SYMBOLIC AND ALGEBRAIC COMPUTATION

Göktürk Üçoluk *

February 1990

1 A Prologue

To introduce something new the best way is to start with some examples, here they are:

Consider the matrix **A** defined below:

$$\begin{pmatrix} 1+2x & x+x^4 & x+x^9 \\ x+x^4 & 1+2x^4 & x^4+x^9 \\ x+x^9 & x^4+x^9 & 1+2x^9 \end{pmatrix}$$

and we want to find the characteristic polynomial defined by:

$$det(\mathbf{A} - \lambda \mathbf{1})$$

As it is well known the result is a third order polynomial in λ:

$$\begin{aligned} &-\lambda^3 + (2x^9 + 2x^4 + 2x + 3)\lambda^2 \\ &+ (2x^{18} - 2x^{13} - 2x^{10} - 4x^9 + 2x^8 - 2x^5 - 4x^4 + 2x^2 - 4x - 3)\lambda \\ &- 2x^{18} + 2x^{13} + 2x^{10} + 2x^9 - 2x^8 + 2x^5 + 2x^4 - 2x^2 + 2x + 1 \end{aligned}$$

As a second example, consider the algebraic expression

$$f(x,c) = \frac{1}{\sqrt{x^2 - c^2}}$$

and assume we seek to find the following partial derivative:

$$\frac{\partial^6 f(x,c)}{\partial^4 x \, \partial^2 c}$$

Which has the result:

$$45 \times \frac{6c^6 + 101c^4x^2 + 116c^2x^4 + 8x^6}{c^{12} - 6c^{10}x^2 + 15c^8x^4 - 20c^6x^6 + 15c^4x^8 - 6c^2x^{10} + x^{12}} \times \frac{1}{\sqrt{x^2 - c^2}}$$

*Middle East Technical University, Department of Physics

As the third and last example let us have a polynomial in three variables:

$$
\begin{aligned}
&-750s^6x - 1500s^6y + 200s^5x^2 + 2025s^5xy + 3250s^5y^2 + 60s^4x^3 - 355s^4x^2y \\
&- 1950s^4xy^2 - 2000s^4y^3 - 24s^3x^4 + 250s^3y^4 + 2s^2x^5 + 51s^2x^4y + 124s^2x^3y^2 \\
&- 153s^3x^3y + 140s^3x^2y^2 + 825s^3xy^3 - 15s^2x^2y^3 - 150s^2xy^4 - 4sx^5y - 30sx^4y^2 \\
&- 29sx^3y^3 + 30sx^2y^4 + 2x^5y^2 + 3x^4y^3 - 2x^3y^4
\end{aligned}
$$

It is quite hard, infact imposible, to find out by conventional hand calculation that it actually factorizes as:

$$(x + 2y) \cdot (s - y)^2 \cdot (x - 5s)^3 \cdot (2x - y + 6s)$$

Those were exactly what we are able to perform with computer algebra systems.

2 Introduction

Since the early days of the sixties we are using the computers to perform symbolic and algebraic manipulations (*abbreviated as* **SAM**). In this talk we will attempt to give a very short overview of this field of computer science, mention about its problems, try to introduce the present state. But before doing all thess, the subject is worth, I believe, stating a definition. The question is "What is the field of SAM?" or in other words "What is Computer Algebra?".

> *It is the part of computer science which designs, analyses, implements and applies algebraic algorithms.*

SAM is in close encounter with the mathematical disciplines like algebra, analysis (calculus) and numerical analysis and in an imperative relation to logic. However one has to point out that there is a small difference in SAM compared to these disciplines. In algebra the general rules underlying an algorithm are the primary objects of study whereas in SAM the tool itself, the algorithm in all its aspects, including its efficiency and implementation, is the subject of concern.

It would not be a mistake to say that the origination of SAM programs in 1960's were due to some needs emerging in theoretical physics problems. In those days QED, for example, was trying to compute the sixth order corrections to the anomalous magnetic moment of the electron. The analytic form for the fourth order contribution was computed by hand in 1957. Because of the sheer magnitude of the calculation, the sixth order result, which involves looking at 72 diagrams, has been computed by a group effort involving many different independent researchers. To be sure the result is correct, each diagram was being computed by two different groups, usually with different SAM computational tools and a tool developed by Campbell and Hearn that later gave rise to the famous SAM language REDUCE.

3 Computer Algebra Systems

- **MACSYMA**, the most developed, but unfortunately available on only few kinds of computers. Also very large in code and expensive too!
- **REDUCE**, The most widely available of all computer systems. Not as large as MACSYMA, and much more convenient in price.
- **SCRATCHPAD**, A very new system, with extremely restricted availability, but which, because of its completely different structure, can overcome some limitations the other systems share. People mean that it is not user friendly, although I have not got the same impression.
- **MATHEMATICA**, Also a new, commercial, hybrid system. Also nice to draw fancy 3-dimensional graphs. Available on a range of computers from the MAC to the VAX.
- **MAPPLE**, A Canadian product, semi-commercial, still under development.
- **muMATH**, A system for micros. Compared to the ones above, a toy product. Gives a feeling of the early days of SAM, where you perform computation with rationals, substitution, differentiation etc. It has even the concept of matrices, but as it is said, not for professional applications. Sold extremely over-priced.

4 The present state: What Can Be and What Cannot be Done in SAM

Before getting deeper in the problems of SAM, it is worth saying a few words on the present abilities of the computer algebra systems. Especially users with no computer science background get somewhat disappointed when starting to play around with computer algebra systems. As a good example one can speak of the theoretical physicists which form also the oldest user group. Being in contact with this group over years one can immediately figure out that the demands (or better to say, the expectations) of a theoretical physicist from SAM is somewhat different what he (or she) gets. Theoretical physics is an area in which a good degree of discernment, high rate of intelligence is required. The theoretical physicist knows what his goal is. He can (usually) figure out the rough form of an uncalculated result. With this picture in mind he carries out the hand calculation and at each step arrived performs a consistency check; furthermore at each step he evaluates the present state of the calculation with a sight to get rid of the unwanted terms. In a sense he is guiding the calculation through a dark forest. He knows or decides at each step to apply which identity, mathematical trick or whatsoever. Unfortunately, the present state of SAM is very far from this end. In the above stated sense it has no intelligence at all. It has no

concept of "better". Its mind is simple, relays on a binary logic. A substitution is either always made (if it is told to do so beforehand) or never. The ability of decision on reasoning is absent: It will never do something like

> ...At this point I shall keep this $\sin^2\theta$ term because I know that at the next step I will perform a differentiation of another term which will give a $\cos^2\theta$ contribution, then I will be able to add them and substitute a 1 instead ...
> ...But the second $\sin^2\theta$ term, yes that must be immediately substituted by $\cos^2\theta - \cos 2\theta$ since ...

But it will be able to take the 1001th derivative of $(1-x^2)^{1/2}$ since the steps to do this is quite simple. The length of the calculation, the size of the intermediate results is of minor importance for it, provided the algorithm is well defined. What can be done in SAM is listed below

- expansion and ordering of rational functions
- symbolic differentiation of rational functions
- procedural facilities for defining functions
- substitution and pattern matching (*in an orthodox and deterministic manner. Always one directional substitution*)
- automatic simplification of expressions (*e.g., cancellation of the Greatest Common Divisor*)
- calculations with symbolic matrices
- Dirac algebra for the interest of QED, QCD
- various transformations like Fourier, Laplace etc.
- calculation with complex quantities
- arbitrary precision arithmetic
- facilities for solving some differential and integral equations
- symbolic integration
- factorisation of multivariate polynomials
- exterior form algebra

Two of these items are worth talking about. One is the symbolic integration. Indefinite integration is now a mathematically solved problem. That means there exist a well defined algorithm (as it exists for differentiation) that decides whether an indefinite integral can be taken in closed form or not and if the answer to this is yes then what the result of the integration is. Unfortunately, the algorithm is very complicated and up to now only SAM applications, which do not implement the whole of the algorithm have existed. The theory which was proven 20 years ago is based on an old conjecture originally mapped out by Liouville and Hermite:
If a function $u(x)/v(x)$ where u and v are polynomials over an algebraic extention field is integrated then the result will have the form

$$p(x)/q(x) + c_1 \log(v_1) + c_2 \log(v_2) + \dots$$

Here p and q are again polynomials of the same field and v_i are factors of the denominator of the integrand.

The second is the factorisation. This is may be the most impressive application field of SAM. Using advanced algorithms adopting p-adic methods, nowadays it is possible to factorise multivariate polynomials consisting of a thousand of terms in less then a minute of time.

As it is seen from the list above, some fields which would be of great interest are absent. There exists preliminary work on some of them but no satisfactory SAM applications have existed yet. I would like to mention some of them right away:

- non-abelian computation (*eg. calculation with Lie algebra valued entities*)
- symbolic indicial tensor manipulations
- graphical study of symbolic parametric functions
- expert system applications for SAM (*By this we mean a system that would take decisions among various calculation methods, justifies and explains them*)

5 An Inside Glimpse to the Problems of SAM

It is for shure that systems performing computer algebra share some problems of numeric computation, for instance: an efficient way to access indexed quantities, a possibility of parallelling the computation, methods to improve the efficiency in number crunching. But, frankly speaking, SAM has much not in common than in common with numerical computation. In numerical computation the most complicated domain is the set of floating point numbers. That is so even for complex and matrix calculations, since you are dealing with vectors of floating point numbers. Furthermore in the computational process all the intermediate results and the final result belong to this set, namely the set of floating point numbers. This simplifies the problem of data

representation a lot. But when it comes to SAM implementations you have another picture. Soon we will enter the problem of data representation in SAM, but before doing that I would like to emphasize some *other* headlines of basic technical problems which are encountered in SAM implementations.

- Even the simplest domain has to include all the functional and representational concepts of calculus that we are used to.

- The indivisible (or irreducible) piece of information in SAM consists of numbers and strings (so called ATOMs). These entities have to be accessed very frequently during the computation, so, special data access techniques, like hashing have to be used.

- Advanced algorithms for GCD calculation, factorization and integration heavily demand arbitrary precision and modular arithmetic.

- SAM has to cope with the intermediate-expression-swell problem. To get a notion of this consider the problem where you have to compute the characteristic polynomial of a matrix let's say of dimension 5, even if the matrix consists only of integers, the system has to perform 120 polynomial multiplication, store these intermediate polynomial results and then add them to get the final determinant, which will be a univariate polynomial of degree 5, that means with integer coefficients. Solution to the problem of intermediate-expression-swell requires efficient dynamic memory handling and advanced garbage collection techniques.

- Consider

$$\frac{x^2-1}{x-1}$$

  everybody would agree that the expression with the GCD cancelled out is more compact a representation, the result has only two terms. But consider now

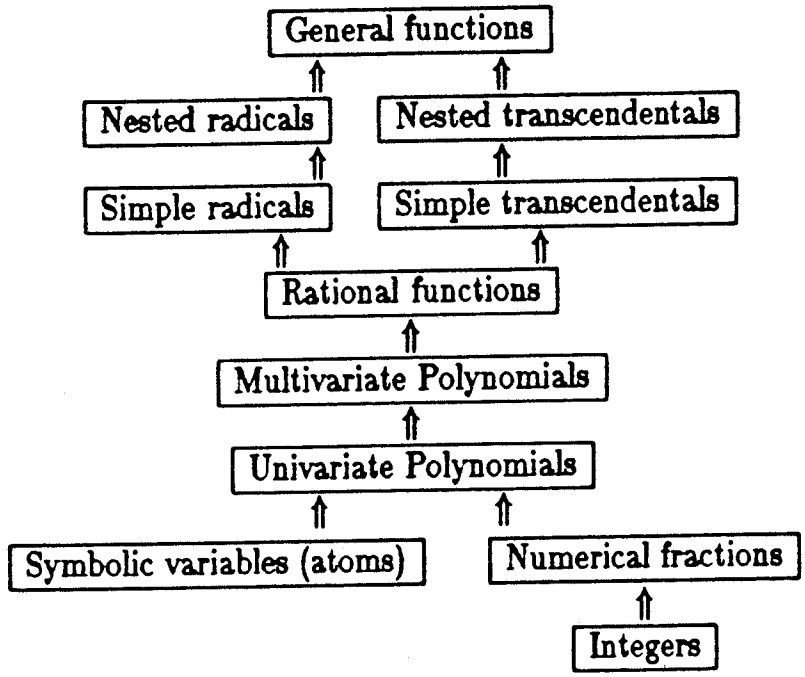
$$\frac{x^{100}-1}{x-1}$$

  the same process will lead to a polynomial with 100 terms! Similar problems show up in the decision whether a polynomial multiplication shall be kept in factor form or expanded. So, the system must allow a control over the simplification style.

Beside all these items, current research intensifies on developing/improving algorithms for factorization of multivariate polynomials, fast GCD calculation, integration in algebraic extension fields, Non-Abelian computation, etc.

5.1 The Problem of Data Representation

The domain entities in SAM can be viewed in a hierarchical relation to each other. Where, at every level that we go up in this hierarchy we have a freedom of choice for the corresponding data representation. Roughly the hierarchy is something like below:



5.2 Representation of Integers

Most numerical computation oriented programing languages treat the integers as a finite set. The dimension of which is usually determined by the hardware. So for a 32 bit processor it is something like $\{-2^{31}, \ldots, 2^{31}-1\}$. It is very easy to say "my data is small, therefore the answer will be small, so I shall not need large integers and a finite set will be enough". But this argument is completely false in SAM terms. To clarify our point suppose we want to calculate the GCD of the following two polynomials:

$$\begin{aligned} A(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5 \\ B(x) &= 3x^6 + 5x^4 - 4x^2 - 9x + 21 \end{aligned}$$

Using the polynomial generalization of the **Euclid's algorithm**, the first elimination gives a residue

$$-\frac{5}{9}x^4 + \frac{1}{9}x^2 - \frac{1}{3},$$

The subsequent eliminations give

$$-\frac{117}{25}x^2 - 9x + \frac{441}{25},$$

$$-\frac{233150}{6591}x - \frac{102500}{2197},$$

and finally,

$$\frac{1288744821}{543589225}$$

We could restrict the polynomials that are subject to the algorithm to be integers only then then sequence would be:

$$-15x^4 + 3x^2 - 9,$$

$$-15795x^2 - 30375x + 59535,$$

$$1254542875143750x - 1654608338437500$$

and finally,

$$1259333879550074310093115199218750 0$$

So, just after this result we were able to judge that the polynomials were relatively prime. The last integer calculated had 35 digits, that is 117 bits! Therefore, it is necessary to be able to deal with the true integers of mathematicians, whatever their size. Almost all computer algebra systems do this. Usually the implementation makes also use of the hardware characteristics just as we do by choosing a calculation base of 10 (due to our count of fingers). It is wise to choose half the largest integer that the hardware allows to be the base. Once we have chosen such a representation, the four arithmetic operations are, in principle at least, fairly easy. Two digits can be added without much of difficulty. Certainly, multiplication and division will be more complicated, but even for these operations some hardware provide special registers.

5.3 Representations of Fractions

Fractions, in other words, the rational numbers, are almost always represented by their numerator and denominator. As a rule they must not be replaced by floating point numbers since this causes not only loss of precision, but completely false results.

The main problem with fractions is the calculation of the GCD. Therefore, most of the SAM systems manifestly keep the numerator and denominator mutually prime. That means the GCD cancelled. Such fractions are called *reduced*. It is trivial that the four arithmetic operations are trimmed for efficiency considering this property.

Consider for example multiplication:

$$\frac{a}{b} \times \frac{c}{d} = \frac{p}{q}$$

The obvious way is to calculate $p = ac$, $q = bd$ and then remove the GCD. But if we keep the fractions always manifestly reduced then we can conclude:

$$gcd(p, q) = gcd(a, d)\, gcd(b, c)$$

Same would be true for addition:

$$\frac{a}{b} + \frac{c}{d} = \frac{p}{q}$$

Usually we calculate $p = ad + bc$ and $q = bd$, but it more efficient to calculate

$$q = \frac{bd}{gcd(b,d)}$$

$$p = a\frac{d}{gcd(b,d)} + c\frac{b}{gcd(b,d)}$$

5.4 Representation of Polynomials

All computer algebra systems are expected to manipulate polynomials in several variables (normally without any limitation on the count of variables). We can add, subtract, multiply and divide them (at least if the remainder is zero), but infact the interesting calculation is **simplification**. "Simplification" is a word with subjective meanings. To deal with this somewhat nebulous idea, let us define exactly two related ideas.

If we consider being algebraically identical a relation we can speak of equivalence classes of this relation. Those members which belong to the same equivalence class are algebraically identical. Here is an example:

an equivalence class of being algebraically identical
$(4x^2 - 1)$
$-(1 - 4x^2)$
$(2x - 1)(2x + 1)$
$4(x - 1/2)(x + 1/2)$

Now we state the definition of a canonical representation:

> *Choosing a* **canonical representation** *is to give a handle to favor one of the member of the equivalence class of the algebraically identical representations.*

So, if our SAM system converts any algebraic expression to its canonical representation then we can easily determine wether two such expressions are the same. We let the SAM system convert both of them to their canonical representation and then we check these representations for equality.

Is there a possibility to loosen this criteria. Do we have to define a canonical representation always, in order to decide for two expressions to be equal or not. If our SAM system provides a "normal representation" then the answer is yes.

> Any representation where "ZERO" is represented uniquely is called a **normal representation.**

A normal representation over a group gives us an algorithm to determine whether two elements **a** and **b** are equal: it is sufficient to see whether **a** – **b** is zero or not.

So, we can say that a **simplification** should yield a representation which would be at least normal, and if possible canonical. But we want much more, and that is where the subjectivity comes into play. We want the simplified representation to be **regular**. This criteria would exclude the following:
(we have defined $A = x^2 + x$)

$$\begin{array}{rcc} \textit{Object} & : & \textit{representation} \\ A & : & x(x+1) \\ A+1 & : & (x^3-1)/(x-1) \\ A-x & : & x^2 \\ A+x+1 & : & (x+1)^2 \end{array}$$

The internal data representation of polynomials is another problem. How to store a polynomial?, that is the question. The representations mainly are of two types. Namely, the *sparse representation* and the *dense representation.*

> *A representation is called* **sparse** *if the zero terms are not explicitly represented.*

Conversely, a representation is called **dense** if all terms (or at least those between the monomial of highest degree and the monomial of lowest degree) are represented—zero or non-zero. Our normal mathematical representation convention is sparse: we write $3x^2+1$ not $3x^2+0x+1$. The most obvious computerized representation of a univariate polynomial

$$a_0 + a_1 x + \cdots + a_n x^n$$

is an array of its coefficients: $[a_0, a_1, \ldots, a_n]$.
Since all the a_i, zero or non-zero, are represented: the representation is **dense.**

A less obvious representation, from the computing point of view, but closer to the mathematical representation, is the **sparse** representation: we store the exponent and the coefficient, that is the pair

$$(i, a_i)$$

for all **non-zero** $a_i x^i$ terms.
An example:

$$3x^2 + 1 \quad \Rightarrow \quad ((2,3),(0,1))$$

Please note that the n-touple is an ordered one. Otherwise the a polynomial would not have a single representation and therefore it would not be possible to introduce a canonical representation.

There exist languages very suitable for this kind of data handling. May be the best of those languages is LISP (a language preferred by most designers of computer algebra systems).

5.5 Representation of Multivariate Polynomials

There is a new difficulty as soon as we begin to calculate with polynomials in several variables: do we write $x + y$ or $y + x$. We are dealing with the same object (same in the sense of being algebraically identical) so, if we want a canonical representation then we ought to write only one of these expressions. We introduce a new idea — that of an order among variables. We have already used an order among the powers of variable, in the preceding section. The most common ways are the following:
{In the examples below consider the polynomial

$$(x + y)^2 + x + y + 1$$

and assume x is chosen to be the "principle" variable}

- **lexicographic :** We can decide that the main variable is to determine the order as exactly as possible, and we will consider the power of other variables if the powers of the first variable are equal.
  Example : $x^2 + 2xy + x + y^2 + y + 1$
- **total degree then lexicographic :** We can decide that the total degree of the monomial is the most important thing, We use the previous method to distinguish between terms of same total degree.
  Example : $x^2 + 2xy + y^2 + x + y + 1$
- **total degree then inverse lexicographic :** Similar to above but to be high in the inverse lexicographic list is a reason to be treated later.
  Example : $y^2 + 2xy + x^2 + y + x + 1$

Above the second and the third are not equivalent, but with the variable order inverted. To show the difference consider

$$(x + y + z)^3$$

With the second way we get:

$$x^3 + 3x^2y + 3x^2z + 3xy^2 + 6xyz + 3xz^2 + y^3 + 3y^2z + 3yz^2 + z^3$$

The third way would do (with a reversed lexicographic order):

$$x^3 + 3x^2y + 3xy^2 + y^3 + 3x^2z + 6xyz + 3y^2z + 3xz^2 + 3yz^2 + z^3$$

Lexicographic representation has an interesting consequence: as all the powers of the main variable are grouped together we can therefore consider the polynomial as a polynomial in that variable, whose coefficients are polynomials of all the other variables. So, the polynomial in the first case above, can be written in the form

$$x^2 + x(2y + 1) + (y^2 + y + 1).$$

This form is called **recursive**, in contrast to the **distributed** form $x^2 + 2xy + x + y^2 + y + 1$. The recursive form is used by most systems.

5.6 Representation of Rational Functions

As known, a rational function is a polynomial divided by another polynomial. Here again, we have the problem of a canonical representation. It is apparent that $(x-1)/(x+1)$ is algebraically identical to $(x^2-2x+1)/(x^2-1)$. The definition of "canonical" implies that we have to declare at least one of them to be not canonical. The natural choice is to say that any rational with the GCD not removed is non-canonical. Unfortunately, this criteria is insufficient, as the following example shows:

$$\frac{-2x+1}{2x+1} = \frac{2x-1}{-2x-1} = \frac{4x-2}{-4x-2} = \frac{-x+1/2}{x+1/2} = \frac{x-1/2}{-x-1/2}$$

To resolve these ambiguities, most existing systems take into account the following rules:

- No rational coefficient in expression (which eliminates the last two expressions);
- No integer may divide both the numerator and denominator (which eliminates the expression in the middle);
- The leading coefficient of the denominator must be positive (which eliminates the second).

There are several other possibilities, but these rules are the ones most used and are sufficient to give a canonical form.