COMPUTER ALGEBRA IN THE CLASSROOM: PROMISES, PERILS, AND PEDAGOGIC PERSPECTIVES

Edmund A. Lamagna
Department of Computer Science
University of Rhode Island
Kingston, Rhode Island 02881
eal@cs.uri.edu

William C. Bauldry
Department of Mathematical Sciences
Appalachian State University
Boone, North Carolina 28608
bauldrywc@appstate.edu

J. Douglas Child
Department of Mathematical Sciences
Rollins College
Winter Park, Florida 32789
child@rollins.edu

Wade Ellis
Mathematics Department
West Valley College
Saratoga, California 95070
wade_ellis@wvmccd.cc.ca.us

Summary

Computer algebra systems (CAS) have revolutionized how mathematics is done and taught. Their evolution is traced, emphasizing their application to collegiate mathematics. Successes and benefits of CAS in the classroom, along with potential pitfalls and perils, are considered. Several design and implementation issues are discussed to illustrate the inherent and practical limits of CAS. Finally, a case study of their use in conjunction with the unit on integration in introductory calculus is presented.

1. Introduction

This session is a follow-on to one the panelists gave at ICTCM-15 in Orlando. Last year's panel focused on the history of CAS, and on some of the issues involved in their design and implementation. Our goal was to demystify these systems for math faculty who may use CAS but are generally unfamiliar with their inner workings. While we continue in a similar vein this year, our emphasis is more on the use of CAS in collegiate mathematics.

Using a CAS in teaching presents numerous challenges to a math instructor. Often, students have substantially more experience with computers than faculty, and this can be very intimidating for an instructor using a CAS for the first time. Moreover, CAS have a steep learning curve, and so their use can take away time and energy from the underlying subject matter.

CAS very often produce results that are surprising to both faculty and students. Sometimes the answer is not completely correct mathematically. Knowing the issues involved in the design and implementation of CAS helps explain what is happening when these situations arise. Furthermore, there are both theoretical and practical limits to what these systems can do.

2. Historical Perspectives (Wade Ellis)

The development of computer algebra systems began at MIT in 1959-60 under the direction of Marvin Minsky. These early systems were intended to show that computers could exhibit intelligent behavior by performing rule-based operations that human beings thought to require intelligence. They were forbearers of the current MACSYMA system and were not created with instructional use in mind.

Twenty years later, in 1979, the first version of muMath appeared. This software ran on the relatively inexpensive TRS-80 computers. Though muMath did not have the capabilities of MACSYMA, it was designed to be useful in instructional settings. The muMath system was met with anger, disbelief, or excitement. Many users were angered by the output muMath gave. For instance, ln(2) returned ln(2) which, of course, is a correct answer. Some instructors were expecting the scientific calculator output and, therefore, demanded an incorrect approximation. Others were incensed by the partial factoring response $x(x+2)+1$ to `FACTOR(x^2+2x+1)`. Still others were amazed that a computer could produce correct symbolic results for `EXPAND((x+y)^6)`.

In 1980, Maple was developed at the University of Waterloo expressly for instructional use. By 1985, the software was running on a one megabyte Macintosh. Although Maple's symbolic capabilities were impressive, it had very limited graphics and, on the Macintosh, presented a command line interface in a graphical user interface environment. Eventually, it did have the advantage of including packages for calculus and linear algebra, which allowed students easier access to the Maple features most useful in these courses. Maple was also a very open system that allowed users to see the code used to produce almost all results.

In 1988, Derive (a successor to muMath) was introduced as a user-friendly CAS intended for instructional use. Derive was inexpensive and ran on inexpensive computers. The initial versions had limited graphics but used a graphical user interface in a command line environment. Derive enjoyed considerable success, both at the high school and college levels, because it was easy to use and inexpensive. As with Maple, both the graphing capabilities and the graphical user interface were greatly improved over time.

Also in 1988, Mathematica was introduced. A Macintosh version appeared the following year. This CAS had amazing graphics, long and bewildering error messages, and used a subtly different notation. Although it was a much less open system than Maple or Derive, Mathematica introduced the notion of "notebooks", a combination of text, commands and graphs that could be used to illuminate some concept or skill. Many instructors embrace this capability in developing materials for student use.

Mathematics instructors used Maple, Derive and Mathematica initially to do the things that they were already doing in class. Because these systems were easy to use, instructors utilized them to demonstrate the capabilities of a CAS in doing mathematics. As time passed, many instructors created laboratory assignments that students could complete using these systems. Eventually, standard calculus textbooks began to include exercises and projects requiring the use of a CAS.

Textbooks (and courses) like *Calculus & Mathematica* (Porta and Uhl) and *Calculus* (Devitt) were developed that integrated CAS commands, programs, and notebooks into the text under the assumption that students would have access to a CAS all the time and would use it both to understand and to do mathematics. Although these efforts did not find a wide following, they were very successful in teaching students calculus.

The capabilities of CAS tended to make some topics more important and others less important, some topics more accessible and others no long necessary. As a result, a few educators began to think about restructuring their courses and began to develop materials that moved beyond the traditional topics covered in introductory mathematics courses.

In 1995, the TI-92 was introduced as a useful, "in-your-hand" CAS with graphics and tables. A variety of books were developed that showed how to use this inexpensive and portable technology. Since 1995 other CAS have been introduced including Scientific Workplace, LiveMath, the TI-89, and MuPad. These systems include all of the features that are now part of a CAS including excellent graphing capabilities, extensive symbolic capabilities, notebooks, and extensibility.

As we move farther into the 21st century, mathematics instructors will have to determine what to do when students have software packages that do what they teach more easily and quickly. If this brief history of CAS is any indication, there will be many changes in the future in what and how we teach our students.

3. CAS for Students (Doug Child)

Both pedagogical opportunities and impediments arise when using CAS with typical calculus students. Some issues are entering math objects, the organization of commands, and alternate representations of solutions.

The trade-off for entering expressions is ease of use versus pedagogical value. If the students using a CAS do not understand basic expressions, then you might want to choose a method of entry that helps develop such understanding. The basic method of entry used by CAS is a linear representation, like `x/((x+1)*(x+2))`. Students have to deal with parentheses and "extra operators". However, the difference between a product, `x*(x+1)`, and a composition, `sin(x^2)`, is made clearer than when a 2D editor is used. Unfortunately, linear representations are error prone. A complicated expression like `abs((sin(x)+cos(x))^3)/abs(x^2+5)^3` will cause difficulty for many students. Such an expression is best entered using a 2D editor like Word's Equation Editor or by constructing it in an application like Math T/L. An instructor should consider her particular students, and might be wary of asking students who have trouble understanding basic math expressions to use powerful CAS to attack difficult problems.

A second issue is the organization of commands. Traditionally, CAS have a command line where the user types a command together with all of its arguments. This works well if the arguments are simple and the students use only a few commands in a course. Computer algebra systems have now added menu systems and this helps, but these are still awkward if students use lots of commands. Context sensitive command systems are necessary to cope with large numbers of commands. Some versions of CAS provide such a facility. An extreme case is the Symbolic Math Guide, or SMG, a flash application for TI-89 calculators. SMG helps students produce correct solutions to symbolic calculation problems and knows over 950 commands. SMG provides context sensitive help based on problem type and the expression to be transformed. If the problem is to differentiate $\ln(x^3)$, SMG offers the student choices that include the chain rule and $\ln(A^U) = U*\ln(A)$. Often, students do not think about using the properties of logarithms to simplify differentiation problems, but SMG will provide them with this suggestion. Students searching through 950 items in menus would never even notice the log properties.

A third issue is alternate representations of solutions. The home screen on TI-89 calculators provides a scripting facility that captures how a student has solved a problem. For example, in finding the maximum value of a differentiable function f on a closed interval $[-1,4]$, a student might use the following steps: [Define $f(x) = 3x^4 - 16x^3 + 18x^2$, define $a = -1$, define $b = 4$, $d(f(x),x)$, zeros(ans(1),x), f(ans(1)) // use only those zeros in $[a,b]$, $f(\{a,b\})$ // the largest value of f

is the maximum and the smallest value is the minimum]. These steps provide a general plan for solving such problems. The plan can be saved and used for other similar problems. Such meta-solutions help some students learn to solve classes of problems and certainly save lots of calculation.

Computer algebra systems are powerful programs that serve as valuable tools for artful instructors. But their use can cause weak students stress unless they are chosen and used carefully. The question remains, "What do you want your students to learn?"

4. Limitations of CAS (Ed Lamagna)

Students, and often faculty, can be intimidated by the impressive computational power of CAS. Yet there are limitations and dangers, both theoretical and algorithmic, lurking in these systems. An example of a theoretical limitation is the problem of recognizing zero, or determining whether two expressions are equivalent once we get beyond the rational functions. Another example is expressing the solution to quintic polynomials in terms of radicals. Moreover, while an exact solution to quartic and cubic polynomials can, in principle, be given in terms of radicals, the result is completely useless in most cases. Algorithmic limits of CAS arise from practical constraints on the amount of memory available to perform a computation and the time we are willing to wait for a result. A famous example is factoring integers that are the product of large primes. Another is the infeasibility of computing symbolic inverses, determinants, and eigenvectors of all but the smallest matrices since their sizes grow as $O(n!)$.

Most CAS will report $\int x^k\,dx = x^{k+1}/(k+1)$. This answer is not completely correct mathematically, as any first year calculus student knows. The correct answer is $x^{k+1}/(k+1) + C$, where C is an arbitrary constant of integration. Moreover, the answer is wrong for the special case when $k = -1$, for which the solution is $\ln x$. This example illustrates how implementers of CAS have opted to give what is probably the most useful result, rather than one that is strictly correct.

Fractional powers, multivalued functions, and inverse functions pose similar problems for CAS. For example, the transformation $\sqrt{(ab)} = (\sqrt{a})\,(\sqrt{b})$ that we were taught in school is not valid when a and b are both negative. Most students would like to see $32^{1/5}$ transformed to 2 but, in fact, there are five roots, four of which are complex.

CAS like to return a single value as the result of a computation. Most reply that $\arctan(0) = 0$, taking the principal "branch cut" through this multivalued function, although $\arctan(0) = n\pi$ for any integer n. Complex numbers pose another problem for students. They would like to see the logarithm of a negative quantity transformed to "undefined", as they have been drilled. However, the log of a negative quantity is actually a complex number, and so most CAS will not perform the desired simplification.

The inverse functions sin and arcsin behave in such a way that $\sin(\arcsin(x)) = x$ for all x, but $\arcsin(\sin(x)) = x$ only if $-\pi/2 \le x \le \pi/2$. Similarly, the exponential and logarithmic functions behave in such a way that $e^{\ln x} = x$ for all x, but $\ln e^x = x$ only for real x. Students would like to expand $\ln(ab)$ to $\ln(a) + \ln(b)$, but this transformation is valid only when at least one of a or b is non-negative. In recent years, there has been a trend in CAS to allow users to define the domains of variables to avert such problems. This increases the complexity of the software and places an added burden on the user, particularly students.

For an in-depth discussion of these issues, see the classic article by David Stoutemyer, the co-author of Derive and the TI-92 and TI-89 calculators, entitled "Crimes and Misdemeanors in the Computer Algebra Trade" [*Notices of the American Mathematical Society* **38**, 7 (1991), 778-785].

5. Integration in the Classroom (Bill Bauldry)

One of the primary uses of CAS in calculus is to illustrate antidifferentiation. In last year's panel, we discussed the main methods used by CAS to produce indefinite integrals: heuristics, "look-up tables", and the Risch Algorithm. Here, we'll look at some of the pedagogic pitfalls that must be considered when using CAS to teach integration.

Integration is hard to do
While we present students with a small set of rules to handle the "calculus functions" that arise in introductory classes, elementary catalogs of antiderivatives appearing in the end pages of calculus texts often contain 150 or so basic forms. (See, for example, Thomas' *Calculus*.) Going beyond the basics, Dan Zwillinger's excellent *Handbook of Integration* has over 350 pages of techniques. The Risch Algorithm hasn't been completely implemented in any CAS to date. Antidifferentiation is, in general, very difficult. To watch how Maple solves an integral, set `infolevel[int]:=2`.

It's correct, but ...
Stewart's *Calculus* offers examples that highlight problems with the way CAS integrate: the "plus constant" is missing, polynomials are expanded rather than factored, "simpler forms" are not always chosen, etc. These problems arise from design choices made by the implementers of CAS for efficiency reasons. They affect the classroom and need our consideration as educators. CAS often express results in a form that is more general or use notation than students do not expect. They are surprised by `RootOf` and other unusual answers.

In the "real domain"
Integration is more easily done by CAS in the complex domain, so they naturally do most of their work there. We can sometimes force the issue, as with Maple's

```
use RealDomain in int(sqrt(1+sin(x)^2), x) end
```

but we may lose results and end up with `undefined` for our answer.

The "dreaded erf *function"*
At Appalachian State, we use Maple in our Business Calculus course. Business students need preparation (warning) about special functions; their first response to *erf* is to run away. Even more esoteric functions like Lambert's *W* and hypergeometrics arise from seemingly the simplest integrals. My business students believe that a single log more than suffices; they're not happy when polylogs appear (and think they're probably illegal).

Maple's "Integration Tutor"
To help with these difficulties and to provide a structured environment, Maple has added an "Integration Tutor" to the `student[calculus1]` package. Load the package and try `IntTutor(1/(x^3+x+3),x)`. Students now have a "micro-world" in which to work that will help them avoid the problems we've noted above.

CAS provide an extremely powerful environment for learning antidifferentiation but, like any "power tool", they must be introduced carefully and used thoughtfully.