

FUNCTIONS AND OPERATORS IN MAPLE AND MATLAB

MATTHIAS KAWSKI

(Received September 23, 2003)

Abstract. Some of the most distinctive features of computer algebra systems and numerical packages are their implementations of the concept of a function. When thoughtfully matched and integrated into the curriculum these realizations may be extremely helpful for the students' development of a multi-faceted concept image. On the other hand, thoughtless use of either kind of package may cause more confusion, do more harm than good.

This article contrasts two major features: On one side are the overloaded definitions that make MATLAB so powerful, convenient, and beloved by the advanced user. Typical examples include the sine and reciprocal of a vector, the exponential of a matrix, and the division by a matrix, in particular, when it is not square. At a time when students are supposed to learn distinguish different objects such as scalars and vectors, and different rules for algebraic operations among them, such overloaded functions are potentially hazardous – but when purposefully integrated, these features can much facilitate learning to work with different algebras.

On the other side, both MAPLE and MATHEMATICA distinguish between pure functions and expressions. For the unprepared user this can be a nightmare, but when thoughtfully integrated, these features are wonderful aides to help students develop powerful ways of working of functions. We focus on the key operation of function composition (including taking inverses), and on derivatives f' without a need to name the variable of differentiation. This is very different from the usage in physics.

The main objective of this article is to develop awareness for the consequences of choosing a specific technology, and both its hazards for curricular integrity as well as exciting opportunities to improve the curriculum.

1. Introduction: Functions, technology, and clients' preferences

The concept of a function is arguably the central object of studies in mathematics at both the late secondary and beginning post-secondary level. A vast body of literature has been assembled on this topic, and it is simply impossible to survey it here in detail. Instead, we just mention a few perspectives that are directly related to our concerns. Functions appear at the introductory level in the form of co-varying quantities, i.e. special classes of relations that satisfy an abstraction of the vertical line test, compare [2]. Modern computer technology allows one to replace the traditional static descriptions with dynamic interactive notations, some of which even allow the study of functions at the elementary level [6]. Among these, new forms of graphical representations, going as far as entirely visual languages such as the graphical MATLAB environment SIMULINK, provide many new challenges

1991 *Mathematics Subject Classification* Primary ; Secondary .

Key words and phrases: Function, computer algebra system.

Supported in part by NSF-grant DMS 00-72369.

for the educational community. Clients such as engineering colleges press forward with expectations that such new environments are implemented at a rapid pace. The mathematics education research naturally lags behind in evaluating the merits of such innovations. We refer to [9] for exemplary investigations of the role of visualization.

The concept of a function has many different faces. Students develop understanding of such different aspects only over the course of many years. One extreme is the abstract definition of a function f from a set X to a set Y as a subset $f \subseteq X \times Y$ that satisfies $(x, y_1), (x, y_2) \in f \implies y_1 = y_2$. From this point of view every function is a function of a single variable. While found in most textbooks at various levels, this formal definition is next to irrelevant for the large majority of mathematics courses at the high school and introductory collegiate levels. Instead, the vast majority of class-time and of examination questions is devoted to descriptions in terms of algebraic formulas.

An important alternative or intermediate way of looking at a function is as a mapping $f: X \mapsto Y$. Several modern calculus texts, e.g. [4] employ this language and use the term map (or mapping) as a primitive that needs no definition. In contrast the earlier mentioned formal definition starts with the notion of set with remains undefined at this level.

However, for the learner the notation $f: x \mapsto f(x)$ is fraught with possible misconceptions. This article revisits some of these dangers in relation to the notation employed in computer algebra systems. This notation of mapping is practically synonymous with the practical notions of function, procedure, subroutine, method, etc., as employed in computer science and programming. In contrast to the abstract mathematical notion of a function, here the usual understanding is that the $f(x)$ in $f: x \mapsto f(x)$ stands for some effectively computable expression, i.e. basically a formula. This formula may take the form of several pages of nested subroutines etc., but it is usually understood as computable in a technical sense.

The approach of constructing functions in some basic programming language, whether ISETL or MAPLE or any other, has been explored and studied in depth by Dubinsky and his school [3]. In this article we come to some closely related observations. But our point of departure is entirely different, starting pragmatically: Teaching introductory collegiate level courses in calculus, linear algebra and differential equations at a large public university in the USA, the preferences and wishes of the client disciplines are of importance to the mathematics department. The clients are foremost a diverse collection of engineering programs. While very different in their individual needs and expectations, they all share the accountability via regular strict reviews by the accreditation agency ABET [1]. Consequently, the rate at which innovations are integrated into their curricula is much faster than that in typical mathematics departments. For example, since 1990 this author has taught practically all courses in fully mediated classrooms, in which all students have access to computers, usually shared by pairs of students. This is accompanied by routine teamwork and interdisciplinary curricula, and, to keep it all honest, full Internet access in most exams. For over a decade the preferred software for mathematical tasks at the calculus level were the spreadsheet EXCEL and the computer algebra system (CAS) MAPLE. Calculators were not considered professional tools by our engineers. In differential equations we kept, for many years, a truce with

about equal numbers of sections using CAS and using MATLAB. The first linear algebra course relies predominantly on MATLAB. However, increasingly the engineers are asking that we start right away in the first calculus course with MATLAB instead of CAS.

After having spent the last decade fine-tuning the way CAS may be used to enhance the learning of calculus (which, of course, in turn includes substantial changes to the calculus curriculum), the first reaction is one of disbelief and shock. On one side one loses so many CAS features that have become integral components of our calculus courses. On the other side one has to face a completely different collection of potential problems such as heavily overloaded functions. Thus initially we saw it as our primary task to educate the engineers why their preference simply made no sense: “you can’t have both integration of technology and demand that we use MATLAB in calculus”. But after studying the details, we have begun to wonder whether the real demand is for a radically revised mathematics curriculum. They say MATLAB, but mean new mathematics courses. It is just that the desired courses are most simply characterized as being such that MATLAB can be integrated!

Discussions with colleagues at many institutions suggest that our situation is typical. Moreover, very few engineers and few mathematicians seem to be familiar with the pedagogical implications of adopting MATLAB versus adopting CAS as primary software for calculus level courses. Finally, the available research literature in mathematics education appears to lag far behind the incessant cycles of major innovations of either software.

This article aims at laying out some of the fundamental issues that relate the study of functions (as in calculus level curricula) to the opportunities offered and dangers posed by using either CAS or MATLAB as the primary supporting software at the first and second year college levels. We hope that this article will serve both as a reference for faculty who need to make adoption decisions and communicate with client faculty, and as a starting point for more quantitative education research that investigates the possible benefits and hazards of professional software.¹

The next three sections survey some distinguishing features of MATLAB and CAS that relate to their use in the first college level courses (calculus, differential equations, and linear algebra). They address plotting, overloaded definitions, and encapsulating functions as objects. The final section concludes with some observations about the use of either software in different style calculus courses.

2. Functions as graphs in MATLAB and MAPLE

From an abstract point of view a function is a graph – it is defined as a subset of the Cartesian product of its domain and range that satisfies the vertical line test. Taking a closer look at a function means investigating its graph. In the centuries before easy access to computing technology became standard, much effort was devoted to finding this graph, i.e., tabulating pairs $(x, y) \in f$ that were published as logarithm tables etc. The drudgery of this task, given an $x \in \text{dom}_f$, find a y such that $(x, y) \in f$, motivated analytical approaches that described the function from

¹A preliminary version of some sections of this article has been presented at the ICTME 2 in Beirut, Lebanon, in June 2003.

the formula without having to tabulate large sets of such pairs. On the other hand, due to its repetitive structure, this task is ideally suited for automation. Indeed, the task of manually creating tables has almost vanished from our courses. But instructors at advanced levels deplore that when students encounter new complicated functions, operators, simulations, etc., very few now instinctively first think of (an abstracted) table of pairs. A comparatively benign, but indicative example is presented by the strange machine-generated graphs of $x \mapsto \sin(nx)$ as n increases. Few students understand what is going on – apparently due to their lack of experience of first tabulating values and then transferring them to a plot. In MAPLE, the first interesting patterns on the standard domain $[-10, 10]$ appear at $n = 26$ and $n = 29$. See [8] for a detailed description of these phenomena that are due to aliasing effects.

From the pedagogical side one may consider as the most important learning goal that students develop a good intuition of when to trust a computer generated graph. Desired is an uncanny ability to apply appropriate tests that will either confirm that the graph is trustworthy, or that will refute it and explain why it can't be right. From this side, the sophisticated plotting routines of CAS such as MAPLE, but also the new `fplot` of MATLAB are less desirable: they employ adaptive procedures that increase the number of points as needed – but such decisions are not at all transparent, and failures are often hard to predict or to recognize. On the other hand, the classical way of plotting in MATLAB forces the user to explicitly construct the table of pairs of values to be plotted – albeit all of them at one time with a few keystrokes, not one point at a time. Typical syntax is `xx=-10:0.1:10; yy=sin(26*xx); plot(xx,yy,'ro')`. In this case, the user gets exactly what (s)he asks for. We found that students, who routinely have to explicitly think about what is being plotted, are much more at ease with new problem situations, especially at advanced levels and complicated applications.

There are many other features and specialized plots in either package that provide major learning opportunities or present major possible pitfalls. But this is not the place to address these individually.

3. Learning about mathematical structures: CAS versus MATLAB

At the elementary and secondary level students work with very few, very similar algebraic structures: They start with the commutative semi-groups of the natural numbers with addition and multiplication. Slowly they develop working knowledge of the field of rational numbers, and eventually also a naive understanding of the field of real numbers. The algebras of functions considered at this level naturally inherit the commutative ring structure from their range of real (or rational) numbers. At the post-secondary level students are confronted with a rapidly growing collection of objects that exhibit different kinds of algebraic structures. All of a sudden $a \cdot b$ need not be equal to $b \cdot a$. Similarly, $a \cdot x = a \cdot y$ need not imply $x = y$ even if $a \neq 0$. Consequently, division by such objects is not admissible. Even worse, $a \times (b \times c)$ no longer needs to equal $(a \times b) \times c$. On the same single line of a calculation one may have to use that $c^{-1} = \frac{1}{c}$ while recognizing that f^{-1} is not the same as $\frac{1}{f}$. For many students at this level it is a major challenge to become comfortable and fluent working in these different algebraic structures. Don't divide by a vector. Don't cancel a matrix (unless one has verified that it is invertible).

Don't forget the lack of associativity of the cross-product. Closely related is the increased importance of paying attention to the domains of functions. A silly example is the function $f(x) = \log \log \sin x$ and the statement $\frac{d}{dx}f(x) = \cot x / \log \sin x$. In the context of functions of a real variable the domain of f is empty, whereas the domain of its purported derivative has nonempty interior. More important are the different domains of `atan` and `atan2`, only the second one being a partial inverse of the change to polar coordinates. Another typical example of the importance of domains is that only for finite, square matrices $A \cdot B = I$ implies $B \cdot A = I$.

Now add to this tumultuous situation, in which students for the first time really have to sort out a multitude of different mathematical (algebraic) structures, the request of the engineers for MATLAB as opposed to CAS. We claim that the major CAS MAPLE and MATHEMATICA help as organizing tools, providing much needed stability. Neither one is as strongly typed as classical first programming languages like PASCAL and C, but both force the student to carefully distinguish between constants and variables, between expressions and (pure) functions, between scalar and vector quantities, between commutative associative products and noncommutative and/or nonassociative products. The feedback is immediate and unrelenting. An almost silly example defines a function `y:=x^2-6*x-7`; Differentiate `dy:=diff(y,x)`; and solve to find the critical points `x:=solve(dy=0)`; Any subsequent attempt to use the second derivative test `ddy:=diff(dy,x)`; must fail as MAPLE does not take derivatives with respects to constants. But even earlier, the typical first conflict is the distinction between unordered sets (as returned by `solve(...)`;) and ordered lists: The complex roots of a polynomial are naturally an unordered set. There is no meaningful way to extract a first and a second root. This instructor's experience is that after a short period in which students learn to accept the software's refusal to work with ill-defined commands, students very quickly raise all their work to such higher standards. In such classes, e.g. division by vectors becomes unseen very quickly, simply because the routine use of a CAS forms the habit of constant awareness of the algebraic structures.

Contrast this with MATLAB whose quest to minimize the number of keystrokes needed for any task is legendary. Almost all functions in MATLAB are severely overloaded. This means that there are many different ways in which the functions may be called, i.e. with different numbers of parameters and different types of parameters. Abstractly this means that the domains of most MATLAB functions are often nontrivial unions of sets, with nontrivial rules of which definition applies when. As a simple example, the sine of a vector is defined to be the vector of the sines of its components. In contrast, in a traditional class students learn to evaluate the sine function only on scalars! More caution is advised when calculating `exp(A)` for a matrix `A`. MATLAB interprets this as the matrix whose entries are the exponentials of the entries of `A`, which is different from the exponential of `A` as it is used in the first courses in differential equations and linear algebra.

Our favorite example is the interpretation of the symbol “`\`” for the binary function of left division. In the case of an invertible matrix `A` and a column vector `b` of matching length, `A\b` is beautiful syntax for the solution `x` of the linear system $Ax=b$. After all, at the end of the first course in linear algebra we want all students to think of solving linear systems to be a routine process that generalizes division of numbers. But what if `A` is not an invertible matrix? MATLAB would not want

to waste the precious single keystroke “\”. After all, there could be no confusion for the mature user. Arguably, the best interpretation of $A \setminus b$ in that case is the least squares solution x of $\|b - Ax\|^2 \stackrel{!}{=} \min$. Indeed, in MATLAB the following is perfectly legal, and meaningful: After defining column vectors $v = [1\ 2\ 3\ 4]'$ and $w = [1\ 4\ 9\ 16]'$, the command $v \setminus w$ yields 3.333. This is, of course, the slope of the line through the origin that best fits (in the least squares sense) the data points (i, i^2) , $i = 1, 2, 3, 4$. MATLAB does the thinking for the user – always interpreting to the best of its ability what an assumedly intelligent user must have meant. This is completely opposite to the behavior of all the standard CAS, which play dumb and return error messages that try to convey in which sense the requested operations are illegal.

Teaching first year college level courses with MATLAB can be done – and we have done it – but it is a major challenge. It requires a much higher level of mathematical maturity of the students, a constant high level of alertness by the instructor, and a willingness to constantly bend the rules that the textbooks proscribe. It is a fantastic experience to get a student to bite her/his tongue and smilingly say: “Sure, I can divide by a vector – as long as I understand what I want to do with it.” But it is not easy to get to this level.

In direct contrast, both students and instructor take much more passive rules if supported by a CAS – the CAS becomes responsible for meaningful syntax, and even encourages trial-and-error approaches.

4. Functions as objects

At the beginning of the first calculus course students analyze one specific function at a time. It is a major step forward to work with generic functions. At the end of the first semester all students are expected to reliably differentiate any specific formula, with x 's, e.g., by repeatedly applying chain and product rules. But simplifying $(f \circ g)''$ is a completely different task, which most students at this stage cannot handle. In our experience, CAS can make a major difference in getting significant numbers of students to this higher level where they can work with generic functions f and g as above. On the other hand, it is unclear how a predominantly numerical package such MATLAB can be utilized effectively in such settings.

Both MAPLE and MATHEMATICA distinguish between expressions such as $y := x^2$ and pure functions such as $f := q \rightarrow q^2$. The fundamental difference is that one may “plug in” (substitute) a specific value for a variable in an expression whereas one evaluates a function at a point. In MAPLE: `subs(x=3, y)` versus `f(3)`. A significant number of students who experienced only an algebra-oriented calculus course at their secondary school often have difficulties with pure functions: We have met numerous students who misunderstand the juxtaposition in $\sin x^2$, leading them to use the product rule when differentiating this expression to erroneously obtain $\cos x^2 + \sin 2x$. The same students typically also read $f(3)$ as “ef - three” as opposed to “ef-at-three” or “ef-of-three”. It is here where the arrow notation employed by CAS for pure functions has its biggest impact: It immediately catches such misconceptions about functions, and allows all students to proceed to the next level.

Understanding that the primary operation on functions is function composition, we have found CAS a very effective tool for studying calculus topics addressing functions in general. The first major point is that a pure function need not make any reference to any input variable. E.g., `sin` is a function and its derivative is `cos`. This is very different from the way scientists differentiate one quantity always with respect to another quantity (the b in $\frac{da}{db}$). Without requiring names for input variables, CAS invite one to do analysis in this new algebra of function. For example, in MAPLE `D(D(f@g))`; immediately returns the desired output corresponding to $(f \circ g)'' = (f'' \circ g) \cdot g'^2 + (f' \circ g) \cdot g''$ – the test question so many graduates fail. Other test items on which we observed improved performance after systematic use of pure functions in CAS are of the form: “If f is decreasing and convex, what can you say about its inverse?” (decreasing and concave), and “If f and g are both decreasing, what can you say about their composition $f \circ g$?” (increasing).

The predominantly numeric character of MATLAB makes it much harder to work with compositions of functions – just consider the difficulties of generating a table for the composition of two functions each given by a table. This immediately forces one to reconsider teaching interpolation, which arguably is much underrated in current curricula. A special case in the context of function composition is finding (a table for) the inverse of such a function. The easy answer is to simply switch the columns. However, this is a much harder problem if the sample points in the new domain are prescribed and are different from the sample output values supplied for the original function. Clearly, MATLAB takes one here into entirely different direction when addressing compositions of functions.

We note on the side that the symbolic toolbox in MATLAB does provide access to MAPLE – but at the high cost of cumbersome notation that distinguishes numeric and symbolic objects. In particular, the conversion of large symbolic expressions (that result from symbolic work) into executable function calls is a nontrivial obstacle for any beginner. On the other hand, MATLAB’s function handles, and encapsulation of user defined functions into traditionally external “m-files” provide some intriguing material for further study regarding its impact on students beginning to manipulate functions as objects or as black-boxes.

5. Conclusion and outlook: CAS versus MATLAB in calculus

We claim: Traditional first year calculus courses are courses in algebra, at least as determined by the content of the best-selling textbooks published in the USA and distributed worldwide. For clarification we recall the standard distinction that mathematicians make between derivations and derivatives. Derivations are a purely algebraic object defined as linear operators (on an algebra) that satisfy the Leibniz rule $D(fg) = (Df)g + f(Dg)$. On the other hand, derivatives are analytic objects defined as limits via approximability by linear objects. Limits themselves belong to the topological realm, but in the context of derivatives they are usually defined in the setting of metric spaces.

Every calculus textbook that we are aware of proclaims an analysis point of view. Yet immediately after the first definitions of limits, practically all exercises on limits are entirely in the domain of algebra. This includes all those limits that can be evaluated after factoring and cancelling common factors. Even further, every limit that can be evaluated using Taylor expansions of real analytic functions

(and anything using L'Hopital's Rule) is basically algebra. Differentiation and integration in the somewhat generalized algebras of elementary functions clearly amount to almost purely algebraic operations [5].

Most every mathematics instructor will include a fair dose of analysis in her/his lectures – but the students know what really matters is what is on the final exam. With this the ability of any CAS (operated by a comparatively unskilled data-entry person) to earn a grade of A on practically every traditional calculus exam, constitutes proof that the traditional calculus course is basically an algebra course. After all, algebra is basically all that CAS can do – the A stands for algebra!

Now add computer software to this course: MATLAB can add some pictures, and can perform some numerical checks of algebraic work – but by itself it is almost useless on most traditional exams. As such it remains an add-on that will not even interfere with exams. There is not even any need to discuss rules for what the software may be used for in the exam – and what has to be done by hand. On the other hand, a CAS reduces typical traditional exams to trivial data entry – as such it cannot be allowed on a traditional final exam without rendering them meaningless. Students understand this well. They generally will not invest much effort into mastering a tool that may not be used in exams, when it counts.

But our client engineers not only demand that our students use professional computer technology, but that we even integrate these into the courses (as opposed to “add-on”). In view of the above discussion on how either software conflicts with standard goals and examinations, this can only be possible with a radically reformed curriculum such as [4], which no longer places almost exclusive emphasis on algebra. The first step, taken widely in the 1990s allowed for quite sensible integration of CAS into a reformed curriculum: Much of the new emphasis is on modeling, approximation etc. Routine symbolic calculations are done by hand, while messier computations that result from realistic models motivated by real applications are relegated to the CAS.

This leaves us to wonder about the next phase: Our engineers are very clear about their preference that on one hand CAS be replaced by MATLAB, and on the other hand MATLAB be integrated with the course content. We are looking at another major shift of the courses, and a further move away from algebraic manipulations that apply only to very narrow classes of functions, e.g., those that can be integrated in closed form. Any, even only cursory, look at modern engineering courses, and, even more, engineering practice exhibits substantially larger universes of functions than those studied in traditional calculus classes. Many of these functions are implemented as often quite large scale simulations – i.e., basically very large formulas – which often include many black boxes. At first sight, black box components that implement numerical solutions of differential equations may seem out of place in first year calculus. But upon some reflection, one notices that such black boxes are not very different from those utilized for generations in calculus such as roots, logarithms and trigonometric functions. It really does not matter if the building block is implemented as a logarithm table, or as an m-file in MATLAB. It is just a component function inside a bigger problem whose analysis demands calculus.

Acknowledgement: The author gratefully acknowledges many helpful comments and suggestions from the two anonymous referees, which much helped improve the article – even though we did not manage to implement all of them.

References

1. <http://www.abet.org/> ABET, Accreditation Board for Engineering.
2. M. Carlson, *A Cross-Sectional Investigation of the Development of the Function Concept*; Research in Collegiate Mathematics Education III, Conference Board of the Mathematical Sciences, Issues in Mathematics Education Volume 7; American Mathematical Society, (1998) 114–163.
3. E. Dubinsky and G. Harel, *The nature of the process of function*, in E. Dubinsky and G. Harel, eds., *The concept of function*. MAA Notes **25** (1992).
4. D. Hughes-Hallet et. al., *Calculus* (2001). Wiley, Boston.
5. K. Geddes, S. Czapor, and G. Labahn, *The Risch Integration Algorithm*, in: *Algorithms for computer algebra*, (1992) 511–573. Kluwer, Amsterdam, Netherlands.
6. J. Kaput, *Technology and mathematics education*, in D. Grouws, ed., *Handbook of Research on Mathematics Teaching and Learning*, (1992), 515–556. Macmillan, NY.
7. M. Kowski, *CAS or MATLAB in first year collegiate math?* Proc. ICTME 2 (2003) Beirut, Lebanon (to appear).
8. D. Hardin and G. Strang, *A Thousand Points of Light*, Coll. Math. J. **21** (1990) 406–409.
9. D. Tall, *Functions and Calculus*, in: A. J. Bishop et al (Eds.), *International Handbook of Mathematics Education*, (1997) 289–325. Kluwer, Dordrecht.

Matthias Kowski
Department of Mathematics and Statistics
Arizona State University
Tempe, Arizona 85287-1804, U.S.A.
kowski@asu.edu