Computer algebra and symbolic computational mechanics

Nikolaos I. Ioakimidis

Division of Applied Mathematics and Mechanics, Department of Engineering Sciences, School of Engineering, University of Patras, GR-265 04 Patras, Greece
e-mail: n.ioakimidis@upatras.gr

Abstract Although, undoubtedly, numerical computations are the computations of primary importance in computational mechanics, nevertheless, symbolic computations also constitute a significant related tool. Here at first an extensive review of the major computer algebra systems is made, i.e. (in alphabetical order) of Derive, Macsyma, Maple, Mathematica and Reduce. Next, several applications of symbolic computations to computational mechanics are mentioned in brief (including the computer algebra system used in each of them) as well as the related very recent research results by the author. Extensive conclusions with respect to the usefulness of computer algebra systems in computational mechanics are also mentioned and several comments concerning the trends and the possibilities in this very interesting application of computer algebra systems are made.

Keywords Computer algebra · Computer algebra systems · Computational mechanics · Applied mechanics · Symbolic computations · Derive · Macsyma · Maple · Mathematica · Reduce

Copyright © 1989 by the author

Final publication details This technical report was published in 1991 (with quite minor changes) in IACM Bulletin, which was the official bulletin of the ‘International Association for Computational Mechanics’ (IACM), Campus Nord — UPC, Universitat Politècnica de Catalunya (Technical University of Catalonia, Barcelona Tech), Barcelona, Catalonia, Spain. (Since 1996 IACM Bulletin was renamed to IACM Expressions.) The final publication details of this technical report are


Therefore, whenever reference to this technical report is to be made, this should be made directly to its above definitive publication in IACM Bulletin.
1. Introduction

During the last few months of 1989 we had the opportunity to become aware of a powerful area of computer-aided computations, that of symbolic (or formal) computations. The related field is usually called computer algebra and we feel that, in spite of the fact that this approach was first suggested in 1953, nevertheless, up to now it has not been sufficiently used in computational mechanics. We really believe that symbolic computations should take their proper position in computational mechanics, which generally uses only numerical computations at present. The purpose of this technical report is simply to review the state of the art in this area with the belief that its impact on computational mechanics has not yet reached its maximum.

In the next four sections, we will review in brief the computer algebra systems (that is, the commercially available software) with some personal comments on these systems (Section 2), some of the applications in mechanics that we have available (Section 3), some quite recent results of ours in the same area (Section 4) and, finally, the trends that we see and suggest for the future of symbolic computational mechanics (Section 5).

We will also give a sufficient number of references in all of the sections of this technical report although we understand that these are a very small portion of the available literature. We hope that this technical report may be considered as a brief introduction in a wide area of research and applications in computational mechanics by a novice in this area addressed to other novices. We will be glad if this contribution to symbolic computational mechanics brings the potential of computer algebra to the attention of those who have not had the opportunity to get acquainted with it up to now. The present technical report is just an introduction and not a complete review of the area.

2. Computer algebra systems

With the recent very wide commercial availability of very powerful microcomputers, workstations and minicomputers, computer algebra systems, that is programs using the computer mainly for symbolic computations (including symbolic manipulations in algebra, expansions and factorizations, polynomials and series, symbolic differentiation, partial differentiation and integration, Taylor and Laurent series, matrix manipulations, vector differential and integral calculus, differential, partial differential and integral equations, systems of linear and non-linear algebraic equations, integral transforms, complex analysis and residues and, essentially, any area in calculus, linear algebra, vector analysis, special functions, applied mathematics in general up to set theory and many more branches of mathematics) have become also more accessible, less costly, more powerful and with better documentation.

An interesting paper on symbolic manipulation of algebraic expressions in Fortran was presented in IACM Bulletin [1] very recently and gave us the idea of considering the preparation of the present technical report as well, but with quite different aims explained in Section 1. Yet, Fortran is not the preferred language of computer algebra. The main language in this area is Lisp (the so-called language of artificial intelligence, AI) in its various dialects, but, recently, it seems to have given its position to C (probably the most popular programming language for serious programs at present). Of course, other languages, like Pascal and BASIC can also be used (or rather have already been used), but for small computer algebra programs.

An old review of computer algebra systems and the related possibilities for symbolic computations (during 1981) was made by Stoutemyer [2]. Today there are five computer algebra systems which are the most common, easily accessible and sufficiently powerful. These systems are described and reviewed (through appropriate tests) by Foster and Bau [3]. These systems are (in alphabetical order) Derive [4], Macsyma [5], Maple [6], Mathematica [7] and Reduce [8]. Details
about the availability and documentation of these systems can be found in the References. Here we make some informal comments about these systems, which may essentially constitute a short guide for the interested researcher and scientist during the selection of such a system or systems.

Derive [4] is a very recent computer algebra system (released during 1988). Its present version is 1.56 (November 1989) kindly brought to our disposition by the creators of this software. (In fact, this is the only system available to us now.) Derive was designed exclusively for microcomputers under the MS-DOS operating system and with the limitation of 640 kB of RAM memory. Derive has a sufficiently large collection of commands in all classical areas of symbolic computation, graphical possibilities, numerical possibilities in real and complex arithmetic with arbitrary accuracy, possibilities of defining functions and creating files for further use, etc. Moreover, Derive is very user friendly and quite easy to use. Its manual is also very good. It is written in a dialect of Lisp (muLisp) and its code is not available to the user. In spite of its restrictions in its library and memory management, we have already used this system very successfully (as will be described in Section 4). Furthermore, Derive is a compact system and not divided into independent parts for specific purposes. The main drawback of Derive is, in our opinion, the lack of programming capabilities (beyond the definition of functions), which does not permit its use for large amounts of symbolic computations in an environment with essentially indispensable programming capabilities.

Macsyma [5] is surely the most powerful computer algebra system. Designed for mainframes, it was transferred to the microcomputer environment just this year (1989). Its length and possibilities cannot be compared with any other system. It supports a very large collection of functions, commands, operations, etc. Moreover, it is a tested system, since it has been used since 1969. It is written in Lisp and its code is not available to the user. It requires about 40 MB of hard disk capacity and about 6 MB of RAM on a microcomputer. Its use is restricted to 80386 microcomputers. Of course, it is also available on a very large variety of workstations, minicomputers and mainframes. Surely, this is the appropriate system for the serious user, the professional, who has available the necessary hardware. Although it cannot be said to be a modern computer algebra system, surely it is still the standard in computer algebra software. Its main disadvantage is the relatively low speed. The support of a mathematical coprocessor for numerical computations is a partial remedy to this situation.

Maple [6] is a more modern computer algebra system (since 1980) also recently available for MS-DOS microcomputers. It is written in C and its code is mainly accessible to the user. It also has versions for many more workstations, minicomputers and mainframes. For microcomputers it is also restricted to the 80386 family. It seems that it does not support a mathematical coprocessor. Its speed, although better than that of Macsyma, does not seems excellent especially for numerical computations. The idea in Maple’s architecture is to have just a very small kernel and load the applications on the memory only if and when required by the user. This permits the use of the system with a moderate amount of available RAM (possibly 640 kB) for ordinary applications. Moreover, the user can easily construct his own programs in the Maple language, very easy to learn. Maple seems to be a light (contrary to Macsyma), but very powerful, computer algebra system.

Mathematica [7] is the most modern computer algebra system (together with Derive), released in 1988. Today’s version is 1.2. Mathematica borrowed and enhanced many good ideas from other systems, combined together, in the C language, with the aim that it become the favourable system for any mathematician. It has excellent symbolic, numerical and graphical possibilities, as well as interface capabilities (for example, with \TeX—a standard mathematical word processor). We feel that its creators aimed to create the unique system for mathematicians, engineers, scientists and students. For this reason, they included a very large library of functions and a lot of facilities (such as the \TeX interface and excellent graphics). These are two main new ideas in Mathematica not
explicitly included in any other system: (i) the ability to learn very easily not only function definitions but also mathematical rules and (ii) the facility of preparing research reports, papers, etc. directly by combining the results of Mathematica with text by the user (these are Mathematica's notebooks). This will minimize errors from typing formulae and \TeX \ will permit a really professional appearance (e.g. in the preparation of a whole book). Of course, both of these possibilities are not so pioneering, since the preceding computer algebra systems can also be used for the same purposes. Among the drawbacks of Mathematica we mention the fact that it requires 1 MB of extended memory for microcomputers (although it is or it is planned to become available essentially for any kind of computer) with an 80386 processor, the fact that its symbolic capabilities (e.g. in integration) seem to be sufficiently inferior to those of Macsyma, the lack of accessibility in the code (besides special-purpose files), as well as the fact that even now minor errors in the code may exist. This is natural for completely new programs. We believe that the wide advertisement of Mathematica together with its really excellent documentation and its various facilities and power will permit it to become a very popular program in the area very soon. Its ability to learn permits the experienced user to prepare a separate file, for example, for Mellin transforms (useful in plane elasticity) and to obtain essentially any such transform by using this file in just few lines of the Mathematica language. Moreover, the casual \TeX \ user will appreciate the excellent and error-free appearance of his symbolic and numerical formulae in his final text through Mathematica. Finally, Mathematica was announced to already support Intel’s mathematical coprocessor and Weitek’s mathematical coprocessor for 80386 microcomputers. We feel that for the first of these coprocessors the corresponding version is available now. We have not seen research papers in computational mechanics using Mathematica and this is natural for the present time.

Finally, Reduce [8] is the most popular computer algebra system. It was originally available in 1968 with successive new versions up to now. Its extremely low price, its availability on any kind of computers (from MS-DOS microcomputers to supercomputers without restrictions), its strong programming capabilities, the availability of its code, the availability of its Lisp language, its portability from microcomputers to mainframes, the lack of errors in the code, the possibility of loading only the appropriate parts of it in each application, its very good documentation, etc. make it the ideal computation surrounding for the programmer with inexpensive hardware. Of course, it does not support graphics, a wide collection of functions, a mathematical coprocessor or extended/expanded memory in microcomputers, but its advantages are also great. From the engineering point of view, it is also the most popular computer algebra system (as is clearly seen from the research papers already appeared).

Of course, the above systems seem to be the most popular ones. Additional systems are also available and successfully used in practical applications. Some of these tend not to be available any more (such as the really very popular system \texttt{muMATH}, having now given its position to Derive). There are several books on computer algebra, such as those by Akritas [9] and Davenport et al. [10]. We recommend the latter (as much more friendly to the engineer than the first), but a look in the SIGSAM Bulletin [11] of ACM (a quarterly publication of the Special Interest Group on Symbolic and Algebraic Manipulation of the Association for Computing Machinery) seems to be also an excellent source of information on symbolic computations. Some recent collections of papers (engineering oriented) seem also to be of interest in computational mechanics [12, 13] although the reviews of software (Reference [14], for example) and the popular articles (Reference [15], for example) also contain important comments.

The conclusion is that most of the available computer algebra systems work in essentially the same way, offering comparable commands, an always interactive and, generally, not imperative environment and languages which are quite similar and always friendly to the user contrary to most of the classical languages.
3. Computational mechanics applications

We report some applications of computer algebra systems in computational mechanics. Clearly, we by no means wish to exhaust or completely review this possibility.

Five applications of *Macsyma* [5] to random vibrations of mechanical systems can be found in Reference [16]. Vibration analysis of systems, possessing damping, with Wilson trial functions by using *Reduce* [8] was studied in Reference [17]. Furthermore, buckling and instability problems for columns were studied in Reference [18] (by using *muMATH* and *Macsyma*), in Reference [19] (by using *Reduce*) and in Reference [20] (buckling and postbuckling analysis with *Reduce* again). More difficult is the case of buckling of polar orthotropic circular plates resting on an elastic foundation, having been studied in Reference [21] (by using *Reduce* once more). In the classical finite element method and its generalizations computer algebra systems were used for the accurate computation of the stiffness coefficients and related quantities. Two such papers are References [22, 23] with the use of *Macsyma* and *Reduce*, respectively. (Additional references can be found in Reference [22].) To be sincere, we found most interesting the cases of plane elasticity and plate bending problems (probably, because of personal experience). Two such references are References [24, 25]. In Reference [25] no concrete computer algebra system was used, but in Reference [24] the Altran system was used. Sufficient symbolic results show this approach [24]. It is unfortunate that only rectangular plates were studied in both of these references and, also unfortunately, no additional related references are reported there or have come to our attention.

4. Recent results

Having ‘discovered’ symbolic computations rather recently, we were happy to see their vast possibilities in plane elasticity problems. In References [26–33] we had the opportunity to prepare symbolic results either manually [26, 27] or, better, by *Derive* [28–33]. The last results were prepared during the last two months. We really appreciated the potential of computer algebra in our symbolic and numerical computations included in these references. Although we have used *Derive* (with rather limited capabilities and almost no programming capabilities), we reduced many times the necessary effort by us in obtaining our results. We have tried to obtain access to additional computer algebra systems, but we have failed up to now both privately and through the Computer Centre and the Institute of Computer Science in our University.

The plane elasticity problems where we have used symbolic computations are the following ones: the equations of caustics about a crack tip [26], the stress intensity factors under loading including a symbolic parameter [27], the crack tip elastic stress field [28], the orders of singularity at wedge apices [30], Chebyshev approximations to stress intensity factors in the case of loading including a parameter [31], the location of interface and other cracks [33] and the solution of singular integral equations [29, 32] either directly [29] or iteratively [32]. Additional applications of computer algebra software are also planned for the next year, 1990.

5. Conclusions

Beyond the brief description of today’s popular computer algebra systems, some of the applications of computer algebra in computational mechanics, our very recent related results and a moderately extensive related literature to follow, one of our aims here is to express our claim that there are or, better, there should be no barriers between symbolic and numerical computations as one might have assumed in the past.

In our results we suggested the term SAN for the semi-analytical/numerical or symbolic–analytical/numerical approach, where computations are made simultaneously with symbols and
numbers. We believe that this approach, where no exact results are expected, but, rather, approximate results are welcome, will prove very useful in future. Modern computer algebra systems are written in C (which efficiently supports numerical computations) and, moreover, mathematical co-processors are also supported and easily accessible hardware (microcomputers) is widely available as well.

This possibility is shown in the best way in References [29, 32], where the SIE (singular integral equation) approach, a classical method for the approximate solution of elasticity (and many more) problems in computational mechanics is combined with symbolic computations. This did not affect its approximate character. Computer algebra cannot lead to exact results in computational mechanics in cases where such results are not predicted by theory. What can be done is the incorporation of symbols in the approximate numerical methods (directly borrowed from numerical analysis) so that the results of computations can be much more general than before, since, in this way, they are applicable to a whole class of geometry or loading conditions because of the symbols appearing in these results. (The Green’s function method for cracks, where the concentrated load should move along the crack or boundary for the determination of this function, seems to be an ideal application of this approach.) Therefore, the aim of computer algebra in computational mechanics (to the writer’s personal opinion) is not the change of algorithms (e.g. those for the iterative solution of SIEs), but, rather, the more or less trivial generalization of these algorithms so that symbols be incorporated in them.

Of course, it is understood that, in such a working computational environment, the computer will require much more time for the solution of a concrete problem than before. This is true, but the gain that we have from obtaining results of general applicability and, moreover, from ridding ourselves of tedious and, probably, error-prone symbolic computations (when necessary) makes this increase in execution times not important. On the other hand, the power of modern computers in MHz or MIPS increases exponentially with time. Moreover, Mathematica was designed to support both Intel’s and Weitek’s mathematical coprocessors (in appropriate versions), being, undoubtedly, very rapid in floating point numerical computations. (Macsyma, although less rapid, also takes advantage of Intel’s coprocessor.) Therefore, there is no essential reason for avoiding the use of this software, also having in mind the possibility of direct transfer of the results to Fortran (or other) programs for further processing by standard available routines. This collaboration of computer languages (e.g. Mathematica and Fortran) also seems very promising for computational mechanics applications. The bilingual and, why not, multilingual environment in computational mechanics, supporting both numerical and symbolic computations, seems to be a reality in engineering.

In a recent review paper [34], Schwarz claimed that the ability of computers to perform automatic analytical calculations is probably even more severe than the introduction of computers for performing numerical calculations about forty years ago. We are curious to see the future orientation of the application of computers to computational mechanics. Artificial intelligence and expert systems are also very modern and active branches in engineering (see, for instance, the review by Shaopei [35] in the IACM Bulletin), whereas current Lisp’s versions (essentially the standard Common Lisp), so popular in computer algebra and artificial intelligence, tend to support numerical computations equally well as symbols.

Restricting ourselves again to computational mechanics, we wish and expect to see the appearance of one more journal, but devoted to that part of the related research where SAN methods will be devised, tested and used in applications.

Finally, we would also be happy to see special interactive computer algebra software oriented towards computational mechanics, that is having commands for critical loads, stress intensity or stress concentration factors, best geometry parameters in mechanics problems and so on. Unfortunately, we are unaware of any such effort at this moment.
References


---

3 All the links (external links in blue) in this section were added by the author on 25 January 2018 for the online publication of this technical report. Moreover, final publication details were added in References [26–33].


