[16] WINKLER, F., *Polynomial Algorithms in Computer Algebra*, Springer, 1996 (in press).

[17] ZIPPEL, R., *Effective Polynomial Computation*, Kluwer, 1993.

4. References

[1] BUCHBERGER, B., COLLINS, G. E., and LOOS, R. (eds.), (with cooperation of R. Albrecht), *Computer Algebra: Symbolic and Algebraic Computation*, Springer, 1982.

[2] COHEN, A. M., (ed.), *Computer Algebra in Industry*, Wiley, 1993.

[3] COHEN, A. M., VAN GASTEL, L., and LUNEL, S. V., *Computer Algebra in Industry 2*, Wiley, 1995.

[4] DAVENPORT, J. H., SIRET, Y., and TOURNIER, E., *Computer Algebra: Systems and algorithms for algebraic computation*, Academic Press, 1988.

[5] FLAJOLET, P., SALVY, B., and ZIMMERMANN, P., Automatic average-case analysis of algorithms, *Theor. Computer Sci.* 79 (1991), 37-109.

[6] FLEISCHER, J., GRABMEIER, J., HEHL, F. W., and KÜCHLIN, W., (eds.), *Computer Algebra in Science and Engineering*, World Scientific, 1995.

[7] GALLOPOULOS, E., HOUSTIS, E., and RICE, J. R. (eds.), *Future Research Directions in Problem Solving Environments for Computational Science: Report of a Workshop on Research Directions in Integrating Numerical Analysis, Symbolic Computing, 1991.*

[8] GEDDES, K. O., CZAPOR, S. R., and LABAHN, G., *Algorithms for Computer Algebra,* Kluwer, 1992.

[9] JANSSEN, R. (ed.), *Trends in Computer Algebra,* Lecture Notes in Computer Science # 296, Springer, 1988.

[10] KALTOFEN, E., Polynomial factorization 1987–1991, in *LATIN '92 : 1st Latin American Symposium on Theoretical Informatics,* proceedings, I. Simon (ed.), Lecture Notes in Computer Science # 583, Springer, 1992, pp. 294-313.

[11] ODLYZKO, A. M., Applications of symbolic mathematics to mathematics, in *Applications of Computer Algebra,* R. Pavelle, ed., Kluwer, 1985, pp. 95-111.

[12] ODLYZKO, A. M., Tragic loss or good riddance? The impending demise of traditional scholarly journals, *Intern. J. Human-Computer Studies* (formerly *Intern. J. Man-Machine Studies*) 42 (1995), 71-122. Also in the electronic *J. Univ. Comp. Sci.*, pilot issue, 1994 (http://hyperg.iicm.tu-graz.ac.at).

[13] ODLYZKO, A. M., The decline of unfettered research, to be published. (Can be obtained by sending the message "send research.decline.txt from att/math/odlyzko" to netlib@research.att.com.)

[14] STURMFELS, B., *Algorithms in Invariant Theory,* Springer, 1993.

[15] WILF, H. and ZEILBERGER, D., Rational functions certify combinatorial identities, *J. Amer. Math. Soc.* 3 (1990), 147-158.

another calculator does justice neither to the high technology behind modern calculators nor to the software programs.

A positive way to think of mathematical software as a tool for the masses is that it not only helps those masses solve their problems directly, but also helps facilitate communications in the mathematical sciences. The traditional system of formal printed communication through journal articles and books is breaking down (cf. [7]), and is being replaced by a more flexible arrangement, with frequent feedbacks at all levels. Mathematical software was a pioneer in the evolution of this system, with netnews groups, user mailing lists, user workshops, as well as the more traditional journals, books, and conferences. This has brought leading-edge researchers into closer contact with the mass users, a contact that seems clearly beneficial.

In conclusion, I continue to be a great fan of computer algebra and mathematical software in general. They have created an entire environment in which one can work much more productively on a variety of mathematical, scientific, and engineering tasks. However, it does appear that their role is not to derive a ”theory of everything” through a giant calculation, but at most to act as an aid in the derivation of such a theory. Experience with computer algebra serves to emphasize again the lesson that computing can supplement thinking, but can seldom replace it. As the authors of [7] have put it:

> Two goals for PSEs [problem-solving environments, including computer algebra systems] are, first, that they enable more people to solve more problems more rapidly, and second, that they enable many people to do things that they could not otherwise do.

Computer algebra, and other mathematical software systems, have not allowed top researchers to travel in space, but have given them, as well as a much broader population, the advantage that a VW Beetle has over a horse-drawn cart.

Acknowledgements

I thank Joe Buhler, John Cannon, Richard Fateman, Gaston Gonnet, Johannes Grabmeier, Y. Lakshman, Susan Landau, Michael Monagan, Michael Trott, Franz Winkler, and Stephen Wolfram for their comments.

They are also likely to help in the development of add-on packages. Such packages are likely to be the future of research on mathematical software. I expect there will be many fewer stand-alone systems, and that they will be confined to specialized applications where the market is small, and there are special requirements. The resources to provide support for a widely used tool are simply growing too large for a small academic enterprise to provide. Instead, I expect research groups to concentrate even more on building specialized packages that work with the widely accepted commercial systems. When there is enough demand, such add-ons might also be produced by commercial companies, but the bulk of them will be distributed for free by individuals or small research groups, just as they are now.

The change of orientation towards the mass market might appear to consign mathematical software to the role of a glorified calculator. While that is an extreme view, that is probably the right way to think about it. Most applications of computer algebra, even by leading researchers, fall into this category. Software is typically used, even by top mathematical researchers, just to provide faster, more convenient way to get an answer that is obtainable by hand. The overwhelming majority of applications by the hundreds of thousands of users of Mathcad surely fall into this category. There are relatively few sophisticated applications that use the full power of the algorithms that have been developed with great effort. The issue is how to react to this state of affairs. We can complain about allowing Boeotians in our midst. However, it seems far better to welcome them, since their presence will provide the resources and justification for further research on the deeper problems of computer algebra and other mathematical software. Their presence will require a change in emphasis, though. Most of the users will know practically nothing about computer algebra, and will often know little mathematics in general. A major challenge will be to develop systems that are intelligible to them, and also that warn them when the answers might not be meaningful or appropriate.

We can complain about users not understanding basic mathematical concepts. However, while a better general education is desirable, it appears inevitable that as mathematical software systems spread, their users' level of understanding will decrease. There does not seem to be any way out of this problem. Our knowledge base is growing so fast that the only way people have to cope is by accepting more basic things without detailed understanding. The role of computers is to help us manage the increased complexity we face, and mathematical software is one of the most useful tools at our disposal. While it seems appropriate to think of mathematical software as a slightly glorified calculator, calling it *just* solved, we have to work harder for each incremental gain. On the other hand, there is substantial payoff from work in areas (such as user interfaces) that do not require deep technical insights. The greatest rewards go for work directed at the mass market, not at the scientific and technological elite. What this evolution calls for is not despair, but a reassessment and redirection. Traditionally, computer algebra systems were written by experts for experts. Now they are beginning to be dominated by the masses. These are the masses of scientists and engineers, so they do have some technical sophistication. Still, they are masses that have little knowledge or interest in learning about computer algebra. Their influence is visible in the evolution of commercial software packages. The emphasis is on graphical user interfaces, display capabilities, and typesetting of reports. This is not dissimilar to what is happening in other areas, such as operating systems, where novel software written in university settings has marginal influence, and commercial systems written for the business market dominate, even though they are based on old and often inferior concepts.

What I envisage for computer algebra for the next few decades is a continuation of the trend we already see. Among the major computer algebra systems, this trend was taken into account earliest and most fruitfully by Mathematica, which has as a result achieved the largest market share of the major computer algebra systems. In numerical analysis, Matlab was the pioneer in this direction. The mass market is likely to dominate. In particular, there is likely to be a convergence of computer algebra systems with those from number theory, group theory, statistics, and numerical analysis. Most users will not care about the differences in these fields. They will be thinking simply of mathematical software that can solve their problems with a minimum of trouble. We can already see commercial vendors reacting to the pressure from these users, with numerical analysis tools like Matlab adding some symbolic capability, and symbolic packages extending their numeric functionality. Further, all vendors are expanding enormous energy on graphics, to enable users to understand and present their results better. All these systems appear to be converging to the same form. I expect these trends will result in a dominance of the market for mathematical software by very few commercial systems, since the economic concepts of network externalities and increasing returns will apply. These systems will offer access to the full range of mathematical software capabilities, and will be similar to each other.

Protocols such as MathLink and OpenMath provide formats for exchanging mathematical objects between different systems, and therefore allow users to access a variety of systems without learning about each one. They are likely to smooth the path to the evolution towards a few dominant main systems.

into human thought. Those hopes are nowhere close to being realized. Not only that, we cannot even say that computer algebra has produced any great breakthroughs of the conventional type, say by grinding through a giant computation and coming up with a simple mathematical or physical law such as $E = mc^2$. There have been significant applications, but no spectacular ones. Moreover, usually it is impossible to claim that the results achieved with computer algebra could not have been obtained without it.

Not only has there been a lack of breakthroughs, progress on the whole has been disappointing. The gains in computational power and usability of computer algebra systems have come primarily from general progress of computer technology. There have been no big algorithmic advances in computer algebra recently that have affected several disciplines simultaneously. The LLL algorithm was invented over a dozen years ago. Gröbner bases, which underlie much of the recent progress in computational algebraic geometry, are over 30 years old. There have been major advances, for example in integration, or in indefinite summation and proving combinatorial identities [15]. However, these advances appear to be utilized primarily by experts, and are slow to be incorporated into working systems. Furthermore, the huge scope of computer algebra systems means that a single advance is not likely to affect a typical user unless it is either in a very basic part that is used by almost everyone (such as simplification), or is in a specialized area that is important to that user. Thus to improve the overall perception of usability of computer algebra systems, we need many substantial advances spread over many subfields. Those advances are not on the horizon. Just the opposite is true, since we have indications, based on long experience, heuristics, and even some rigorous proofs, that many of the important problems are really hard. Exponential complexity is hard to cope with.

Mathcad, a widely used mathematical software package, has some symbolic algebra features in it (and can invoke Maple to some extent in its premium version). A depressing thought is that this package probably would not have been any different had there been no research in computer algebra in the last 20 years. A similar situation prevails in other areas. The *Numerical Recipes* books are selling incredibly well, even though researchers in numerical analysis regard them as obsolete. However, they do contain a variety of useful algorithms in an easy to use format, and that is good enough for most applications.

The revisionist view of computer algebra presented above is one of many that argue that the role of research is changing (cf. [13]). As the scale of the research effort grows, and the easy problems are facets of our lives. As computers are becoming more powerful and more widespread, researchers in all fields are drawn to use computers for a variety of reasons. Sometimes it is to use email, or typeset their own papers. In other cases their jobs have evolved so that they absolutely require the use of computers. In still others, it is the availability of software tools, such as those in computer algebra, that is critical. Once they are hooked on computers, researchers tend to stay hooked, and use more tools as they learn how helpful they can be. This also helps increase their appreciation of the intellectual value of the effort that underlies the technology they use. This same process operates in computer algebra, computational number theory, computational group theory, computational algebraic geometry, numerical analysis, statistics, and other areas. It is easy to argue (as I will do in the next section) that the boundaries between these fields are fuzzy, and are rapidly disappearing, at least from the perspective of prospective users, who are beginning to think just of mathematical software.

The evolution of computer algebra systems is brought to mind most vividly by recalling what they were like 25 years ago, just as I arrived at MIT. In those ”ancient” days, there were already many programs. However, most were special purpose systems, were mostly batch-processing based, and depended on particular hardware and software configuration. Even Macsyma, which was then the most general system, and the most powerful one, and was interactive, was only available locally. It ran on a machine that was puny by today’s standards, that crashed frequently, and had to be accessed through a slow printing terminal. I vividly remember a discussion during a break at the 1976 SYMSAC conference, where most of us listened with great longing and not a little incredulity as Joel Moses described the idea of a portable briefcase-sized computer with a screen display that could run Macsyma. Well, we have had such computers, each more powerful than the machines that Macsyma ran on in the 1970s, for half a dozen years. Moreover, we now have systems such as Macsyma, Maple, and Mathematica, that are available on many platforms, and are used by hundreds of thousands of people. Further, these systems are much more powerful than those of a quarter century ago. For example, it is now possible to provide high level specifications of certain algorithms and obtain an automatic analysis of their average running time [5]. We have advanced far.

3. The Future (and Another View of the Past)

While there has been progress in the last 20-30 years, just how significant has it been? Some of the roots of computer algebra lie in AI, and in the hope that symbolic computation would lead to insights although sometimes memory constraints were also important, and occasionally there were questions about the rigor of the computations.)

I have written one paper explicitly about applications of computer algebra systems [11], in that case largely to problems in mathematics. I could easily write several more, based just on my own work. The point of mentioning this is to emphasize that I am not a totally unbiased observer, but an enthusiastic and grateful user. Computer algebra has given me many insights that I might not have had otherwise, and has made me much more productive as a researcher.

2. The Past

The scope of this note does not allow for a comprehensive overview of computer algebra. Unfortunately, there does not seem to be any single source that one could point to that would present a survey similar to that in the two articles by Calmet and van Hulzen in [1]. Those articles are still interesting to read, but they are both obsolete since there has been rapid development in the last 15 years. However, there are numerous books and surveys that can be consulted, such as [2, 3, 4, 6, 8, 9, 10, 14, 16, 17].

Computer algebra is usually traced to the 1950s. In principle, though, most of the pioneers of computing, starting with Augusta Ada Byron, understood that symbolic computations were possible on any digital machine. By now, this is known universally, and computer algebra is an established field. The proliferation of books on it (of which there are many more than those listed above, especially if we start including the numerous how-to manuals, and not just the serious textbooks) shows the vitality of the field. Further evidence of this vitality is provided by the annual ISSAC conferences, by the thriving *Journal of Symbolic Computation*, and by the numerous computer algebra papers that appear in a variety of other conferences. Advances, both theoretical and practical, have been numerous. Algebraic simplification, integration, polynomial and integer factorization, and solving differential or nonlinear equations are all much better understood today than they were two or three decades ago.

Computer algebra has attained substantial stature in the intellectual community. The reasons are similar to those that have led to higher evaluations of other areas of computational science, which used to struggle for respect in the academic hierarchy of prestige. Passage of time has shown that the problems in these areas are interesting and hard, and the results nontrivial. Most important of all, we are moving rapidly towards the digital age, and computer and communication networks are affecting all

COMPUTER ALGEBRA AND ITS APPLICATIONS: WHERE ARE WE GOING?

ANDREW ODLYZKO [1])

Abstract

Computer algebra is over 30 years old. It has a record of major theoretical advances as well as important applications. What about the next 30 years? They are likely to see a continuation of the trend towards providing tools accessible to large populations, as opposed to aiming at a select group of leading researchers.

1. Introduction and Disclaimer

Speculation about the future, even informed speculation, is inherently uncertain and subjective. Since speculating is what this note is about, I decided also to be subjective about the past. The historical view that is offered below is only a set of personal impressions, not a scholarly study.

I have never been part of the mainstream computer algebra community. On the other hand, I have been on its periphery since the beginnings of my scientific career, as a graduate student at MIT in the early 1970s. I was fortunate enough to be at what surely was then the most active research center in computer algebra, and had access to Macsyma. I used Macsyma in several research projects, including my Ph.D. thesis, where it helped to improve and strengthen my results (which used analytic methods to prove estimates in algebraic number theory). Ever since that time, I have been an enthusiastic user of several computer algebra systems. While I do have a couple of papers in computer algebra (some purely theoretical, some that have influenced implementations), my influence on this area has been slight, and probably due mostly to the bug reports, complaints, and suggestions that I have made. On the other hand, computer algebra has had a major influence on my own research work. I have not only used several of the major systems, but have written innumerable special purpose programs for various types of symbolic and related computations, computations that I could not do using available computer algebra systems. (The main reason for writing the special purpose codes was to achieve higher speed,

[1] AT&T Research, Murray Hill, NJ, 07974, USA. Email: amo@research.att.com