problems. Then, the students can show their ability to analyse problems stated verbally, to represent the problem in some abstract model and to apply some procedure to come up with the hopefully correct result.

3.2 Limitations

As outlined in the generic script or m-file, assessment consists of two checks, a mathematical correctness check and a formal correctness check:

1. to check the mathematical correctness, i.e. answer == target answer
2. to check the formal correctness, i.e. check whether the answer has some target form, format or representation, e.g. a mathematical correct answer to the problem 'write the natural number 6 as product of its prime factors' is also 1+5 whereas the correct formal answer is 2*3 or 3*2.

The first check is performed easily by any computer algebra system. The second check is more intricate. In the example above one can take to

- comparing factorisation of the computer algebra system with the factorisation of the student whereby taking commutativity into account and/or
- checking that the expression given by the student exhibits as only operation a multiplication (which is not a multiplication by 1) and powers by 1 covering admissible answers like 3^1*2^1.

The formality check is difficult to cope with as long as it is not possible to transform an answer into some standard representation (stripped from white space by transforming any subexpression " expr" and "expr " to "expr", stripped from multiple parentheses by transforming any subexpression "((expr))" to "(expr)" etc) without at the same time changing the structure of the expression as to some extend the very input operation itself or as thoroughly the MATLAB function simplify do. The best will be to provide a function, say rpn which converts its argument into reverse polish notation (without parentheses) and the inverse, say npr of this function which converts any expression in reverse polish notation into its standardised traditional form (with only the necessary parentheses). Then, by npr(rpn(answer)) each answer expression is transformed into a standardised representation without superfluous white space and superfluous parentheses and without unwanted simplification by the compute algebra system [14].

3.3 Useful Feedback

Because of the many ways the students might fail a test a full-blown intelligent tutoring system is needed to generate useful feedback. Within our limitations we propose as a compromise to monitor student's answers test by test in order to identify frequently made errors. Comparison of an erroneous answer with the frequently observed flaws offers the chance to give helpful comments at least in case of frequently made errors [14].

4 Conclusion and Outlook

The advantages of our approach are obvious:

- a variety of tests parameterized by symbol names and coefficients which can be determined at random (for examinations on a certain date all students run the same test if the seed of the algorithm to generate the parameters is controlled by the date)
- automatic assessment
- test outcomes are stored, displayed and statistically evaluated by the learning platform

Also, the disadvantages of our approach are obvious as well:

- Right now, problems are stated and students have to phrase their answers as one would do in a programming language.
- Only pure arithmetical or calculatory skills can be assessed (cp [11], [12]) even though problems might be packaged as math story problems. Obviously, problems with graphical input as answers, e.g. problems like '*sketch the function graph of the parabola*', are excluded [13].

As of 2007, ILIAS in the current version 3.8 supports LaTeX elements in learning modules, forums and especially in tests, i.e. it is possible to represent mathematical formulae in the proper way and with the quality (La)TeX provides [6]. Combining this feature with Ilias-Matlab would allow presenting the mathematical text of tests in an adequate way. However, students still have to enter their answers in programming style as long as there is no equation editor in ILIAS.

Octave [10], [2] the open source counterpart of MATLAB, does not feature symbolic computations. However, there is the open source library GiNaC [1], [3]. Substituting MATLAB/Maple by Octave/GiNaC will render the implementation of our approach an entirely open source application.

References:

[1] Bauer, C.; Frink, A.; Krekel, R.: GiNaC www.ginac.de/csSC-0004015.pdf
[2] Eaton; J.W.: GNU Octave Manual; Network Theory Ltd 2002, ISBN 0-9541617-2-6
[3] GiNaC is Not a Cas www.ginac.de
[4] ILIAS Learning Management www.ilias.de
[5] Klimke, A.: How to Access Matlab from Java; Institut für Angewandte Analysis und Numerische Simulation, Universität Stuttgart 2003 http://preprints.ians.uni-stuttgart.de/downloads/2003/2003-005.pdf
[6] LaTeX/TeX Users Group web site www.tug.org
[7] Maple T.A.: Online Testing and Assessment www.maplesoft.com/products/mapleta
[8] MATLAB, the Language of Technical Computing; s. www.mathworks.com
[9] Müller, S.: JMatLink – Connect Matlab and Java; http://jmatlink.sourceforge.net
[10] Octave www.octave.org
[11] Risse, Th.: E-Tests – What Can be Assessed in E-Learning Applications; ICL 2002, Villach 25.9.–27.9.2002, in A. Auer, U. Auer (Eds.): Interactive Computer Aided Learning – Blended Learning; kassel university press, 2002, ISBN 3-933146-83-6 www.weblearn.hs-bremen.de/risse/papers/ICL2002
[12] Risse, Th.; Grüter, B.; Loviscach, J.; Vatterrott, H.-R.; Wilkens, U.: teleVISE – a Collaborative (Tele-) Tutoring Environment to Support Education e.g. in Mathematics ; ICL 2003, Villach 24.9.–26.9.2003, in M. Auer, U. Auer (Eds.): Interactive Computer Aided Learning – Learning Objects & Reusability of Content; kassel university press, 2003, ISBN 3-89958-029-X www.weblearn.hs-bremen.de/risse/papers/ICL2003
[13] Risse, Th.: Assessing Graphical Sketches in e-Learning Systems ; ICL 2005, Villach 28.9.–30.9.2005, in M. Auer, U. Auer, R. Mittermeir (Eds.): Interactive Computer Aided Learning – Ambient and Mobile Learning; kassel university press, 2005, ISBN 3-89958-136-9 www.weblearn.hs-bremen.de/risse/papers/ICL2005
[14] Risse, Th.: Using CASs to generate and mark mathematical question/answer type tests; Proc. 25th Scientific Colloquium '*Science for Practice*', Schweinfurt 15.–16.10.2007 to appear www.weblearn.hs-bremen.de/risse/papers/Kolloq25
[15] Schäfer, W.; Georgi, K.; Trippler, G.: Mathematik-Vorkurs – Übungs- und Arbeitsbuch für Studienanfänger; Teubner 2006
[16] Schottmüller, H.: ILIAS Test & Assessment Documentation; 2007 www.ilias.de/docu/data/docu/lm_data/lm_8782/Test_Assessment_38.html#java_applet_questions

Author:

Prof. Dr. Thomas Risse
Institute of Informatics and Automation
FB E-Technik und Informatik, Hochschule Bremen
Flughafenallee 10, 28199 Bremen, Germany
risse@hs-bremen.de