Electronic Linear Algebra Trainer: Towards a system configuration

W.Pasman

Technical report, Delft University of Technology,
V1.1

Introduction

Many students have problems learning linear algebra (**LA**). A lot has been written about what causes this and the nature of the problems. Unfortunately, "it is doubtful that anyone really understands what mathematical concepts students know or how they learned them" [Day01]. We get the impression that problems that students have come from a combination of factors: problems with some pre-linear algebra concepts and the foundations of mathematics, underestimation by students of the complexity until it's too late to catch up, and inefficient learning procedures.

A lot of mathematics problems rise already at early school years [Kilpatrick00]. Kilpatrick et al. argue that a balanced development of five strands of knowledge is essential when learning mathematics. Those five strands are conceptual understanding involving a functional grasp of the mathematical ideas, fluency witth the required mathematical procedures, ability to formulate a problem in mathematical terms ('strategic competence'), adaptive reasoning that enables proper combination procedures, and the acception of mathematics as a useful tool ('productive disposition'). It may well be that part of the students have an under-developed strand but that they do not recognise it as such or do not know what to do about it.

Underestimation of the linear algebra may result in insufficient practice with the basic procedures and ideas. This may be caused by several factors: for example the student may be already familiar with matrices and judge he does not need to pick up training yet, or he may be fooled by the apparently trivial proofs and simple procedures involved with basic linear algebra, not recognising the strategic competence required to create these proofs. Once the course proceeds further, it gets harder to follow the steps because of the missing skills, although the proofs still look deceivingly trivial.

Finally, the student may have ineffective training and learning and training procedures. For example, the student may lack the overview to keep an eye on his progress, which makes it hard to select the appropriate theory and exercises to work on. Or he may never have learned how he learns best, something that is rarely if ever teached.

Unfortunately, the short scope of this project does not allow us to dig further into the real causes and backgrounds of the problems.

To get further, we assume that insufficient and inappropriate learning and practice methods, combined with an already weak mathematical basis, are central to the problems. And that a good training system might be able to alleviate the problems.

A competent linear algebra tutor would be a linear algebra expert with good teaching and explanation capabilities, coupled with excellent skills in recognising the student's problem. He should be able to set challenging questions that focuses on sense making and problem solving as well as skill building [Kilpatrick00]. He might be doing part of the calculation for the student, to make him concentrate on his weak points. To understand how students are reasoning and where they get stuck may make clear the conceptual difficulties they have, which is much more effective than just giving an answer [Day01].
Some of these functions could be supported with an e-learning system. But in this report we restrict ourselves to systems that support the student's training. We think that an electronic trainer is a realistic and useful goal: existing electronic mathematics trainers have been shown to dramatically improve student performance (for instance for calculus, [Beeson99]).
Generally, there are four approaches to training systems:

(1) Quizz-systems where the student types his answer and the system compares his answer with the correct answer. Especially in mathematical context, a wide variety of correct answers is possible, and in some cases a complete computer algebra system is employed to do this comparison.
(2) The use of computer algebra systems, either to illustrate and facilitating experimentation with the course contents, or as an oracle that can give the right answer to problems.
(3) Proof support systems, where the student works his way to the solution by indicating which rules have to be applied in which order.
(4) Proof support systems with hints. This works just as a proof support system, but additionally the system can indicate how to proceed and recognises when the student is finished.

Section 1 discusses current approaches to electronic tutor systems, to see if something approximatly fitting our goals is readily available. Section 2 discusses how we could make a tutor system, using existing technology as much as possible.

1. Electronic tutors

A huge number of electronic tutor and training systems have been proposed and implemented. We are focusing on linear algebra, but to find the best approach to an electronic tutor system it is better to widen the scope to mathematic tutor systems.
As discussed in the introduction, we can categorize the available tools into a few categories: Quizz systems, computer algebra systems, proof support systems, and proof support with hints. We mention examples and discuss each of the categories in the light of our goal, supporting students learning linear algebra.
All these systems can be combined with an extensive scoring system, keeping track of the performance of all students, and supporting negotiation between students and between a student and a (human) support team. In some cases bookkeeping has even become the major component of the whole tutor system. We will not discuss these bookkeeping features.

1.1 Quizz Systems

Quizz systems show a problem to the student, and ask him to type the answer. The system then judges the answer. The student may get a few retries if his answer was judged wrong. If the answer is judged correct, the system cycles through some more questions, to end with a final score.

The most straightforward approach to an electronic quizz system uses plain text with pictures to present the question, and a very simple program that compares the student's answer to the correct answer. For multiple choice questions, the comparison is trivial. For numerical answers, a small tolerance in the numeric value may be added. Such a quizz system can easily build in a web page with a bit of Java or Javascript code. More recently, also Adobe Acrobat files enable embedding of short programs, and quizz systems based on Acrobat have been built [Story03] (Figure 1). There has been a good effort at recognising alternative mathematical answers (swapping parts in an equation, using x^-1 instead of 1/x, etc).

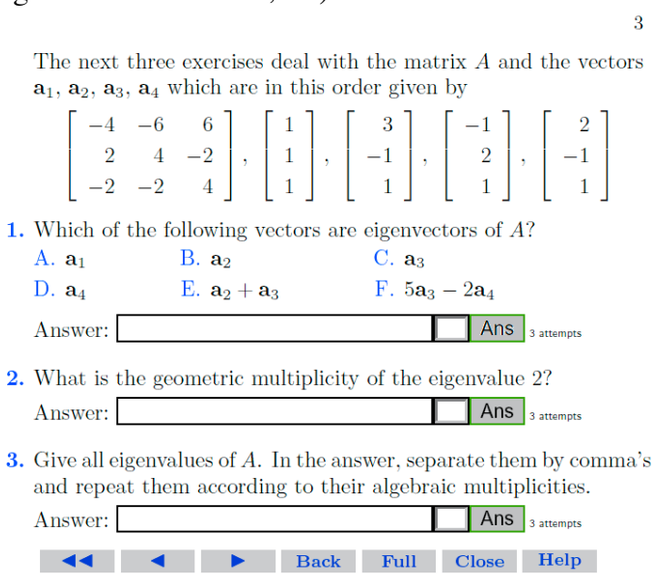


Figure 1. Snapshot of a linear algebra quizz in Acrobat, by Joost de Groot at Delft University of Technology. Students can enter numbers, lists and matrices in plain text style (e.g., "`{{1,2,3},{4,5,6}}`"), and letters (e.g., "`a,c`").

Maple Testing and Assessment Tool (Maple TA) is a quizz system with extensive quizz development and student progress and rating system [MapleTA05]. Maple can be used under the surface, to generate questions and do typesetting of formulas. Maple also allows to parse and rewrite mathematical answers, which is essential for open answers as a mathematically correct answer does not have a unique form. A large amount of material is available for Maple TA.

Lyryx ILAW (Interactive linear algebra on the web) presents lessons with animations [Lyryx02]. It's a bit dubious whether some of their animations are really instructive (e.g., animated arrows, growing slowly till they reached proper length) but the idea is probably that animations motivate students and keeps them more focused. As with Maple TA, there is extensive instructor and student progress/score database, and a forum to discuss questions with students and instructors. ILAW focuses on an extensive report on the

user's quizz, rather than giving tips during the quizz itself (Figure 2a-c). For example, it may indicate that a plane has the right orientation but does not go through the requested points. This is useful and stated in a positive way. However frequently, the report just says "your answer is more than 0.05 of its correct value", which probably just as useful as "wrong" for a student.

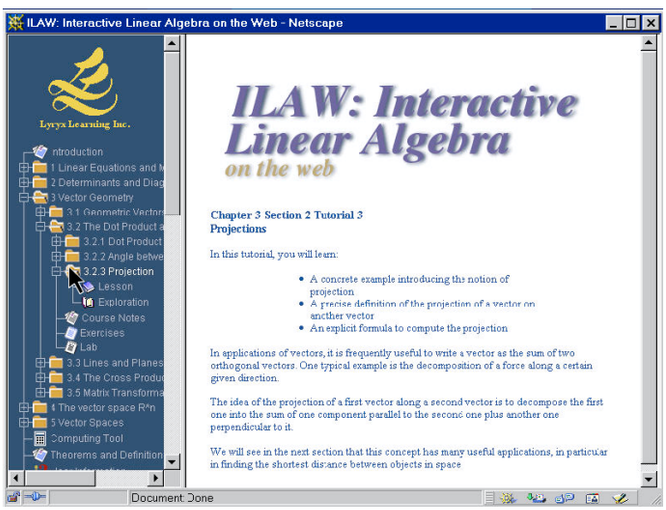


Figure 2a. Lyryx ILAW lesson hierarchy (left) and lesson page (right)

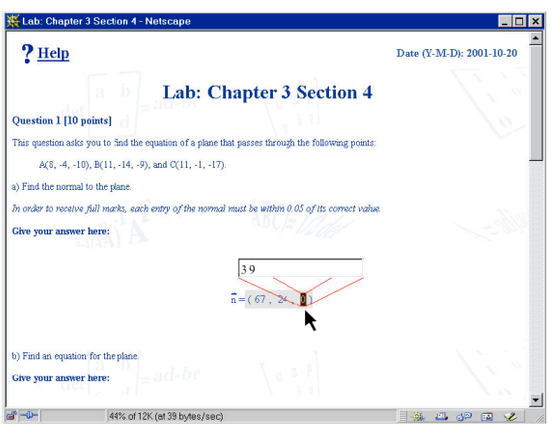


Figure 2b. Lyryx ILAW "lab session" (quizz) where user is filling in the third number of a 3-vector.

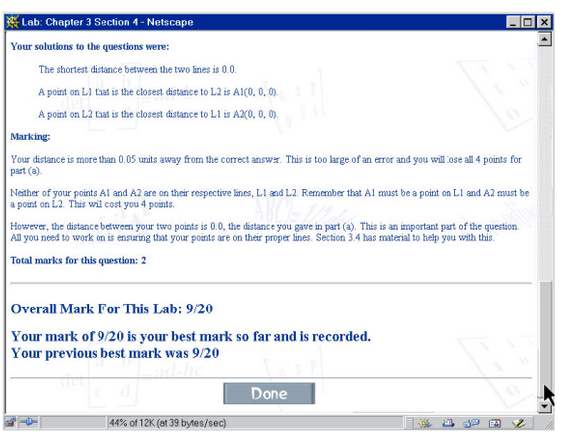


Figure 2c. User reading the report on his lab session.

Transmath [Transmath02] contains both lessons and exercises (Figure 3). It has a number of linear algebra components, including determinants, inverses and eigenvalues. The early reports on Transmath indicate that they put high emphasis on sympathetic, discerning, corrective interaction instead of right/wrong answers. Unfortunately their demo system fails to install itself so we could not test this further. From the snapshots the typesetting looks a bit simple, and more complex formulas involving square roots and fractions might be a bit problematic.

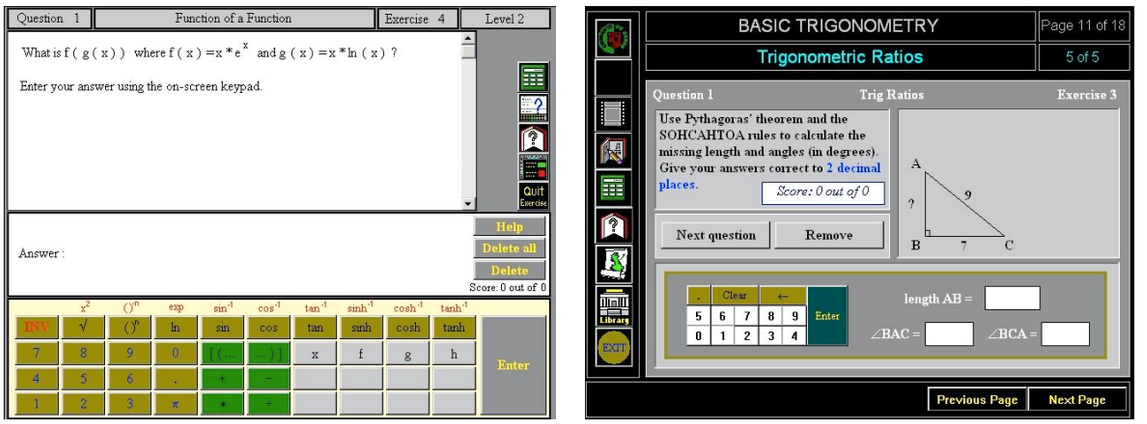


Figure 3. Two snapshots from Transmath.

The Etude/Sonate system, developed at Delft University of Technology [Etude05] is another quizz system. Six types of quizz questions are possible: multiple choice, maximal-two-answers questions, open numerical question, and hotspot (click on the map where you think that the requested item is). We did not see demos but from the manuals it seems similar to the ILAW system.

ActiveMath [ActiveMath05] is also contains lessons combined with quizzes (Figure 4). It behaves a bit weird in recognising alternative expressions, for instance in one example 100/4000 is judged wrong while 1/40 is correct. But just a few exercises further, writing (600-450)/(2000-1800) gives on-spot result "Okay, you've inserted the values correctly into the formula. Yet you should now also evaluate this.".

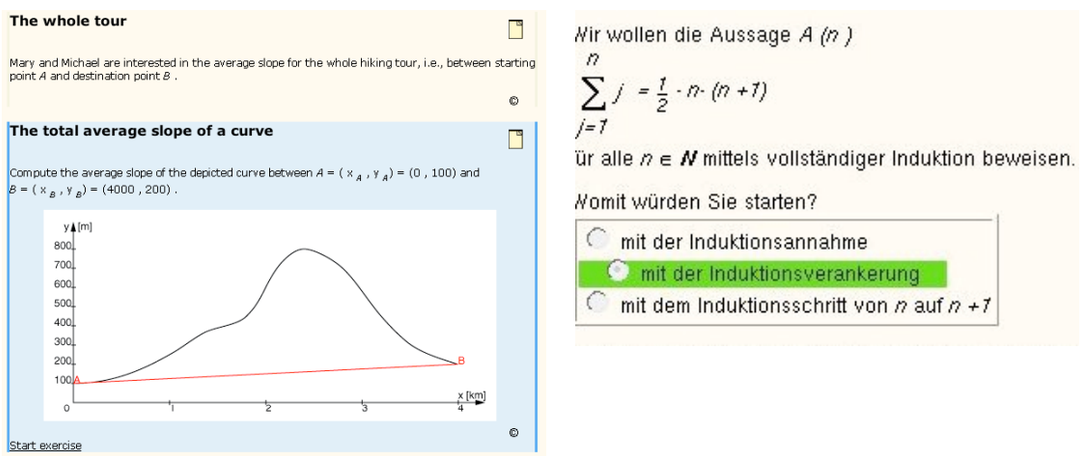


Figure 4. Two snapshots from the ActiveMath quizz system.

A problem with most quizz systems is that they are not really up to the task of handling mathematical answers. In some cases, a simple numeric answer may be possible. But in many cases, a mathematical answer involves either a matrix or vector of answers, or a symbolic answer. Nevertheless some of them, especially the acrobat based systems, have surprisingly effective judgement engines.
A more crucial drawback of all quizz systems is that they do not follow the student closely. The student does his work outside the view of the system (e.g., on paper), and

enters only his answer. This makes it very hard to help the student. Sometimes a "hint" button is available, but the hint must just guess about what problem the student really has. Usually, the hint only focuses on a special trick that the question adresses, and does not help a student if he is stuck on something else. Sometimes, a particular answer can be associated with a certain error. But this is is not a reliable method.

1.2 Computer Algebra Systems

A computer algebra system is a computer program supporting symbolic manipulations of mathematics. Usually, these systems can also do numerical manipulations, even at an arbitrary number of digits. Well known are Maple [Maplesoft05] and Mathematica [Wolfram05]. There are also several free computer algebra systems, the most interesting being the open source system Maxima [Maxima04]. Some recent pocket calculators also include increasingly powerful methods and might be usable for large parts of linear algebra, although generally they still lack power when it comes at symbolic manipulation. Some institutions use computer algebra systems to support the lessons and lab exercises: this is called courseware (e.g., [WolframCourseware05] and [UTCourseware05]). Courseware probably invites students to play with the course information, more than they would if they had to do lengthy calculations by hand or to enter the equations themselves. Alternatively, a computer algebra system can be used to check the answer and the steps in a derivation. It is like an expert only giving a result but not how he found the answer.
The first problem comes with understanding the result. Often derivation traces can be requested, but interpreting these may be non-trivial (Figure 5). And more seriously, the rules that were applied may be very general, high-level rules using complex algorithms, that the student does not know yet or does not need to know. Most humans would take another approach to reach the result. This makes computer algebra systems less useful for students to get a tip on how to proceed on a problem.

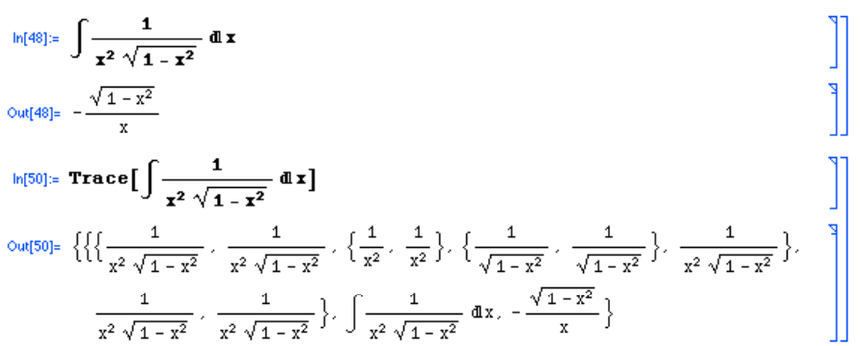


Figure 5. Mathematica fragment on solving an integral (above) and trace of the applied rules (below). A human would use trigonometric substitution here (Figure 9b). Formulas are entered with keyboard, with formula template buttons or by copying from the electronic manuals.

Another problem with the use of computer algebra systems is that the student needs to know how to instruct the computer algebra system properly to solve his problem, which is a complex task in itself. And furthermore, the student in the first place has to find the correct procedures and how they are named, in order to program the computer algebra system to do what he wants. In other words, he needs strategic competence to start with,

and he would not be a student in this area if he had this already. In some cases, such as rewriting an integral, this may be less of an issue because just copying the integral into the algebra system will work most of the time, but for linear algebra this may be a bigger problem.
Finally, "calculus is littered with examples where the failure of standard symbolic algebra packages to track such restrictions allowstudents to wander into nonsense" [Ravaglia99]. It is important that a student training system tries its best to avoid such situations.
Ravaglia et al. [Ravaglia99] discuss all these problems in more detail.
Concluding, the plain use of a computer algebra system does not help the student to learn mathematics.

1.3 Proof Support Systems

Proof support systems give electronic support to making derivation steps in a proof. If set up right, the system only allows correct derivations. In contrast with computer algebra systems, choosing the proof steps is mostly hand work. We looked at a number of such systems.
Omega [Omega05] holds middle between a programming language and a proof support system: it is an evaluatable specification language. Such an approach might allow proofs of validity, but it seems that it is stuck too close on the algorithmic (programming) side of the proof to be of use for our goals.
A whole range of formal proof support system have been developed. These systems basically can check whether a proof is correct, by checking that in each step of the proof, the rules of the theory were correctly applied. Rules can be applied in any direction. To mention a few general systems: Isabelle [Isabelle05], Elf, Coq [Coq03], Otter-λ [Otter05], and IMPS [IMPS01]. Logic reasoning is at the base of these proof systems, and many proof systems focus on logic to start with (Figure 6). For an overview check [Ditmarsch05]. Delft University has also developed a logic proof support system named "BOP" [BOP05].

```
Coq < Goal (A -> B ->
C) -> (A -> B) -> A ->
C.
1 subgoal
  A : Prop
  B : Prop
  C : Prop
=========1==========
   (A -> B -> C) -> (A
-> B) -> A -> C
Coq < intro H.
1 subgoal
       ...
  H : A -> B -> C
==========2==========
   (A -> B) -> A -> C
Coq < intros H' HA.
1 subgoal
       ...
  H : A -> B -> C
  H' : A -> B
  HA : A
==========3==========
   C
Coq < apply H.
2 subgoals
       ...
==========4==========
   A
subgoal 2 is:
 B
Coq < exact HA.
1 subgoal
       ...
==========5==========
   B
Coq < apply H'.
1 subgoal
       ...
==========6==========
   A
Coq < assumption.
Proof completed.
```

Figure 6. Coq proof of a rather trivial lemma `(A -> B -> C) -> (A -> B) -> A -> C.` The rule applications are given by the boldface command after the `"Coq <"` prompt; the rest is feedback from Coq. Sequence has been shortened a bit, replacing repeating subgoals with "...". Formatting is in 3 narrow columns to save space here.

One problem is that the application of a rule becomes quite involved. A rule can be applied in many ways, and learning the formalities alone is a problem in itself. This

comes on top of learning to interface with the proof system in the first place. Usually for algebra and calculus proofs, logic has a role in the background only and would only clutter the proof.
Another hurdle to overcome would be to typeset formulas in a student-friendly way (these systems communicate in all plaint text). Proofgeneral [Proofgeneral05] can encapsulate Isabelle and Coq, for typesetting formulas and easy editing (Figure 7). However, what we saw so far is not powerful enough, layouts were still pretty basic and not ready for even basic mathematics. Alfa [Alfa04] is doing something similar.

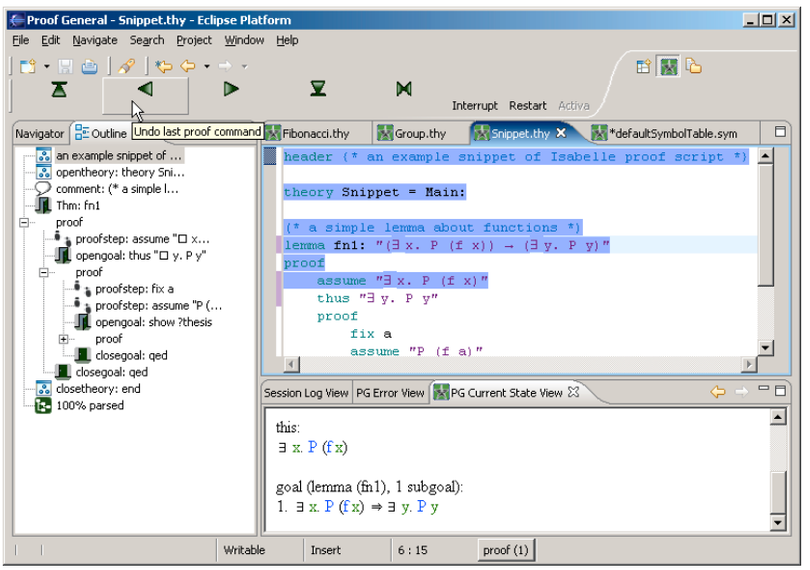


Figure 7. Snapshot of Proofgeneral running Isabelle. Note the symbols in the right top window for "Exists" and "Implies". From [Proofgeneral05].

Within the EPGY remote learning project at Stanford university [EPGY05], another proof support system has been developed (Figure 8). The rewrite-rule buttons use colored boxes instead of x and y in rule templates to avoid confusion with x and y in the equation being worked on. A rule becomes available only after student learned the theory behind the rule in the accompanying course.

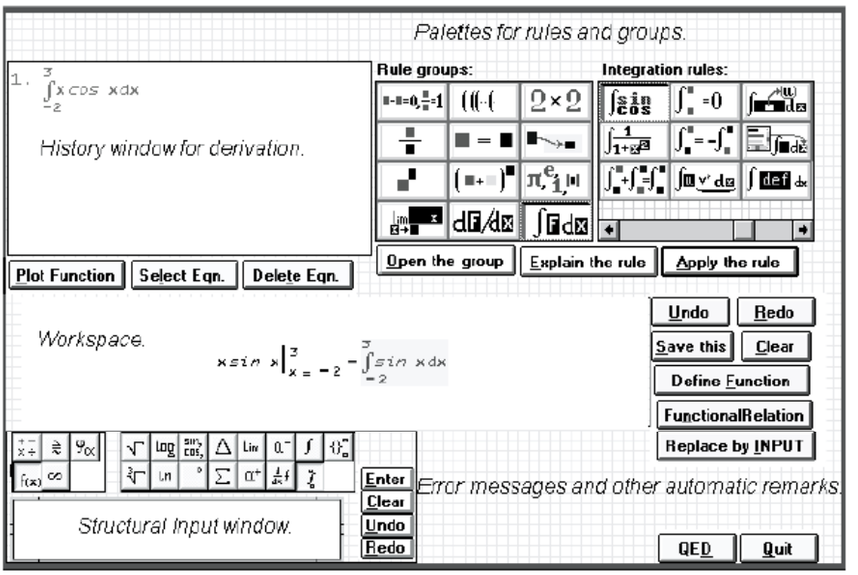


Figure 8. Interface of what appears to be the EPGY proof support system [Ravaglia99].

The main problem with proof support systems is the potentially amazing complexity underlying supposedly trivial proofs. Dan Synek [Synek04] has worked on supporting linear algebra with coq, but he reported for instance for group theory that what seemed a simple proof became a big issue taking over 6000 proof steps. We are not sure if a proof support system would be feasible to create a basic linear algebra tutor. Generally, it is essential to create a set of rules that are appropriate to the level of reasoning that is required. Too low level and the proof becomes unreadable and near impossible to create by hand, too high level and the student only learns to press the "solve" button. But probably an appropriate set can be created, for a restricted topic range like basic linear algebra.

Another problem is that, in general, it can be very hard to determine whether the result is still valid somewhere. If the domain of validity has become empty, results are of no use and may be only misleading [Ravaglia99]. "Unless one keeps track of the restrictions as they arise in the derivation system, the system will not meet the needs of students. If the result of a derivation is literally false, than students who do not catch the error will be misled and those who do catch it will be confused. Neither case is desirable.". Ravaglia et al. internally use a complete dependency graph of logic. One can look at the tree with a menu option. They automatically use previous assumptions to keep things easier for student. It is not clear how important this issue is for linear algebra. But for most linear algebra exercises, this seems not a big issue.

Some of these proof support systems also offer possibilities to indicate a preferred application direction to a rule. If used smart, this might give a smooth path of hints towards a solution. We do not know to what extent this would work, but we expect that it might work for the basic linear algebra topics that we are focusing on.

Interesting sidenote is that Ravaglia quotes Nicaud [Nicaud94] with "Many researchers who worked in this field have changed their goals and are now working on educative but non-teaching environments... Many researchers have more or less reached the conclusion that intelligent tutoring systems are *impossible*.". Ravaglia emphasizes that they did not reach that conclusion. Nevertheless one wonders how true the quote is, as it seems to describe exactly what happened with Ravaglia and the EPGY project...

Concluding, to create an electronic tutor based on a proof support system seems feasible but there are quite some issues that need working on. We need (1) a layout frontend that can standard mathematical typesetting of the equations (2) a frontend that can select the appropriate rule set (3) a hints and tips frontend that can use the current equation set, applicable rules and other (pre-generated?) data to generate tips and support. (4) an appropriate rule set that allows rule availability at the appropriate level and which takes care of the assumptions at hand (5) a mechanism to keep track of the assumptions that were made.

1.4 Proof Support with Hints

The last category is closest to a full tutor system, depending on the level of the hints and tips. If the hints are only superficial, the student may still not learn much from it; but if the hints are well thought out, it at least looks very promising.

Teachers might be concerned about misuse of such a hint button. Beeson however reports "In practice, students seldom misuse auto mode. They seem to take pleasure in finding and applying the correct operations. The fact that the computer carries out the mechanics of the application, so that you can't loose a minus sign or forget to copy a term, or incorrectly apply an operation, increases the user's control over the developing solution." [Beeson98]. We have the same feeling with the software he refers to (MathXPert), that the program does not induce an urge to hit the hint (auto) button, except for a few rare cases when it is unclear which part of the equation has to be selected in order to be able to apply the operation one has in mind. However, students that have missed part of the theory might be more tempted to use the hint button. The software might keep track of the use of that button and return a similar problem sometime later, to see whether the missed theory was picked up.

MathXPert [MathXPert05] is an algebra tutor that covers three years of mathematics topics from linear functions, polynomials, factoring, radicals, inequalities, logs and exponentials, trigonometry, matrices, limits, differentiation, integration to differential equations.(Figure 9a,b). In fact it is more a trainer than a tutor, as it contains no explanation of the material but only exercises and applicable rules. The user can select from a set of exercises for every topic. He selects part of the formula, and then uses the menu to select a rule that can be applied to the selection. He can also ask for suggestions, or to do the next step leading to a result. When the user thinks the final result is reached, he can ask MathXPert whether that's the final result.

MathXPert does not enforce a single road to a solution, and in fact users may take long detours. We think that this is a plus point, although it might be useful if MathXPert would give some suggestions if strange detours are being made. More problematic is that sometimes, application of a certain rule may lead to nowhere, and MathXPert may not be able to give any suggestions on proceeding further although it could do perfect with the starting formula.

A controlled experiment has been conducted with MathXPert, in a high school in Estonia [Beeson99]. Four sections of calculus were involved. Two of these sections had access to MathXpert after school in a computer lab. The other two sections were denied access. All four sections received the same examinations, lectures, and homework. The lowest final exam score in the MathXpert sections was 79%. The lowest score in the other sections was 12%, and there were many low scores (which is normal in this course). The

MathXpert section scores were abnormally high. There were some, probably minor, issues with the experimental setup that might have influenced the results.
Overall, we think MathXPert has a well thought out useful interface that seems appropriate also for linear algebra tutoring. There are a few bugs but the advantages are outweighing them. It is unfortunate that MathXPert has no linear algebra tutoring modules except for Gauss-Jordan elimination and the use of inverted matrices. The author, prof. Beeson, has no plans to add those. MathXPert apparently was written in C, there is no way to hook up our own modules to extend the functionality, and there is no source code available.

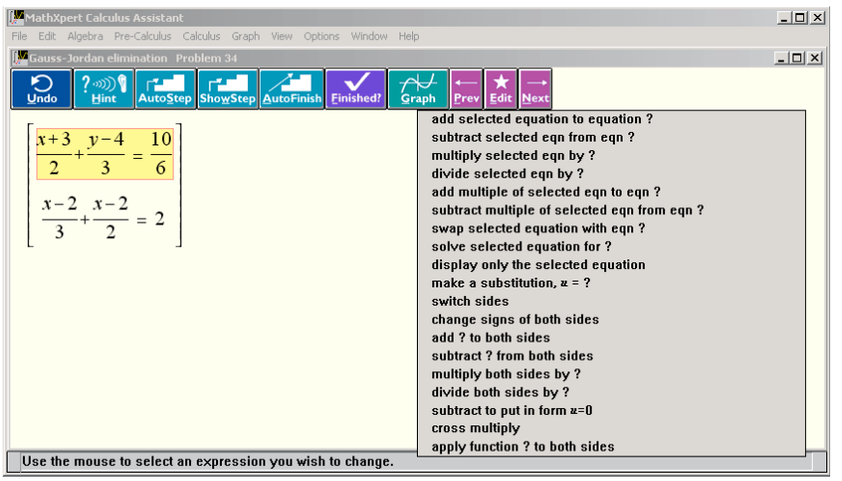


Figure 9a. MathXPert interface, showing the applicable rules with one of a set of equations being selected.

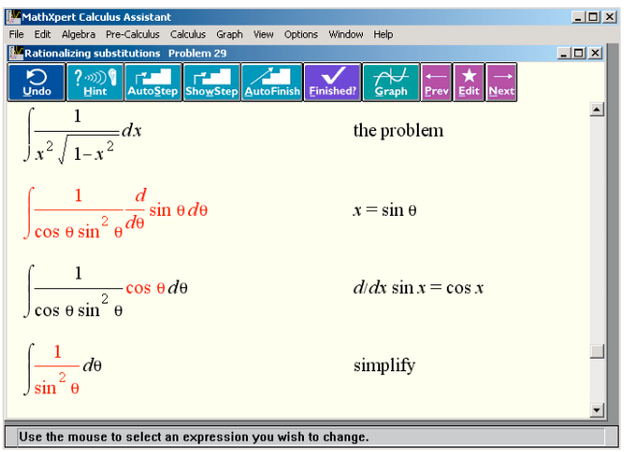


Figure 9b. MathXPert, showing a few steps of a more complex derivation, with the rules that were applied on the right.

Weierstrass [Beeson00] is a further development to MathXPert, adding quantifiers. It is not clear what the target of these developments are, but reported results point in the direction of academic applications rather than student support.
A problem not adressed with MathXPert is that this apply-rule-by-mouseclick approach may result in students knowing which buttons to click to reach a result, but having insufficient mastery of the procedures behind the buttons, or just click all the buttons

until the result is accepted. Students might have to show that they understand and can apply the rule by themselves, besides knowing that they have to press a certain button to apply a rule. Ravaglia et al. [Ravaglia99] ensures this by, after a new rule was introduced in the course, occasionally having students (partially) evaluate a procedure for a rule themselves a few times. This process also makes sure that basic procedures are repeated often enough to keep them ready for use. Avoiding dumb clicking may not be a problem if many alternative steps are available. If a typical derivation is more than two or three steps, and at least three or four clickable options are available at each stage in the derivation, dumb clicking would be highly inefficient and time consuming.

1.5 Conclusion on Electronic Tutors

We have reviewed a lot of electronic mathematics tutor systems. MathXPert looks very close to our goals, with no aspirations to be a tutor but a promising training and support system with a good hint system, robust and intuitive interface, rule application depending on the student level, and good mathematical layout. But unfortunately it does not have a linear algebra module and showed not extendable either. Several proof support systems look powerful enough to technically do what we want, but the graphical front end is too primitive to typeset mathematical formulas. Thus, the linear algebra training system that we want is not yet available, and neither is a simply modifyable system. We will have to make one ourselves.

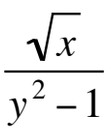
2. Components for an Electronic LA Trainer

We found that there is no 'off the shelf' system that does, or can easily be modified, to get the linear algebra trainer that we aim at. This section looks at possibilities to combine existing components that could form the basis for our LA Trainer.
The first section discusses an approach combining an existing proof support system with a mathematical typesetting front-end. We will find that interactivity is a problem with these typesetters. Therefore, the second section proposes to modify one of the available interactive equation editors.

2.1 Mathematical Typesetter as Frontend

The first approach would be to create a new, powerful frontend for one of the available proof support systems. As we saw in the previous section, there are sufficiently powerful proof support systems, but a good mathematical front end is necessary and not readily available. This section discusses the available mathematical typesetters and their suitability to be transformed into a frontend for a tutor system.
Traditionally, mathematical equations have been represented in processes using a variety of formats such as TeX, MathType (the format generated by the MS Word Equation editor), and PowerMath. Recently, MathML recently entered in this arena. It seems that only TeX and MathML are open standards for which open source software has been written.
MathML is a attempt to standardize the description of mathematical formulas (Figure 10). There are many products that can convert a MathML object into a picture (mostly scalable vector graphics), for instance [Soft4Science05] and [SVGMath05]. (see [w3orgMathml05] for more). SVG has interactivity capabilities, but interaction code has

to be embedded into the SVG code. Interaction code seems not included with those converters. Standard SVG code allows detection of roll-over mouse events, which could be combined into selection events. Howerver, modification of for instance SVGMath, and enforcing proper hierarchical selection would take quite some programming effort. There are also MathML to TeX converters.



```
<math>
   <mfrac>
      <msqrt> <mi>x</mi> </msqrt>
      <mrow>
         <msup> <mi>y</mi> <mn>2</mn> </msup>
         <mo>-</mo>
         <mn>1</mn>
      </mrow>
   </mfrac>
</math>
```

Figure 10. MathML Code (right) and the corresponding mathematical layout (left).

Tex usually is converted into bitmap format for display. But by using Latex MetaPost [Hobby92], it is possible to generate bounding boxes for objects, and use those bounding boxes to detect which object(s) are being selected when a user clicks or drags on a generated bitmap. Programs as MetaGraf [Muelas02] Illustrate the idea. This would probably a bit less effort as with as with the SVGMath approach, but still a substantial effort.
Concluding, there are some layout tools available to create nicely formatted equations. However there is a problem when it comes to interactive selection of parts of an equation possible. Quite some work will have to be done to realize this.

2.2 Modifying an Interactive Equation Editor

Selecting part of an equation is essential to indicate what a rule has to apply to. Selecting this with a menu or keyboard would be a cumbersome taks. If we want to develop a serious electronic mathematics tutor ourselves, we will need interactively selectable mathematical equations (e.g., by mouse clicking and dragging).
As we saw in the previous section, modifying available mathematical layout modules to enable interactive selection of parts of the equation will be quite some work. Instead, we might try to modify one of the available equation editors to our needs. This section discusses the possibilities. Because we are going to work a lot with matrices, we will particularly check how versatile selection of parts of matrices is.

2.2.1 Modifying Plain Editors

A lot of plain equation editors are available. We discuss their suitability for modification as a tutor frontend.
There are quite a lot of equation editors that are completely pre-compiled. For example, EqMagic Lite [Micropress00], Integre TechExplorer [Integre05] and TerraDotta's MathIWYG [TerraDotta05] (Figure 11). We can not use these, because we need to know when the user selects something and what he selected. We could not check out TechExplorer in more detail as the installation required administrator permissions that we did not have at the moment of evaluation.

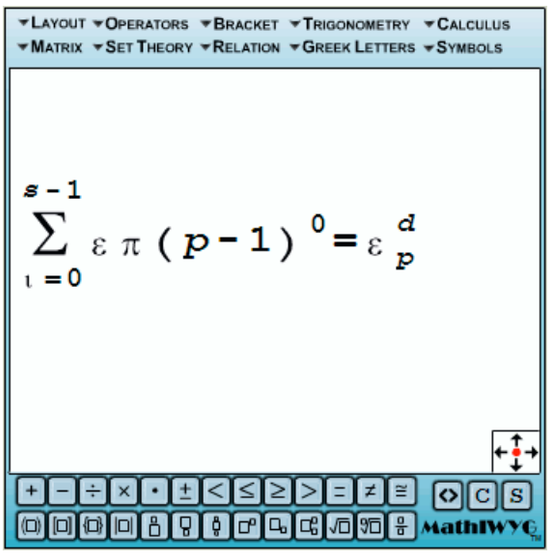


Figure 11. Snapshot of TerraDotta's MathIWYG editor.

Formulator [Hermitech05] is an equation editor that enhances the MS Internet Explorer browser to display and edit mathematical formulas. It runs only on MS Windows and uses ActiveX components. It allows to create new layout styles both in terms of graphics and edit boxes. Formulas can be zoomed in as needed. Template buttons are available for formula creation. Selection of rows or columns of matrices is not possible. But in spite of this apparent open-ness, the package is precompiled and we can not catch individual select actions or change menu options.

Some approaches even completely avoid creation of an editor, by using an already existing pre-compiled editor. For instance, the MathFlow Editor [Mathflow05] can use the equation editor in MS Word or a few commercial XML editors, and converts the resulting MathType objects for further processing.

Then, there are a number of editors that allow the user to modify the plain text MathML code, after which a kind of preview window shows the mathematically typeset version of the code. Examples are EzMath [EZMath05]. It is clear that we do not want a student to be messing with MathML code, so this is not an option either.

sMArTH [Smarth05] is an interesting option based on several other open source modules. As we saw in the previous section, there are open source MathML to SVG convertors available. This project uses the Adobe SVG viewer to show the resulting SVG. It uses a public domain font set, and Internet Explorer to show the SVG. But there are some issues. Netscape does not show the formulas correctly: square root symbols are visible but no text appears. There are a few bugs, for instance pluses and minuses always appear in duplicate ("++" instead of "+"), and it complains about a missing server such that save is not possible. Unfortunately, it is not possible to select rows and columns in matrices.

All editor code, including MathML and TeX output conversion is available. Everything is in JavaScript. So we could probably easily modify this, which would give us a full front-end for a tutor system. The system is based on a browser, easily enabling distributed set-ups with for instance a Mathematica kernel running remotely.

A serious problem is that the sMArTH editor is extremely slow. Four minutes after start-up, still not all buttons have appeared. Buttons also appear to behave buggy. It seems that the Javascript in the SVG viewer is just too much to be useful at this point in time.
Publicon [Publicon05] is a Mathematica-style text editor from the creators of Mathematica [Wolfram05]. We did not check this out further, as it would be most natural to use Mathematica straight away if we were going this way.
Math is the equation editor in the Open Office suite [Math05]. With this editor it is not possible to select a subset of the rows or columns. Selection goes pretty awkward, apparently you are editing in the plain latex formulas instead on formula level. The suite is all open source so it is possible to modify as necessary. However, the sources are huge (214 Mb) and it is not clear how the modules are interwoven. Therefore this may not be the easiest path to a tutor front-end.
Jex [Jex05] is another open source equation editor for Open Office. It is written in Java and seems more separated from Open Office than Math. Apparently Jex is using LaTeX under water for typesetting. This seems the most promising possibility in this category, but unfortunately the installation was rather complex and failing somewhere halfway so we could not test this further.

2.2.2 Modifying Editors with Underlying Math System

One interesting possibility is to follow the approach of DirectMath [DirectMath03]. DirectMath (formerly named Leibniz) is a mathematical word processor and easy to use front end to the Mathematica computer algebra system (Figure 12). Template buttons are available to create formulas. Formulas, part of the rows or part of the colums can be selected, after which a context sensitive menu allows application of Mathematica rules to the selection. The result is then inserted on the next rule in the document. Arbitrary square subsections of matrices can be selected.
DirectMath works as follows. After the user has selected something, DirectMath makes a call to the Mathematica code, passing the current formula and the selection that the user made. The Mathematica code is then expected to return a list of applicable rules, including the text to be placed in the context sensitive menu. When the user selects a rule, Mathematica is called again to apply the requested rule. It is possible to extend the Mathematica code, such that new rules become available fitting our teaching purposes.
One small drawback is that, when the user selects single columns, DirectMath indicates to Mathematica that the entire matrix has been selected. This makes it impossible to make the context sensitive menu sensitive specifically for column-operations. Consequently, column operations will have to be provided allways when a matrix is selected, and the user will have to be requested which columns he wants to operate on when he applies one of the column operators.
A larger problem is that the user can type new formulas at any time. For a tutor system this might pose problems. For instance, the user could directly type in the answer. The tutor then has no way to determine how the user found this answer, and consequently can not give proper advice. Also, this makes it hard to track what the user is doing: is he still working on the last problem he requested, or did he start working on something else?

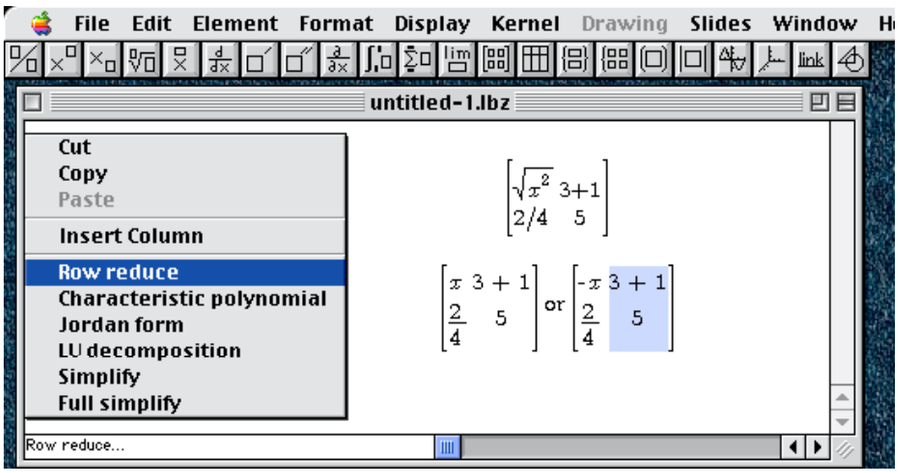


Figure 12. Snapshot from DirectMath. The square root was just simplified by DirectMath. The menu shows applicable rules for the current selection.

TeXmacs [TexMacs03] is a text editor (Figure 13), in which mathematical equations can be inserted interactively in WYSIWYG style. TeXmacs is open source, so it may well be possible to modify the editor such that TeXmacs would serve as a front-end for our tutor. And even more interesting, TeXmacs has been used already as a front-end for several computer algebra systems. One of those, Maxima [Maxima04], is open source (Figure 14).

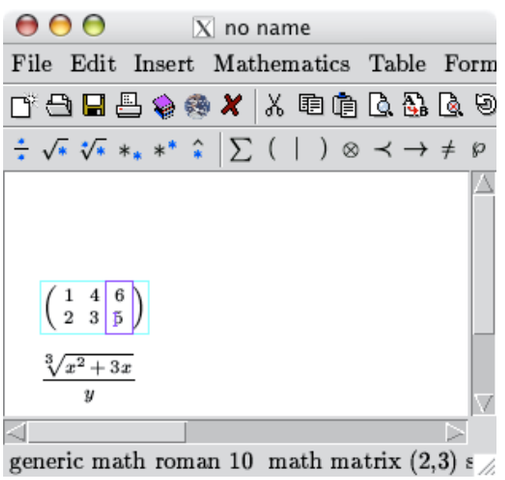


Figure 13. TexMacs document with matrix and formula. A column of the matrix has just been selected.

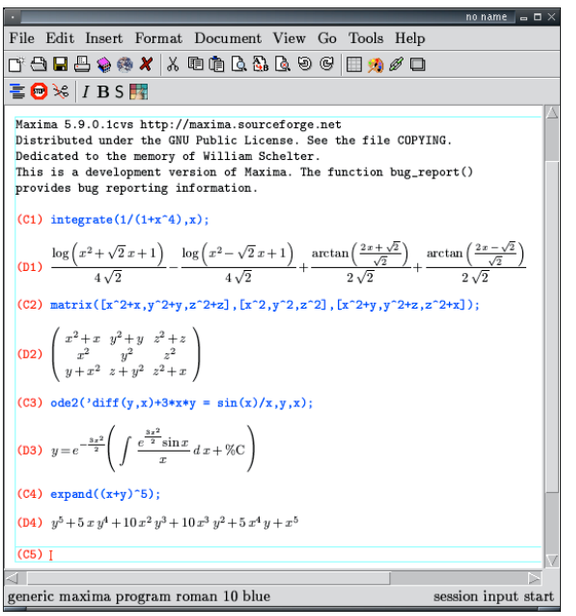


Figure 14. Maxima running with TeXmacs as front-end.

It is not entirely clear if a powerful system like Mathematica kernel under water is going to give benefits versus building from scratch. This issue was already a discussion point

between Ravaglia [Ravaglia99] and Beeson [Beeson99]. It is not clear what the main arguments are, but it seems that the problem of which form is acceptable as the final answer is a central point. To mention two extremes: a simple plain text comparison between a correct answer and the answer that was reached is insufficient as mathematical answers do not have a unique form; on the other hand, having Mathematica compare the final results would be inappropriate too, because even the starting equation might often be found equal to the correct answer, leaving no work for the student. For the moment, we think that having some kernel available will readily provide us with useful mechanisms that would have to be programmed anyway.
We are not aware whether this discussion has been resolved. Transmath, one of the discussed proof support systems with tips, was originally based on Mathematica, but in later implementations the dependency Mathematica was completely removed [Transmath02]. The main reason however was not technical but commercial: most schools did (and do) not have Mathematica, and Mathematica is quite expensive.

2.3 Conclusion

We looked at quite some equation editors, both without and with a mathematical kernel running below it. The fastest approach to our goal seems to take an editor with an already established mathematical kernel link, and to tweak it to our needs. DirectMath looks promising but there are some issues getting the interface work for us in tutor-style. The two main contenders that remain are TexMacs+Maxima and Mathematica.
For development of a prototype, it may be best to go with Mathematica first. We have all knowledge in house to quickly develop something to start with. We know pretty sure what is possible and what not.
But financial considerations might also come in play. TexMacs+Maxima is a free, open source software package under GNU public license. Mathematica on the other hand is expensive: a one-year single user campus license costs €425, a network server license €950 plus costs per active user (€890 if I understand this right).

3 System Configuration

Now we have chosen to start off with Mathematica, we can continue to propose a system configuration (Figure 15). We propose to tweak a Mathematica Frontend both for creating the typesetting frontend of the current result as well as for the rule buttons window. This should be rather straightforward. A kernel process then can glue the parts and do the actual processing of finding applicable rules and applying them. When using TexMacs+Maxima, we expect that this architecture should still work.

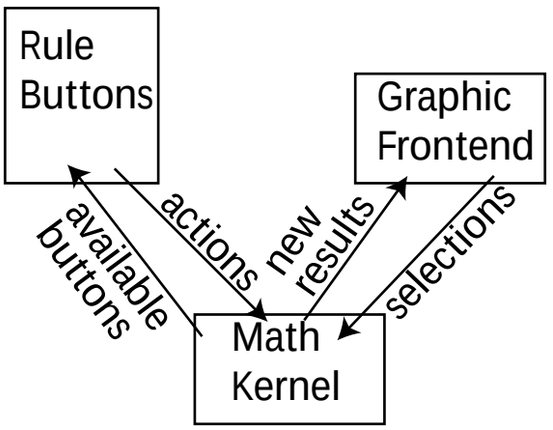


Figure 15. Sketch of the system configuration.

3.1 Tweaking the Frontend

We do not want the user to manipulate the frontends directly, but this is possible using the read-only option for frontends in Mathematica. For the rule button icons, we may considered colored template buttons in the style of the EPGY project.
Kelmanson reported a number of issues when creating a tutor system based on Mathematica [Kelmanson92]:

> In producing the Matrix module, we encountered the occasional difficulty within the Notebooks environment. For example, once opened, groups remain thus even though the material contained therein may no longer be required. Thus the Notebooks becomes more linear in appearance as progress is made through the module unless the user explicitly closes the “finished” cells. Similarly, when the user is interacting with Mathematica via its dialogue box, the autoscroll in Notebooks ' main window is disabled, thus forcing the user explicitly to search for the responses to their input. In version 2.2 of Mathematica , there is no control over either the size or location of the dialogue box; this, coupled with the aforementioned loss of autoscroll, constitutes a non-trivial problem in terms of ease of navigation by uninitiated users. Finally, the absence of mathematical text and the inability to change fonts within a cell proved somewhat restrictive; much effort is required to circumvent this rather artificial restraint. Mathematica 's absence of hotwords or “intuition-driven” buttons etc. enforce a certain degree of linearity on the style of the module. We hope that future versions of Mathematica will accommodate the necessary features to support a more tree-structured module design: if they do not, then Toolbook may yet be employed –upon arrival of MathLink– to front-end the interaction, which is a further reason supporting the shielding of students from Mathematica 's programming language.

In contrast with Kelmanson, we don't plan to create an endless string of exercises. Instead, a single notebook should focus on only a single exercise. For a new exercise a new window is supposed to open. This way, there should be less navigation issues. We certainly do not expect that the search facility of Mathematica is needed to find responses

to input. For some actions a dialog box may be necessary though, and it seems that fixing the location of the dialog box (just another Notebook) can not be requested or suggested. We hope that this is not giving user interface problems. We also do not expect to need multiple fonts within a single cell, because we do not aim at tutorials within the notebook, only mathematical manipulations and maybe hints. Overall, in contrast with Kelmanson we really try to keep Mathematica kernel out of view of the user and we hope that this will avoid the issues they found.

3.2 Some Tutor Issues

We will have to craete a mechanism allowing appropriate introduction (and removal?) of rules as the course proceeds and the student's knowledge increases.
From [Beeson98] we learn "The original plan was to customize mathxpert to the individual user's level. However user models tend to become unbalanced, and this resulted in awkward solutions, which contained one or two big steps mixed with detailed steps. It showed that 'ideal' student model is better.". Therefore, we suggest that rules can be made available on individual basis, and that rule sets are developed fitting the required 'ideal student' levels, for instance on the basis of the chapters of a standard text book.
[Ravaglia99] makes an important remark if we want to have a Finished/Judge button: "If full blown equivalence is tested for between author and student answers, problems inwhich the student was supposed to have transformed an expression according to algebraic rules into a new expression would be useless. This is because the student could just type in the initial expression, which in the case of problems from an algebra course, is almost always algebraicly equivalent to the answer.". For many cases we think that we can make an acceptable evaluator, based on checking the primitives that are still in the answer. For example, it could be indicated for each exercise if a number only is acceptable as an answer, or a matrix, or a set of vectors etc. It probably is possible to recognise if an answer is far from a normalized format. More complex cases are imaginable though, and we may have to develop solutions for this.

References

[ActiveMath05]. ActiveMath Home. German Research Center for Artificial Intelligence (DFKI GmbH), Saarbr?cken, Germany. Available Internet: http://www.activemath.org.

[Alfa04] Hallgren, T. (2004). Alfa. Available Internet: http://www.cs.chalmers.se/~hallgren/Alfa.

[BOP05] BOP (2005). BOP Help Pagina's. Department of Electronic Engineering, Mathematics and Computer Science, Delft University of Technology. Available Internet: http://logica.its.tudelft.nl.

[Beeson00] Beeson, M. (2000). A theorem prover called Weierstrass. Available Internet: http://www.mathcs.sjsu.edu/faculty/beeson/Papers/weier.html.

[Beeson98] Beeson, M. (1998). Design Principles of Mathpert: Software to support education in algebra and calculus, in: Kajler, N. (ed.) Computer-Human Interaction in Symbolic Computation, 89-115. Springer-Verlag, Berlin. Available Internet: http://www.mathcs.sjsu.edu/faculty/beeson/Papers/pubs.html.

[Beeson99] Beeson, M. (1999). MathXpert: Learning Mathematics in the 21st Century. Available Internet: http://www.mathcs.sjsu.edu/faculty/beeson/Papers/English-ste/English-ste.html.

[Coq03] The Coq Proof Assistant. Available Internet: http://coq.inria.fr.

[Day01] Day, J. M., & Kalman, D. (2001). Learning Linear Algebra: Issues and Resources. The College Mathematics Journal, 32 (3, may), 162-168. Available internet: http://www.american.edu/academic.depts/cas/mathstat/People/kalman/pdffiles/resources.pdf.

[DirectMath03]. Gregg, J. (2003). DirectMath. Available Internet: www.directmath.com.

[Ditmarsch05] Ditmarsch, H. van (2005). Logic Software and Logic Education. Computer Science Department, University of Otago, New Zealand. Available Internet: http://www.cs.otago.ac.nz/staffpriv/hans/logiccourseware.html.

[EPGY05] Education Program for Gifted Youth. Stanford University, CA.. Available Internet: http://www-epgy.stanford.edu/research.

[EZMath05] Raggett, D., & Batsalle, D. (1998). Add Math to Web pages with HP EzMath v1.1. Available Internet: http://www.w3.org/People/Raggett/EzMath.

[Hermitech05] .Formulator: Mathematical Equation Editor. Hermitech Laboratory, Zhitomir State Technological University, Zhitomir, Ukraine. http://www.hermitech.ic.zt.ua/projects/formulator/index.html.

[Hobby92] Hobby, J. D. (1992). Introduction to MetaPost, Proceedings of EuroTeX '92. . Available Internet: http://cm.bell-labs.com/who/hobby/MetaPost.html.

[IMPS01] Farmer, W. M., Guttman, J. D., & Thayer, F. J. (2001). IMPS: An Interactive Mathematical Proof System. The MITRE Corporation. Available Internet: http://imps.mcmaster.ca.

[Integre05] Integre MathML Equation Editor. Integre Technical Publishing Company, Albuquerque, NM. Available Internet: http://www.integretechpub.com/zed.

[Isabelle05] Isabelle (2005). University of Cambridge, Technische Universit?t M?nchen. Available Internet: http://www.cl.cam.ac.uk/Research/HVG/Isabelle.

[Jex05] Levine, D. K (2005). Jex - A Java Equation Editor for Open Office. Available Internet: http://levine.sscnet.ucla.edu/general/software/jex/#Installation.

[Kelmanson92] Kelmanson, M. A., Maunder, S. B., & Cheng, S. Y. (1992). A Mathematica-based CAL Matrix-Theory Tutor for Scientists and Engineers. Available Internet: http://caliban.leeds.ac.uk/download.html.

[Kilpatrick00] Kilpatrick, J., Swafford, J., & Findell, B. (2000). Adding + it up: Helping Children Learn Mathematics. Washington, USA: National Academy Press. Available internet: http://www.nap.edu/books/0309069955/html.

[Lyryx02] Lyrix Interactive Linear Algebra. Lyryx Learning Inc., North York, Ontario, Canada. Available Internet: http://lila.lyryx.com.

[MapleTA05] Maple TA: Automated Online Testing & Assessment. Maplesoft , Ontario, Canada. Available Internet: http://www.maplesoft.com/products/mapleta/index.aspx.

[Maplesoft05] Maplesoft: Command the Brilliance. Maplesoft , Ontario, Canada. Available Internet: www.maplesoft.com

[Math05] OpenOffice (2005) Math: Create Equations and Formulae for your Documents. Available Internet: http://www.openoffice.org/product2/math.html.

[MathXPert05] MathXPert: Software to Help You Learn Mathematics (2005). Available Internet: http://www.helpwithmath.com/. Software also available through http://www.mathware.com/mathxpert.html.

[Mathflow05] Mathflow (2005). MathFlow: MathML workflow tools for STM publishing. Available Internet: http://www.dessci.com/en/products/mathflow.

[Maxima04] Maxima: A Sophisticated Computer Algebra System. Available Internet: http://maxima.sourceforge.net.

[Micropress00] Equation Magic Lite (2000). Micropress Inc, Forest Hills, NY. Available Internet: http://www.micropress-inc.com/eqmlite.htm.

[Micropress00] Equation Magic Lite (2000). Micropress Inc, Forest Hills, NY. Available Internet: http://www.micropress-inc.com/eqmlite.htm.

[Muelas02] Muelas, S. (2002). Welcome to MG-log's page. http://w3.mecanica.upm.es/metapost/metagraf_log.php.

[Nicaud94] Nicaud, J. F. (1994): BuildingITSs to be used: lessons learned from theAplusix project. In: Mendelson, P., Lewis, R (eds.): Proceedings of Lessons from Learning, IFIP Workshop, Archamps, France (1993). IFIP Transactions A, 46: 181-198.

[Omega05] Sheards, T. (2005). Omega download page. Available Internet: http://www.cs.pdx.edu/~sheard/Omega.

[Otter05] Online Otter-(. Available Internet: http://mh215a.cs.sjsu.edu.

[Proofgeneral05] Proof General: Organize your proofs! Available Internet: http://proofgeneral.inf.ed.ac.uk.

[Publicon05] Wolfram Publicon: Technical Publishing made Easy. Wolfram Research, Inc., Oxfordshire, UK. Available Internet: http://www.wolfram.com/products/publicon/index.html.

[Ravaglia99] Ravaglia, R., Alper, T., Rozenfeld, M., & Suppes,P. (1999). Successful pedagogical applications of symbolic computation. In N. Kajler (Ed.), Computer-Human Interaction in Symbolic Computation, 89-115. Springer-Verlag, Berlin. Available Internet: www-epgy.stanford.edu/research/chapter4.pdf.

[SVGMath05] Grigoriev, N. (2005). SVGMath: MathML to SVG Converter in Python. http://www.grigoriev.ru/svgmath.

[Smarth05] About sMArTH. Available Internet: http://smarth.sourceforge.net/index.html.

[Soft4Science05] Soft4Science (2005). MathML Renderer. Soft4Science, Munich, Germany. http://www.soft4science.com/products/MathML_Renderer/s4s_MathML_Renderer.html.

[Story03] Story, D. P. (2003). Tutorial on the AcroTeX system of Online Assessment. Theoretical and Applied Mathematics, University of Akron. http://www.math.uakron.edu/~dpstory/tutorial/online/howto.pdf.

[Synek04] Synek, D. (2004). Formalizing Lagrange's Theorem in Coq. Proc. TYPES2004 Conference, Jouy-en-Josas, France. Available Internet: types2004.lri.fr/SLIDES/synek.pdf.

[TerraDotta05] MathIWYG v2.0. Available Internet: http://www.terradotta.com/index.cfm?FuseAction=TerraDotta.Home&Selected=Products&Link_ID=118&Page_ID=27.

[TexMacs03]. Hoeven, J. van der (2003). Welcome to GNU TeXmacs. Available Internet: http://www.texmacs.org.

[Transmath02] Transmath: CBL Mathematics Tutor. Transmath Group (2002), University of Leeds, Leeds, UK. Available Internet: http://caliban.leeds.ac.uk.

[UTCourseware05]. Courseware for Mathematics courses. University of Toronto, Department of Computer and Mathematical Sciences. Available Internet: http://www.math.utsc.utoronto.ca/mathlab/courseware.html.

[Wolfram05] Wolfram Research (2005). Mathematica: The Way the World Calculates. http://www.wolfram.com/products/mathematica/index.html.

[WolframCourseware05] Mathematica Information Center: Courseware. Wolfram Research, Inc., Oxfordshire, UK. Available Internet: http://library.wolfram.com/infocenter/Courseware.

[w3orgMathml05] World Wide Web Consortium (2005). MathML Software Editors. Available Internet: http://www.w3.org/Math/Software/mathml_software_cat_editors.html