Symbolic algebra systems in teaching and research

A. D. Fitt

Mathematics Group,
Royal Military College of Science,
Shrivenham, Wiltshire, UK

Abstract Recent years have seen great advances in the development of Symbolic Algebra Systems (SAS). An SAS is a tool which allows symbolic calculations to be performed using a computer, and may handle anything from simple differentiation to complex nonlinear differential equations. A number of different systems are now available, including REDUCE, MAPLE, MACSYMA, muMATH, SMP and Scratchpad. Using MAPLE as our chosen SAS, illustrations will be given of the power of such tools. We also consider the place of an SAS in modern teaching and research. Although on a simple level an SAS may serve only to save time and ensure accuracy, more specialized applications show that it may now be feasible to carry out calculations which were not possible before such systems were available. In this respect an SAS may be a major contribution to research. Some comments are also made concerning the limitations and faults of MAPLE, and possible future developments.

Key words: Symbolic algebra, Software development, LISP environment.

1. Introduction

The last decade has seen many changes in the way in which mathematicians run their daily lives. The increasing availability of powerful mainframes and microcomputers and the growing popularity of word processors, electronic mail and other aids to communication mean that although the overall goals and

ambitions of mathematicians may not have altered much, the means by which they are realized have altered significantly. One of the most important technological innovations has been the emergence of the Symbolic Algebra System (SAS). An SAS is a software system (possibly with allied dedicated hardware) which allows mathematical calculations to be made using a computer (or a microcomputer). In this definition the important word is *mathematical*. Calculations are performed from a totally different standpoint to say FORTRAN computations, and no approximations are made. This philosophy will be more clearly explained in subsequent sections.

2. Symbolic Algebra Systems Presently Available

The symbolic algebra system is essentially a development of the 1980s, and so although there is quite a wide range of packages available, the market has not yet been 'flooded'. Although in subsequent sections we will illustrate our ideas by considering a specific system (namely MAPLE) it is illuminating to try to give a summary of the capabilities of some other systems which are currently in use. A comprehensive comparison of different systems is given by Harper, Wooff & Hodgkinson (1988). They consider the six packages MACSYMA, REDUCE, MAPLE, muMATH, SMP and SCRATCHPAD. All are written at the most basic level in what may broadly be descibed as a 'LISP-type' language, and most can either be run interactively, the session proceeding on a 'question and answer' basis, by running a file containing a list of 'interactive' commands, or by writing code in the basic language. All share the capacity to read from and write to external files, a capability which is essential for longer calculations. Their distinguishing features may briefly be summed up as follows:

MACSYMA: This is probably the most powerful SAS currently available. It performs well on many standard problems, and is particularly good at integration, outperforming standard integration textbooks (for example Gradshteyn & Ryzhik (1980)). The price is paid in terms of machine requirements however and the CPU time demands which the system makes render it unsuitable for a multi-user environment. It is capable of producing both TEX and FORTRAN output, as well as graphics.

REDUCE : REDUCE is probably the oldest SAS and was designed specifically for mainframes. It is also the most popular, and is capable of producing FORTRAN output It cannot, however, solve ordinary differential equations

MAPLE : This system is powerful and has the advantage that it has been designed with multi-user environments in mind. It is also able to produce FORTRAN and graphical output.

muMATH : Although the range of muMATH is somewhat limited (for example it has few matrix handling routines, cannot solve simultaneous non-linear equations, and knows few special functions) it does have the distinct advantage that it was designed specifically for microcomputers It is therefore compact and makes small demands on storage and CPU time.

SMP · This is another large and powerful system, with the capability to produce FORTRAN and graphical output. One feature that distinguishes it from all of the other systems which we have mentioned is that polynomial coefficients are represented as floating point numbers by default. This must be regarded not only as dangerous from a conditioning point of view for some calculations, but also contrary to the philosophy of symbolic algebra calculations where approximations are 'never' made.

SCRATCHPAD : SCRATCHPAD is really at its best when used as tool by pure mathematicians It is capable of both TEX and FORTRAN output. It is not able to solve ordinary differential equations.

As well as these systems, there are some others, notably ALTRAN, CAMAL and SHEEP but as these are ouside our experience no comment can be passed on them It is also worth commenting that we have specifically noted above the capacity of each system to produce FORTRAN or TEX output. The user who has had to code or reproduce very long formulae will appreciate how important this capability can be

3. What MAPLE can do for you

From now on for the sake of definiteness, we will consider only calculations performed by MAPLE. We have chosen this SAS not because we believe it is the best, but simply because it is the most well-known as far as the author is concerned. MAPLE was developed at the University of Waterloo, Canada during the period 1980-1985. As well as the basic kernel, MAPLE includes a large library of procedures and functions, some directly online and some easily available by reading other libraries There is also an online HELP facility which gives definitions and examples

One of the ideas which is crucial to the success and feasibility of MAPLE is that it exists in a 'LISP environment'. By this we do not mean necessarily that the system is written in LISP (actually MAPLE is written in the language C, with the system kernels existing in the form of macros which can be translated by a dedicated macro-processor into versions of the kernel written in C) but rather that to the user, MAPLE has the following characteristics ·

(i) MAPLE can be used very much as an interactive language This means that on a simple level where only a small number of easy computations are being carried out, sessions with MAPLE are of the 'question and answer' variety When more complicated procedures and routines must be used however these are still built up in an interactive fashion, with each more complicated function defined locally in terms of a heirarchy of easier functions. This is completely different from, for example, the FORTRAN idea where the usual sequence of events is to write large chunks of code, compile them, and then see if they work.

(ii) The MAPLE environment supports, indeed actively encourages, the process of recursion, the ability of a procedure to call itself. This feature makes the environment especially suitable for mathematical computations as it allows great elegance and efficiency

(iii) 'Things' in the MAPLE environment may be variables, lists, constants or programs themselves, to name but a few possibilities 'Things' may all interact with each other and operate on each other Moreover, it is possible to have a variable which is itself a program. This gives us the necessary equipment to write pieces of code which themselves write or modify other pieces of code

(iv) The basic language of the MAPLE environment is in a very 'pure' form, uncluttered by the seemingly arbitrary additions to the basic syntax which cloud

other languages such as FORTRAN. It thus forms a 'distilled' starting point from which to work and so has great flexibility.

(v) In many ways the structure of the MAPLE kernel reflects rigorous mathematical thinking as the basic approach is an axiomatic one. This may also be said to be true to a certain extent of other programming languages but it is the 'distillation' mentioned above inherent in the MAPLE environment which makes this form especially suitable as the starting point for far more complex sorts of work which may encompass parts of proof theory.

Given that MAPLE enjoys all these advantages, what can it actually do for the user ? A few examples can illustrate this most succinctly, though limited space makes it possible to indicate only a small part of what MAPLE is capable of. To begin with, MAPLE works in infinite length arithmetic. All rational numbers are held in the form n/m where n and m are integers, and non-rational numbers are stored as themselves, so that for example $\sqrt{2}$ appears as $2^{1/2}$ and π as 'Pi'. Although in some cases results in rational form may be somewhat awkward to handle, floating-point versions of the numbers can easily be produced. As an example of this, consider the evaluation of the binomial probability of throwing say 50 heads and 50 tails in 100 throws of a fair coin, and suppose that for some reason we wish to evaluate the answer to 30 decimal places. The MAPLE commands for this would appear as follows (> is the MAPLE prompt and the double quotes mean 'the entity above' To signify commands from a MAPLE session we have used plain typeface, and displayed the MAPLE responses exactly as they would appear at the terminal, namely centred and 'prettyprinted' with exponents and subscripts on different lines)

```
> 100!/50!/50!*(1/2)**100;

                                126114180681955241668515621 57
                               ------------------------------
                               158456325028528675187087900672

> evalf(",30);

                              .079589237387178761498127050 2422
```

Differentiation, integration and the solution of some ordinary differential equations is easy to carry out using MAPLE. For example

```
> diff(exp(sin(x)),x);
```

$$\cos(x)\ \exp(\sin(x))$$

```
> int(1/x,x);
```

$$\ln(x)$$

```
> dsolve(diff(v(t),t)+2*v(t)*t = exp(-t),v(t));
```

$$v(t) = (-\ 1/2\ \mathrm{Pi}^{1/2}\ \exp(-1/4)\ \mathrm{I}\ \mathrm{erf}(\mathrm{I}\ t\ -\ 1/2\ \mathrm{I}\ +\ C)\ \exp(-\ t^2)$$

The last example shows that MAPLE not only knows about things such as arbitrary constants, but can also handle complex numbers and has heard of the error function. MAPLE also is capable of spotting some situations where a seemingly difficult problem is actually an easy problem in another form. The example we give here is an equation which is a quadratic in $\log(x)$. The reader will appreciate that it is no small feat for MAPLE to recognize this fact and solve accordingly

```
> solve(log(x)*log(x)+5*log(x) + 6 = 0,x);
```

$$\exp(-2),\ \exp(-3)$$

Also, when no analytic solution exists, MAPLE can be asked to try to give a numerical solution ·

```
> solve(log(x)+sin(x)=0,x);
```

```
Warning: no solutions found
```

```
> fsolve(log(x)+sin(x)=0,x);
```

$$.5787136435$$

As an example of a calculation which MAPLE is able to perform where normal methods fail, consider the solution of the $N \times N$ system of linear equations

$$Ax = b$$

where b is the $N \times 1$ vector with all entries set to unity, and A is the notorious Hilbert matrix given by

$$A_{ij} = 1/(i+j) \quad (i,j = 1 .N).$$

Any manipulations which MAPLE carries out on these equations will, of course be exact as they are expressed in terms of fractions. We compared the solution of this problem via the MAPLE 'linalg' routines for solving simultaneous linear equations and finding determinants and condition number for various orders N of the problem to that obtained using the NAG FORTRAN double-precision routine F04ARF which uses Crout's method of factorization, and is well-known for reliability and robustness. For $N = 2$ the solutions given were:

$$MAPLE \cdot x = \begin{pmatrix} -6 \\ 12 \end{pmatrix} \quad F04ARF \cdot x = \begin{pmatrix} -6.000000000000000 \\ 12.000000000000000 \end{pmatrix}$$

so that both methods were performing equally well. For $N = 5$, the NAG routine still gave answers correct to 10 significant figures, but when N reached 10 the numerical method was seriously in error, so that for example x_3 which should be -102960 was given by -102972.7208248292 and x_5 as -3784168.723631925 (correct answer -3783780). By the time N had risen to 15, the NAG routine would not execute, returning with IFAIL=1 ('Matrix singular or too ill-conditioned for solution'). MAPLE was quite happy to continue to solve the equations exactly. This should not be interpreted as a criticism of the NAG routine, since for the 15×15 problem the determinant of A is 1/146538740183666877481333971725269067959384123502368457086504025547136338728156717325918547786279155714360764006400000000000000000000000 or roughly 6.8×10^{-131} and the condition number of the matrix (defined as $\| A \|_\infty \| A^{-1} \|_\infty$) is 6008864201304641616585 or roughly 6×10^{21}. Whether or not is is ever sensible to solve such problems is of course another question entirely.

As a final example of the capabilities of MAPLE, we illustrate one mode of the plotting package in operation. To be able to plot results which may consist of complicated algebraic expressions is an extremely useful facility, and the user has the added convenience that MAPLE knows about singularities and badly-behaved functions, and can deal with these itself. Figure 1 shows a polar coordinate plot of the function

$$r(\theta) = 1 + \theta \sin(1 + \theta \sin(1 + \theta \sin(\theta)))$$

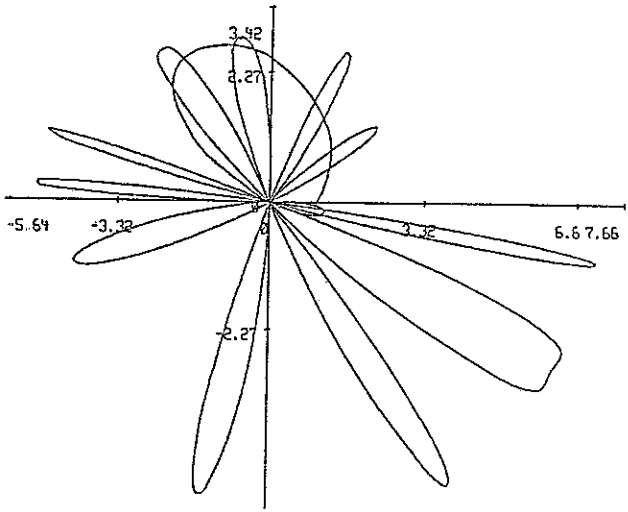

Figure 1 MAPLE plot

We are sure that the reader will agree that this is not a function which could readily be sketched.

4. Limitations and Failings of MAPLE

Having given some examples of calculations which MAPLE can perform successfully, we now turn to an examination of some of the failings and weaknesses of the system. The examples which we shall give below should not necessarily be thought of as 'black marks' for MAPLE - in some cases we are asking questions which are unreasonably hard and in others we are seeking to establish the present power of MAPLE by pushing it to the limit.

Firstly let us address the problem of simplification of results and expressions, a process that for a computer is fraught with difficulties Anybody who has ever tried to explain to a student how to simplify a long expression will know that most of us carry out such operations mainly by instinct. These sorts of processes are very difficult to code, even to such a receptive environment as MAPLE In many ways the problem is similar to that faced by chess-playing computers who fail to play strategic moves which to a grandmaster would instinctively 'look right'. All bound up in the same question is the notion that long calculations

'usually come out to be neat' if done properly - a long held belief which proves to be true surprisingly often. A good example of some of the deficiencies of MAPLE in this respect is given by factorization. The complicated polynomial

$$p(x) = 1536x + 4096x^4 + 3840x^2 + 4096x^6 + 1696x^{10} + 3936x^8 + 1456x^{12} + 2112x^9 +$$
$$5120x^7 + 3072x^5 + 5120x^3 + 1920x^{11} + 256 + 241x^{16} + x^{22} +$$
$$6x^{21} + 15x^{20} + 20x^{19} + 31x^{18} + 102x^{17} + 320x^{15} + 336x^{14} + 672x^{13}$$

is easily factored by MAPLE using the command

```
> factor(p(x));
```

$$(x + 1)^6 \, (x^2 - 2x + 2)^4 \, (x^2 + 2x + 2)^4$$

but the very easy

$$q(x) = 1 + 3x^{1/3} + 3x^{2/3} + x$$

which any first year student could factorize as $(1 + x^{1/3})^3$ yields merely itself under MAPLE's factorization routine. Notice also that in the final expression for $p(x)$, in spite of having performed an enormously difficult factorization on a large polynomial, MAPLE did not spot that

$$(x^2 - 2x + 2)(x^2 + 2x + 2) = x^4 + 4 \, !$$

Another critisicm which is often levelled at MAPLE is that it sometimes produces answers in an inconvenient form. The answer to this is of course that every user has his or her own preferences as far as expression presentation is concerned, and consequently it is not unreasonable to expect the user to know the correct commands to convert the expression into the desired form. It is true however that sometimes manipulation of subexpressions contained in large formulae is not as straightforward as it could be.

A good test of any symbolic manipulator is the ease with which it handles the operation of integration. A particularly difficult type of example is illustrated by the MAPLE command

```
> int(cos(x)/sin(x)/log(sin(x)),x);
```

which gives the MAPLE answer

```
ln(ln(sin(x))
```

Here MAPLE has integrated the function which it was told to, but has recovered a primitive which exists NOWHERE on the real line! ($\sin(x)$ is always between -1 and 1, so that its log has maximum 0 and consequently the log of that is complex). The problem here has come about really because MAPLE does not know quite enough about the properties of the logarithm. It does redeem itself however, for when the integration is carried out between two limits (say 1/10 and 2/10), 'evalf'ing and 'evalc'ing (evaluate to a floating point number and evaluate to a complex number respectively) does actually yield the correct answer. As far as more general integration goes, MAPLE's capabilities vary. For example, it is able to integrate

$$r(x) = \frac{\mathrm{diff}(f(x), x)}{1 + f(x)}$$

to get

$$\int^x r(x)dx = log(1 + f(x)),$$

but is unable to integrate the expansion of the expression $\mathrm{diff}(r(x), x)$ with respect to x to get $r(x)$. It is also true that for some integrals the user may have to help MAPLE slightly by making a judicious substitution, after which the computer can take over and do the rest of the work

One of the traps which MAPLE occasionally falls into constitutes a sacrifice in rigour for the sake of adding a new function to its library An example of the this is the MAPLE function $iscont([function], range)$ which claims to state whether a function is continuous or not over a given range. The function can return three different values · 'true', 'false' or 'FAIL' which indicates that the routine cannot handle the problem. Testing the procedure with the example

```
> iscont(sin(x)/x,x=-1..1);
```

gives the answer

```
false
```

which is wrong. Note here that we really do not know how MAPLE has arrived at this answer Clearly it has not produced the standard undergraduate 'ϵ, δ' argument, and for all we know it may have erroneously concluded that the product of a continuous and a discontinuous function must always be discontinuous

To conclude, we should realize that all of MAPLE's calculations must be [illegible] with reality in mind. MAPLE is not, and never will be the 'expert

system which makes experts superfluous' The somewhat esoteric game of scoring points off MAPLE by catching it out is valuable, and should be played at some juncture by every MAPLE user.

5. Conclusions and Possible Future Developments

The 1980's have seen Symbolic Algebra Systems elevated from computing curiosities to valuable and powerful tools for both teaching and research. This rise may be likened to the increase in the late 1960's and early 1970's in mainframe specifications which allowed realistically complicated scientific calculations to take place. Using an SAS it is now possible to undertake theoretical computations which were simply not feasible ten years ago simply because of the time it took to commit them to paper. In some senses symbolic algebra has now reached a crossroads in its development, for running parallel with advances in the field have been great achievements in the fields of expert systems and artificial intelligence. The great challenge in the 1990's will be to combine all of these disciplines to produce a new breed of symbolic manipulator which may have the ability to prove simple theorems, recognize abstract structures and show some elementary form of mathematical creativity. It is possible that such advances will also be linked to pure mathematics (as in the use of Lie groups to solve ordinary differential equations). Certainly if advances such as this can ever be made, the effort required to bring them about will lead to a massive increase in our own basic understanding of 'how mathematics works' It is to be hoped that throughout any such development, the rigorous general philosophy of symbolic algebra computations will be adhered to - we must always be willing to allow our systems to give the answer 'don't know' if the result is not absolutely watertight, and never fall into the trap of settling for second best as far as rigour is concerned in order to speed development or compete with other systems If these advances can be made, then the Symbolic Algebra System will assume a role orders of magnitude greater than the already indespensible one which it enjoys at the present time

References

Gradshteyn, I.S. & Ryzhik, I.M. (1980) 'Table of Integrals, Series and Products' Academic Press

Harper, D., Wooff, C. & Hodgkinson, D (1988) 'A Guide to Computer Algebra Systems' University of Liverpool Report 1988