Some Introductory Remarks on Computer Algebra

Wolfram Decker

Abstract. Computer algebra is a relatively young but rapidly growing field. In this introductory note to the mini-symposium on computer algebra organized as part of the third European Congress of Mathematics I will not even attempt to adress all major streams of research and the many applications of computer algebra. I will concentrate on a few aspects, mostly from a mathematical point of view, and I will discuss a few typical applications in mathematics. I will present a couple of examples which underline the fact that computer algebra systems provide easy access to powerful computing tools. And, I will quote from and refer to a couple of survey papers, textbooks and web-pages which I recommend for further reading.

1. Some Historical Remarks

Most of mathematics is concerned at some level with setting up and solving equations, for example to model applications in science and engineering. In many cases this involves tedious computations which are difficult to get right or too extensive to be carried through by hand. Numerical analysis and, more recently, computer algebra originated from this problem. That mechanized computing is not restricted to numerical computation was already evident to Charles Babbage and Lady Ada Augusta in the 19th century (see [63] for the following quotations and for more on the story of these pioneers of computer science). Besides working on his Difference Engines (see [94]) Babbage spent many years of his life with designing a more universal mechanical machine (Analytical Engine) to be programmed with punch-cards invented by J.-M. Jacquard for the control of his automatic looms. Babbage's earliest ideas on how such a machine could also perform mechanized algebraic manipulations are documented in his notebook of July 1836 [5]:

> "This day I had for the first time a general but very indistinct conception of the possibility of making an engine work out algebraic developments —I mean without any reference to the value of the letters. My notion is that as the cards (Ja[c]quards) of the calc. engine direct a series of operations and then recommence with the first, so it might perhaps be possible to cause the same cards to punch others equivalent to any given

> number of repetitions. But these hole[s] might perhaps be small pieces of formulae previously made by the first cards. . . "

In Lady Ada's extensive notes accompanying her translation from French into English of L. F. Menabrea's 1842 paper on the Analytical Engine [70] we find, for example, the following passage:

> "There are many ways in which it may be desired in special cases to distribute and keep separate the numerical values of different parts of an algebraic formula; and the power of effecting such distributions to any extent is essential to the algebraic character of the Analytical Engine. Many persons who are not conversant with mathematical studies imagine that because the business of the engine is to give its results in numerical notation, the nature of its process must consequently be arithmetical rather than algebraic and analytical. This is an error. The engine can arrange and combine its numerical quantities exactly as if they were letters or any other general symbols; and, in fact, it might bring out its results in algebraic notation were provisions made accordingly. It might develop three sets of results simultaneously, viz. symbolic results. . . ; numerical results (its chief and primary object); and algebraic results in literal notation."

The Analytical Engine, however, was never built, and symbolic computations were not carried through in an automated way until electronic computers were available. The first documented computer algebra programs, written for analytic differentiation, were described in two 1953 master's theses [53, 73]. According to the historical account by K. Geddes, S. R. Czapor, and G. Labahn in their introductory textbook [42] the beginning 1960's saw increasing activities in the field, in particular, after the LISP language,

> "a major advancement on the road to languages for symbolic computation"

had been developed. It was the period between 1961 and 1971 in which

> "the field progressed from birth through adolescence to at least some level of maturity."

I would like to mark this progress by some celebrated algorithmic breakthroughs, by the first releases of some well-known LISP-based general purpose computer algebra systems (see [42, 99] for some important systems not mentioned here), and by the beginning organization of researchers in interest groups, most notably in the Special Interest Group on Symbolic and Algebraic Manipulation (SIGSAM) of the Association for Computing Machinery (ACM).

1965	B. Buchberger's algorithm for computing Gröbner bases [16, 17]
1966	SYMSAM'66. First SYMposium on Symbolic and Algebraic Manipulation by ACM [38]
1967, 1970	E. R. Berlekamp's algorithm for factoring univariate polynomials over finite fields [7, 8]
1967	First issue of the SIGSAM Bulletin

1968-1970	R. H. Risch's algorithm for the indefinite integration of elementary functions [81, 82, 83, 84]
1969	H. Zassenhaus' algorithm for factoring univariate polynomials over the integers [100]
1968, 1970	First releases of REDUCE respectively REDUCE2 (A. C. Hearn)
1970	First release of MACSYMA (J. Moses, W. Martin)
1971	First release of SCRATCHPAD (J. Griesmer, R. Jenks)
1971	SYMSAM'71 [76]

A good impression of the speed with which computer algebra has been evolving since its beginnings can be obtained by checking

- the home-pages of its main interest groups and its leading researchers,
- recent textbooks such as [40, 90] (with extensive bibliographies),
- the Journal of Symbolic Computation and other journals such as Mathematics of Computation and Applicable Algebra in Engineering, Communication and Computing, and
- the proceedings of the leading conferences of computer algebra.

SYMSAM'66 was followed by several North American and European conferences which have been amalgamated into the annual International Symposium on Symbolic and Algebraic Computation (ISSAC) since 1988. ISSAC2000, chaired by T. Recio, is the 25th edition in this series. There are various other series of conferences either being organized by continental, national or local interest groups, or gathering together users of one of the computer algebra systems, or being devoted to special features such as COCOA (COmputational COmmutative Algebra), DISCO (Design and Implementation of Symbolic Computation Systems), MEGA (Effective Methods in Algebraic Geometry), or PASCO (PArallel Symbolic COmputation). See [99] for a much larger list.

2. Some Features of Computer Algebra

Calculations in computer algebra are carried through exactly, that is, no approximation is applied at any step. Infinite precision arithmetic allows to actually compute in the *ring* of integers and in the *field* of rationals, in finite prime fields, algebraic number fields, and in arbitrary Galois fields. In fact, there is a much larger variety of algebraic structures in which algebraic algorithms allow to manipulate algebraic objects or the structures itself. And, exact computer algebra methods allow to create algorithms that decide, for example, the solvability of systems of polynomial equations or the solvability of indefinite summation and integration problems in certain specified classes of functions. Let us look at a few examples and introduce some computer algebra systems as well (the home-pages of these and other systems can be easily found on the net).

Example 2.1. *GAP (Groups, Algorithms and Programming), until recently developed under the direction of J. Neubüser, is a system "for computational discrete*

algebra with particular emphasis on computational group theory." In the following GAP4 session we define the Heisenberg group H_3 of level 3 in its Schrödinger representation, check that H_3 is a finite group of order 27, and compute its character table. Here Schrödinger representation means that $H_3 \subset \mathrm{GL}(3, \mathbf{Q}(\xi))$, $\xi = e^{\frac{2\pi i}{3}}$ a primitive 3rd root of unity, is given as the matrix group generated by

$$\sigma = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \textit{and} \quad \tau = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \xi & 0 \\ 0 & 0 & \xi^2 \end{pmatrix}.$$

```
gap> m1:=[[0,0,1],[1,0,0],[0,1,0]];;
gap> m2:=[[1,0,0],[0,E(3),0],[0,0,E(3)^2]];;
gap> G:=Group(m1,m2);;
gap> Size(G);
27
gap> Display(CharacterTable(G));
CT1

      3  3  2  2  2  2  2  2  2  2  3  3

        1a 3a 3b 3c 3d 3e 3f 3g 3h 3i 3j

X.1      1  1  1  1  1  1  1  1  1  1  1
X.2      1  A  A  A /A /A /A  1  1  1  1
X.3      1 /A /A /A  A  A  A  1  1  1  1
X.4      1  1 /A  A  1 /A  A /A  A  1  1
X.5      1  A  1 /A /A  A  1 /A  A  1  1
X.6      1 /A  A  1  A  1 /A /A  A  1  1
X.7      1  1  A /A  1  A /A  A /A  1  1
X.8      1  A /A  1 /A  1  A  A /A  1  1
X.9      1 /A  1  A  A /A  1  A /A  1  1
X.10     3  .  .  .  .  .  .  .  .  B /B
X.11     3  .  .  .  .  .  .  .  . /B  B

A = E(3)
  = (-1+ER(-3))/2 = b3
B = 3*E(3)^2
  = (-3-3*ER(-3))/2 = -3-3b3
```

Via the online GAP user manual we easily find out how to read the output.

Example 2.2. *MAGMA, developed under the direction of J. Cannon, is a computer algebra system "for Algebra, Number Theory and Geometry". In the following MAGMA session we define again H_3 in its Schrödinger representation. Then H_3, as a subgroup of $\mathrm{GL}(3, \mathbf{Q}(\xi))$, acts on the polynomial ring $\mathbf{Q}(\xi)[x_1, x_2, x_3]$ by linear*

substitution. All polynomials which are invariant under this action form the invariant ring $\mathbf{Q}(\xi)[x_1, x_2, x_3]^{H_3}$. *We compute a corresponding fundamental system of invariants, that is, a minimal, finite set of homogeneous invariants generating* $\mathbf{Q}(\xi)[x_1, x_2, x_3]^{H_3}$ *as a ring.*

```
> K<g>:=NumberField(x^2+x+1);
> m1:=[0,0,1,1,0,0,0,1,0];
> m2:=[1,0,0,0,g,0,0,0,g^2];
> G:=MatrixGroup<3,K|m1,m2>;
> R:=InvariantRing(G);
> FundamentalInvariants(R);

[

    x1^3 + x2^3 + x3^3,

    x1*x2*x3,

    x1^6 + x2^6 + x3^6,

    x1^6*x3^3 + x1^3*x2^6 + x2^3*x3^6

]
```

Example 2.3. *NTL, by V. Shoup, "is a high-performance, portable C++ library providing data structures and algorithms for manipulating signed, arbitrary length integers, and for vectors, matrices, and polynomials over the integers and over finite fields." The following is taken from the online Tour of NTL.*

"NTL provides extensive support for very fast polynomial arithmetic. In fact, this was the main motivation for creating NTL in the first place, because existing computer algebra systems and software libraries had very slow polynomial arithmetic. The class ZZX represents univariate polynomials with integer coefficients. The following program reads a polynomial, factors it, and prints the factorization.

```
#include <NTL/ZZXFactoring.h>
main() {
   ZZX f;

   cin >> f;

   vec_pair_ZZX_long factors;
   ZZ c;

   factor(c, factors, f);
```

```
   cout << c << "\n";
   cout << factors << "\n";
}
```

When this program is compiled and run on input

```
   [2 10 14 6]
```

which represents the polynomial $2 + 10 * X + 14 * x^2 + 6 * X^3$, the output is

```
   2
   [[[1 3] 1] [[1 1] 2]]
```

The first line of output is the content of the polynomial, which is 2 in this case as each coefficient of the input polynomial is divisible by 2. The second line is a vector of pairs, the first member of each pair is an irreducible factor of the input, and the second is the exponent to which is appears in the factorization. Thus, all of the above simply means that

```
   2 + 10*X + 14*x^2 +6*X^3 =2 * (1 + 3*X) * (1 + X)^2
```

Admittedly, I/O in NTL is not exactly user friendly, but then NTL has no pretensions about being an interactive computer algebra system: it is a library for programmers."

Example 2.4. *MAPLE, founded by G. Gonnet and K. Geddes and firstly released in 1983, is a general purpose system with a kernel written in C, and with most higher level functions or packages written in the MAPLE user language. In the following MAPLE session we factor a multivariate polynomial over the integers, namely*

$$p = x^2y^4z - xy^9z^2 + xyz^3 + 2x - y^6z^4 - 2y^5z \in \mathbf{Z}[x, y, z]$$

```
> p:=x^2*y^4*z - x*y^9*z^2 + x*y*z^3 + 2*x - y^6*z^4 - 2*y^5*z:
> factor(p);
                          4          3           5
                       (y  z x + y z  + 2) (x - y  z)
```

Example 2.5. *SCHUBERT, written by S. Katz and S. A. Strømme, is a "MAPLE package for Intersection Theory." Intersection rings of algebraic varieties are represented by generators and relations, that is, as residue class rings of polynomial rings. The following two SCHUBERT sessions, taken from the SCHUBERT manual, compute the number of lines respectively conics on a general quintic hypersurface in complex projective* 4*-space (see also* [58]*).*

```
# Lines on a quintic threefold.  This is the top Chern class of the
# 5th symmetric power of the universal quotient bundle on the
# Grassmannian of lines.
>
> grass(2,5,c):        # Lines in P^4.
```

```
> B:=symm(5,Qc):       # Qc is the rank 2 quotient bundle, B its
>                      # 5th symmetric power.
> c6:=chern(rank(B),B):    # the 6th Chern class of this
                           # rank 6 bundle.
> integral(c6);
                                2875

#-------------------------------------------------------------------

# Conics on a quintic threefold. This is the top Chern class of the
# quotient of the 5th symmetric power of the universal quotient on
# the Grassmannian of 2 planes in P^5 by the subbundle of quintics
# containing the tautological conic over the moduli space of
# conics.
>
> grass(3,5,c):         # 2-planes in P^4.
> B:=Symm(2,Qc):        # The bundle of conics in the 2-plane.
> Proj(X,dual(B),z):    # X is the projective bundle of all conics.
> A:=Symm(5,Qc)-Symm(3,Qc)&*o(-z):     # The rank 11 bundle of
>                                      # quintics restricted to the
                                       # universal conic.
> c11:=chern(rank(A),A):  # its top Chern class.
> lowerstar(X,c11):       # push down to G(3,5).
> integral(Gc,");         # and integrate there.
                              609250
```

Example 2.6. *SINGULAR, developed under the direction of G.-M. Greuel, G. Pfister, and H. Schönemann, "is a Computer Algebra System for polynomial computations with special emphasize on the needs of commutative algebra, algebraic geometry, and singularity theory." In the following SINGULAR session we check that the system of polynomial equations*

$$-x^2+xy-xz+y+1=x^2+xy-y^2-2=-x^3-x^2y+xyz+xz^2-xy-y=$$
$$=-x^3+xy^2+x^2z+xyz+xz^2-z^2=0$$

has no solution over the complex numbers. Indeed, by Hilbert's Nullstellensatz, it suffices to compute that the reduced Gröbner basis of the ideal $I \subset \mathbf{Q}[x,y,z]$ *defined by the polynomials is* 1.

```
> ring r=0,(x,y,z),dp;
> ideal i1=-x2+xy-xz+y+1, x2+xy-y2-2;
> ideal i2=-x3-x2y+xyz+xz2-xy-y, -x3+xy2+x2z+xyz+xz2-z2;
> ideal i=i1+i2;
> std(i);
_[1]=1
```

For the very simple examples considered so far "almost no" computing time is needed. Just a little bit more involved is the following example.

Example 2.7. *LIDIA, developed under the direction of J. Buchmann, is "a C++ Library for Computational Number Theory". In the following LIDIA session we factor the 8th Fermat number* $F_8 = 2^{2^8} + 1$.

```
lc> factor(2^(2^8)+1);
$0 =  [(1238926361552897,1)
(93461639715357977769163558199606896584051237541638188580280321,1)]
>
```

We will discuss further examples later-on. Now, let us again quote Geddes, Czapor, and Labahn [42]:

> "There are three recognizable, yet interdependent, forces in the development of this field. We may classify them under the headings *systems*, *algorithms*, and *applications*."

People are concerned with developing (algebraic) algorithms and analyzing the complexity of algorithms, they provide the necessary surroundings for implementing algorithms and for allowing users to work with them, they implement algorithms and test their practical performance, and they apply algorithms. A more detailed impression is obtained by looking at the home-page of ISSAC2000 (which can be easily found on the net).

> "The conference topics include, but are not limited to:
> *Algorithmic mathematics*: Algebraic, symbolic, and symbolic-numeric algorithms including: simplification, polynomial and rational function manipulations, algebraic equations, summation and recurrence equations, integration and differential equations, linear algebra, number theory, group computations, and geometric computing.
> *Computer science*: Theoretical and practical problems in symbolic mathematical manipulation including: computer algebra systems, data structures, computational complexity, problem solving environments, programming languages and libraries for symbolic-numeric-geometric computation, user interfaces, visualization, software architectures, parallel or distributed computing, mapping algorithms to architectures, analysis and benchmarking, automatic differentiation and code generation, automatic theorem proving, mathematical data exchange protocols.
> *Applications*: Problem treatments incorporating algebraic, symbolic, symbolic-numeric and geometric computation in an essential or novel way, including engineering, economics and finance, architecture, physical and biological sciences, computer science, logic, mathematics, statistics, and use in education."

Let me add a couple of observations:.

- The different branches of computer algebra are cross-linked in numerous ways.
- There are increasing efforts of bringing the symbolic and numerical communities together.
- There is an increasing use of computer algebra systems in education.

I would like to illustrate these observations by giving some examples. J. Cannon and D. Holt are the guest editors of the Special issue on computational algebra and number theory: Proceedings of the first MAGMA conference of the Journal of Symbolic Computation. Let me quote from their foreword [20]:

> "... even if one wanted to, it would no longer be possible to treat a particular area of computational algebra, such as computational group theory, as an isolated and self-sufficient branch of mathematics. As the various different branches of computational algebra mature they are seen to rely on a common set of fundamental tools. To take a simple example, if we wish to compute chief factors of a group G, we may perhaps quickly find elementary abelian sections M/N of G (where M and N are normal in G). But we then need to refine the section, and to do this, we need to consider M/N as a module for G over a finite field, and to find a composite series of the module. Currently, the best method known of achieving this involves factorizing polynomials over finite fields, which is drawing us much closer to the realms of traditional symbolic computation. Another example concerns techniques for the efficient computation of Hermite and Smith normal forms for integral matrices (and the LLL algorithm) which are key tools in both computational group theory and algebraic number theory."

H. M. Möller [72] gives several examples of how to use Gröbner bases in numerical analysis. In his introduction we also find the following general remarks:

> " In the nineties, there is an increasing interest in combining symbolic and numerical methods. This can be seen at diverse instances. There are now international symposia supported by organizations from both sides, and the number of contributes displaying the symbolic - numerical interplay is increasing. Other examples are the facilities of using floating point arithmetics and simple numerical procedures in Computer Algebra Systems on the one hand and the (eventually partly) integration of Computer Algebra Systems into numerical packages on the other hand. The most prominent example is here the migration of the Computer Algebra System Axiom to NAG, the Numerical Algorithm Group.
>
> Many interesting results have been obtained by combining symbolic and numerical methods, like in polynomial continuation the avoiding of solution paths diverging to infinity by means of concepts from toric ideals or like the numerical solving of systems of polynomial equations using resultants, see for instance Canny and Manocha (1993) [21]."

For more information in this direction I refer to the articles by G. Gonnet [43] and G.-M. Greuel [48] in these proceedings, to the Special Issues of the Journal of Symbolic Computation on Validated numerical methods and computer algebra [59] respectively Symbolic numeric algebra for polynomials [98], and to the homepage of CASC2000, the Third International Workshop on Computer Algebra in Scientific Computing.

The Special Issue [62] of the Journal of Symbolic Computation presents a couple of articles which deal with the use of computer algebra in courses on group theory, abstract algebra, computational non-associative algebra, differential equations and interpolation. The paper by B. Amrhein, O. Gloor and R. E. Maeder [2] in this special issue is concerned with Visualizations for mathematics courses based on a computer algebra system. Let me quote from the introduction:

> "Computer-algebra systems (CAS) provide the necessary algorithms needed to compute mathematics visualizations. Furthermore, in the last years, the graphics capabilities of computer-algebra systems have improved considerably and now satisfy the needs of visualization in education.
>
> CAS also give teachers and students another and more direct approach to using the computer. Applying a CAS, much less effort is needed to treat a simple practical problem than with the classical approach, learning a full programming language first. Therefore, the focus moves from computer handling to the application. This enables the possibility of applying the computer in education not as a teaching object but as a tool to solve problems in other disciplines. In addition, CAS became of increasing importance and are widely used in the industry. Hence, an introduction to CAS will soon belong to a modern curriculum."

There are in fact numerous textbooks and tutorials on using computer algebra in high-school and university courses.

Example 2.8. *The book by D. Cox, J. Little, and D. O'Shea* [29] *gives an introduction into both Gröbner bases and algebraic geometry, starting from scratch. This approach allows to work out first examples in algebraic geometry without developing too much of the abstract theory behind algebraic geometry, that is, it allows to introduce students into algebraic geometry at a very early stage of their studies. On the other hand, the geometric point of view nicely motivates the need for a variety of algorithms manipulating ideals in polynomial rings.*

3. Some Remarks on Systems and Algorithms

One reason for the great success of computer algebra is, as indicated several times above, that modern computer algebra systems (CAS) provide easy access to powerful computing tools. Nowadays there is a large variety of systems suiting different needs. In addition, a modern CAS comes with a programming language which allows to extend the system (for some systems even the source code is available). Some of these extensions are publicly available, and, typically, a CAS which has

been around for a little while is supplemented by a variety of user-written packages or libraries such as the MAPLE package SCHUBERT introduced in example 2.5.

There are general purpose and special purpose CAS. Typically, one has to pay for a general purpose system whereas many of the special purpose systems are for free. From a general purpose system we expect that it provides tools for *symbolic computations*, for *numeric computations*, and for *visualization*, and we expect that there is a large variety of such tools allowing to attack many different problems. Information on such systems can be found on the net and in numerous introductory textbooks. The practical guide [99] contains articles which compare the capabilities of systems such as REDUCE, MACSYMA, MAPLE, DERIVE, MATHEMATICA, MUPAD, AXIOM (the successor of SCRATCHPAD) and TI-92.

For many of the more special and advanced applications general purpose systems are not powerful enough. Therefore a researcher with a desperate need for computational power in the context of certain problems actually may decide to create his own system. Some of todays' special purpose systems started out in this way. A pioneering and prominent example is SCHOONSCHIP by M. J. G. Veltman which helped to win a Nobel price in physics in 1999 (awarded to Veltman and G. t'Hooft "for having placed particle physics theory on a firmer mathematical foundation", see http://nobelprizes.com/nobel/physics/physics.html). From a special purpose system we expect highly tuned implementations of the algorithms needed for the area in which the system is specializing. As pointed out in the last section this includes special algorithms such as those for computing fundamental systems of invariants in example 2, and it includes algorithms which are common ground to every system such as algorithms for polynomial arithmetic. Again, information on such systems can be found on the net and in recent textbooks.

Most people who want to apply computer algebra are actually not interested in computer algebra itself. They will find "their" system(s) already on the market and they are going to use such a system as a kind of a black box machine without being forced to understand details of the algorithms or of their implementations. In some cases, however, information on which variant of an algorithm is implemented is necessary for understanding the meaning of the output (see the discussion of Risch's algorithm below). Unfortunately, detailed information is often difficult or even impossible to find. Setting a flag might allow to obtain at least partial information.

Example 3.1. *Let us compute*

$$\sum_{0\le k\le m} (-1)^k \binom{n}{k} = (-1)^m \binom{n-1}{m}$$

with MAPLE V, Release 5.1.

```
> infolevel[sum]:=3:
> sum((-1)^k*binomial(n,k), k=0..m);
```

```
sum/indefnew:   indefinite summation
sum/extgosper:   applying Gosper algorithm to
                 a(   k   ):=   (-1)^k*binomial(n,k)
sum/gospernew:   a(   k   )/a(   k   -1):=    (-n+k-1)/k
sum/gospernew:   Gosper's algorithm applicable
sum/gospernew:   p:=   1
sum/gospernew:   q:=   -n+k-1
sum/gospernew:   r:=   k
sum/gospernew:   degreebound:=   0
sum/gospernew:   solving equations to find f
sum/gospernew:   Gosper's algorithm successful
sum/gospernew:   f:=   -1/n
sum/indefnew:    indefinite summation finished
```

$$-\frac{(m+1)^{(m+1)}\,(-1)^{(m+1)}\,\text{binomial}(n,\,m+1)}{n}$$

Here we understand that there is an algorithm due to Gosper which does the job. Indeed, R. W. Gosper's algorithm [46] solves the indefinite summation problem for hypergeometric functions. This problem and Gosper's solution to it can be formulated rigorously by using difference fields (see, for example, [40]).

Differential fields provide the algebraic setting for indefinite symbolic integration. Symbolic integration has a long history starting with I. Newton and G. W. Leibniz, including work by J. Bernoulli in the 18th century, by J. Liouville, M. W. Ostrogradsky and E. Hermite in the 19th century, by J. F. Ritt in the middle of the 20th century, and stunning modern work subsumed under the name Risch's algorithm (see M. F. Singer's survey article [91] and M. Bronstein's book [14] for details and references). As is Gosper's algorithm for the summation of hypergeometric functions, Risch's algorithm is a decision algorithm for integrating elementary functions. Given such a function f, the algorithm computes an elementary function which is an anti-derivative of f or decides that no such anti-derivative exists. An algorithm for the case of transcendental elementary functions was given by R. H. Risch in [83]. Subsequent improvements are due to many people (see again [91]). An algorithm for the general case, which is much more involved than the algorithm for transcendental functions, was outlined by Risch in [81, 82, 84]. New ideas and improvements in the case of algebraic elementary functions are due to J. H. Davenport [30] and B. Trager [97]. M. Bronstein [13] generalized particularly Trager's ideas to give an algorithm for the general case. He implemented a great part of this algorithm in AXIOM, but to the best of my knowledge, no complete implementation exists (I refer to [13] and [24] for a discussion of some of the problems in this direction). As a consequence, if a CAS can't integrate a given

elementary function, this might mean that no elementary anti-derivative exists, or that the variant of Risch's algorithm which is implemented can't handle the particular case (but this might not be evident for the user).

Example 3.2. *Bronstein's ISSAC'98 tutorial* [15] *outlines Risch's algorithm for various classes of elementary functions and gives corresponding examples such as*

$$\int \frac{(x^2+2x+1)\sqrt{x+\log x}\ +(3x+1)\log x\ +3x^2+x}{(x^2+x\log x)\sqrt{x+\log x}\ +x^2\log x\ +x^3}\ dx$$
$$= 2\sqrt{x+\log x}\ +2\log(x+\sqrt{x+\log x}\,)$$

in the algebraic logarithmic case. From among the systems which I tested with this example only AXIOM 2.2 gave an honest answer:

```
>> Error detected within library code:
integrate: implementation incomplete (constant residues)
```

The interested reader is invited to test his favorite CAS more systematically.

A central area of computer algebra, whose history also started in classical papers, and which saw dramatic modern developments with numerous research papers, is polynomial factorization.

There is a classical algorithm for factoring univariate polynomials over the integers due to F. T. von Schubert [89] who generalized ideas of I. Newton. This algorithm was rediscovered by L. Kronecker [60] and implemented in early computer algebra programs. Modern, highly practical algorithms rely on ideas of E. R. Berlekamp [7, 8] and D. G. Cantor and H. Zassenhaus [22] for factoring univariate polynomials over finite fields, and of Zassenhaus [100], who used Hensel-lifting for rediscovering the factors of a (squarefree and primitive) polynomial $f \in \mathbf{Z}[x]$ from those of f reduced modulo a suitable prime p. During the last three decades enormous progress has been made, with new ideas and improvements of the basic algorithms. Now one can handle problems of a size which was inconceivable a couple of years ago.

Example 3.3. *P. Roelse* [86] *reports on an implementation of Niederreiter's algorithm for factoring univariate polynomials over* $\mathbf{F}_2$*. With this implementation he is able to factor a pseudo-randomly chosen polynomial of degree* 300 000*.*

Factoring univariate integer polynomials as described above works well "on the average example in praxis". It has, however, exponential running time in the worst case (see, for example, [40]). A polynomial-time algorithm was given by A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovász [66] who replaced the factor combination step at the very end of Zassenhaus' algorithm by a step involving their celebrated LLL-algorithm for lattice basis reduction. The LLL-algorithm together with its more recent variants has many applications, for example, to integer programming (see H. W. Lenstra's paper in these proceedings [68]).

Kronecker [60] also showed how to reduce multivariate factorization to the univariate case. Again, there are more practical, modern algorithms (for different representations of the multivariate polynomials). A polynomial-time reduction from dense multivariate to univariate integer polynomial factorization is due to E. Kaltofen [54]. Kaltofen is also one of the protagonists of the further development. I refer to his three survey papers [55, 56, 57] for the history of univariate and multivariate polynomial factorization over various coefficient domains until 1991. Further remarks can be found in [40, 90] (according to the notes on univariate factorization in [40] some of the basic ideas of the modern algorithms can actually be traced back to classical work, most notably to work of Gauss).

4. Some Remarks on Applications and Algorithms

Some of the fields in which computer algebra has applications are mentioned above (see also the home-page of ACA2000, the 6th conference on Applications of Computer Algebra by IMACS (International Association for Mathematics and Computers in Simulation)). Some examples for industrial applications are given in the paper by L. Gonzalez-Vega and T. Recio in these proceedings [44]. I will not comment further on applications outside of mathematics. Instead I will illustrate the impact of computer algebra on mathematical research by giving some examples, mostly from group theory, number theory, and algebraic geometry. Needless to say that there are applications in other areas of mathematics as well. Exact computer algebra methods may help

- to correct and supplement old mathematical tables or to create new tables (databases),
- to revive classical problems,
- to solve enumerative problems,
- to verify theorems whose proof has been reduced to straightforward but tedious calculations,
- to construct interesting examples such as counterexamples to conjectures,
- to find mathematical evidence by experiments.

Example 4.1. *The GAP Data Library contains "a number of databases with GAP language interfaces allowing them to be searched and studied efficiently", see http://www-history.mcs.st-and.ac.uk/∼gap/. Among others one can find, "in various databases, all groups of order up to* $1,000$*, excluding* 512 *and* $768''$ *(see the papers by H. U. Besche and B. Eick* [9, 10]*, who also announced a solution for the order 768 case). The case of order 512 is dealt with in* [35]*. Whereas all groups of order* 512 *with p-class at least three can be explicitly generated, the computations in the case of p-class* 2 *are extremely large. Therefore the authors decided to only enumerate these groups. In fact, they showed that there are* $8\ 785\ 772$ *such groups. Altogether there exist* $10\ 494\ 213$ *groups of order* 512*.*

Example 4.2. *In a tour de force of enumerative geometry G. Ellingsrud and S.-A. Strømme* [37] *computed that the number of twisted cubic curves on a general quintic hypersurface in complex projective* 4*-space is* 317 206 375 *(this was one of the motivations for creating the package SCHUBERT). The number* 317 206 375 *had been predicted by theoretical physicists* [19] *with a method originating from string theory (this method allows, in fact, to predict the number of rational curves on a general quintic of any degree).*

Example 4.3. *The Number Theory Website, maintained by Keith Matthews at http://www.maths.uq.edu.au/∼krm/web.html, provides plenty of information on number theory including links to the home-pages of CAS and to tables and databases (numbers, elliptic curves, number fields, modular forms). In particular, there is a link to the Primes Pages by C. K. Caldwell with tables such as Mersenne primes, or largest primes, and with recent records such as*

> *"On 1 June 1999, the team of Nayan Hajratwala, George Woltman, Scott Kurowski et. al. discovered a new record prime:* $2^{6972593} - 1$. *This is the 38th known Mersenne prime..."*

(see http://www.utm.edu/research/primes/index.html). On the page on Fermat numbers (see http://vamri.xray.ufl.edu/proths/fermat.html) compiled by W. Keller one can find, for example, the following:

> *"On May 9, 2000, Ray Ballinger discovered a new factor of a Fermat number using Yves Gallot's program Proth.exe:* $591909 \cdot 22063 + 1$ *divides* F_{2059}. *This is the eighth factor found with Gallot's program."*

H. W. Lenstra, Jr. and A. K. Lenstra [64] give a theoretical and a practical reason for the renewed interest in primality testing and factoring integers in recent years: the introduction of complexity theory, which

> "enabled workers in the field to phrase the fruits of their intellectual labors in terms of theorems that apply to more than a finite number of cases",

and

> "the discovery, by Rivest, Shamir and Adleman [85], that the difficulty of factorization can be applied for cryptographical purposes ... we note that for the construction of the cryptographic scheme that they proposed it is important that primality testing is easy, and that for the unbreakability of the scheme it is essential that factorization is hard. Thus, as far as factorization is concerned, this is a *negative* application..."

We refer to [64] for a discussion of modern algorithms for primality testing and factoring including, for factoring, the quadratic sieve by C. Pomerance [77, 78] and the elliptic curve method by H. W. Lenstra, Jr. [67]. Various articles on the number field sieve can be found in [65]. An introduction to computational number theory in general is given in H. Cohen's books [25, 26]. These books also contain a short discussion of some of the CAS suitable for number theory including MAGMA, PARI-GP, developed under the direction of H. Cohen, KANT/KASH,

developed under the direction of M. Pohst, LIDIA, and SIMATH, developed under the direction of H. G. Zimmer.

Classical invariant theory (see [71] and [47]) is a 19th century field involving tedious calculations (see, for example, the many pages of extensive explicit tables in A. Cayley's collected works [23]). Invariants such as the discriminant $b^2 - ac$ of a quadratic binary form $ax^2 + 2bxy + cy^2$ come into play when one asks for properties of sets of solutions of polynomial equations which are invariant under certain classes of transformations.

Example 4.4. *J. J. Sylvester* [95, 96] *enumerated the fundamental invariants (more generally covariants) of binary forms of some low degrees. Together with Franklin* [39] *he arrived at his figures by manipulating the generating functions of the rings of covariants under consideration following ideas of Cayley and Sylvester. In fact, this approach yields for each given degree only a lower bound on the number of fundamental covariants. In order to conclude that this lower bound is actually the correct number a*

> *"... fundamental postulate still awaiting demonstration is necessary.... The validity of the fundamental postulate... is verified by its conducting to results which have been proved to be accurate for single binary quantics up to sixth order inclusive..."*

Unfortunately, the fundamental postulate proved to be wrong in the next case, that is, in degree 7 [49]. *That there are indeed four more fundamental invariants in this degree than those enumerated by Sylvester was computed by Dixmier and Lazard* [34] *in 1986 with the help of computer algebra. The correct number of fundamental covariants, however, seems to be still unknown. Whereas nowadays every reader can easily recheck Sylvester's manipulations with his favorite computer algebra system, the contemporaries of Sylvester had to travel to Baltimore:*

> *"The manuscript sheets containing the original calculations... are deposited in the iron safe of the John Hopkins University Baltimore, where they can be seen and examined..."*

Classical invariant theory culminated in two papers of Hilbert [50, 51] which contain several "lemmas" which deeply influenced the development of modern abstract algebra and algebraic geometry. The first paper, for example, contains the basis theorem, the syzygy theorem and the theorem on the structure of generating functions (Hilbert functions). In our context these results play a crucial role in the theory of Gröbner bases. An interesting side remark is that the concepts of monomial orders, normal-form reductions and Gröbner bases were already introduced in 1899 by P. Gordan [45] in order to give another proof of Hilbert's basis theorem. Gordan's paper does not contain, however, Buchberger's celebrated algorithm for computing Gröbner bases [16, 17]. This algorithm contributes as one of its numerous applications (see [18]) to nowadays' revival of computational invariant theory (see [93, 32]) which in turn has many new applications ranging from topology

and geometry, to physics, continuum mechanics, and computer vision (see also [75, 41, 92]). Let us quote J. P. S. Kung and G.-C. Rota [61]:

> "Like the Arabian phoenix rising out of its ashes, the theory of invariants, pronounced dead at the turn of the century, is once again at the forefront of mathematics. During its long eclipse, the language of modern algebra was developed, a sharp tool now at last being applied to the very purpose for which it was invented."

Let me quickly mention some further examples. A classical example of finishing a proof with the help of computers is the solution of the four-color problem by K. Appel and W. Haken [3]. The celebrated conjectures of B. J. Birch and H. P. F. Swinnerton-Dyer [11, 12] are based on extensive computer calculations. The same holds for the heuristics on class groups by H. Cohen and H. W. Lenstra, Jr. [27, 28]. An example from algebraic geometry is the systematic treatment by D. Eisenbud and S. Popescu [36] of some counterexamples to the minimal resolution conjecture [69]. The authors are grateful to D. Bayer and M. Stillman respectively D. Grayson and M. Stillman for the systems MACAULAY respectively MACAULAY2

> "which have been extremely useful for us; without them we would probably never have been bold enough to guess the existence of the structure that we explain here."

The first counterexamples to the minimal resolution conjecture had been found by F.-O. Schreyer (unpublished, compare [52]) with a sophisticated random search using MACAULAY. Schreyer's approach also yielded some interesting examples of projective varieties of low codimension [87, 88]. The construction method of W. Decker, L. Ein and F.-O. Schreyer combines syzygy theory, the theorem of Beilinson [6] and various computational techniques to construct smooth surfaces in projective 4-space and to find out where these surfaces stand in the Enriques-Kodaira classification (see [33, 31, 4, 80, 79, 1]). These authors also used MACAULAY. For some applications in algebraic geometry and singularity theory obtained with the help of the more recent system SINGULAR we refer to G.-M. Greuel's article in these proceedings [48].

Finally, let me come back to number theory and mention the indirect disproof of Merten's conjecture by A. M. Odlyzko and H. J. J. Riele [74] for which the LLL-algorithm

> "was the main new ingredient that allowed us to obtain much stronger results than those of previous authors."

References

[1] H. Abo, W. Decker, N. Sasakura, *An elliptic conic bundle in* $\mathbf{P}^4$ *arising from a stable rank-3 vector bundle,* Math. Z. **229** (1998), 725–741.

[2] B. Amrhein, O. Gloor, R. E. Maeder, *Visualizations for mathematics based on a computer algebra system,* J. Symbolic Computation **23** (1997), 447–452.

[3] K. Appel, W. Haken, *Every planar map is four colorable,* AMS, Providence, 1989.

[4] A. Aure, W. Decker, K. Hulek, S. Popescu, K. Ranestad, *Syzygies of abelian and bielliptic surfaces in* $\mathbf{P}^4$, International J. Math. **8** (1997), 849–919.

[5] C. Babbage, *Scribbling books*, volume 2 (1836), Science Museum Library, London.

[6] A. Beilinson, *Coherent sheaves on* $\mathbf{P}^N$ *and problems of linear algebra,* Funkt. Anal. Appl. **12** (1978), 214–216.

[7] E. R. Berlekamp, *Factoring polynomials over finite fields,* Bell System Tech. J. **46** (1967), 1853–1859.

[8] E. R. Berlekamp, *Factoring polynomials over large finite fields,* Math. Comp. **24** (1970), 713–735.

[9] H. U. Besche, B. Eick, *Construction of finite groups,* J. Symbolic Computation **27** (1999), 387–404.

[10] H. U. Besche, B. Eick, *The groups of order at most 1000 except 512 and 768,* J. Symbolic Computation **27** (1999), 405–413.

[11] B. J. Birch, H. P. F. Swinnerton-Dyer, *Notes on elliptic curves. I,* J. Reine Angew. Math. **212** (1963), 7–25.

[12] B. J. Birch, H. P. F. Swinnerton-Dyer, *Notes on elliptic curves. II,* J. Reine Angew. Math. **218** (1965), 79–108.

[13] M. Bronstein, *Integration of elementary functions,* J. Symbolic Computation **9** (1990), 117–173.

[14] M. Bronstein, *Symbolic integration I – transcendental functions,* Springer, Berlin, 1997.

[15] M. Bronstein, *Symbolic integration tutorial, ISSAC'98,* downloadable from http://www-sop.inria.fr/cafe/Manuel.Bronstein/bronstein-eng.html

[16] B. Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal,* PdH thesis, Lepold-Franzens-Universität, Innsbruck, 1965.

[17] B. Buchberger, *Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems,* Aequationes mathematicae **4** (1970), 374–383, English translation by M. Ambramson and R. Lumbert in [18], 535–545.

[18] B. Buchberger, F. Winkler (eds.), *Gröbner bases and applications, Linz 1999,* Cambridge University Press, Cambridge, 1999.

[19] P. Candelas, X. C. de la Ossa, P. S. Green, L. Parkes, *A pair of Calabi-Yau manifolds as an exactly soluble superconformal theory,* Nuclear Phys. B. **359** (1991), 21–74.

[20] J. Cannon, D. Holt, Foreword of the guest editors to the Special issue on computational algebra and number theory: Proceedings of the first MAGMA conference, J. Symbolic Computation **24** (1997), 233–234.

[21] J. F. Canny, D. Manocha, Multipolynomial resultant algorithms, J. Symbolic Computation **15** (1997), 99–122.

[22] D. G. Cantor, H. Zassenhaus, *A new algorithm for factoring polynomials over finite fields,* Math. Comp. **36** (1981), 587–592.

[23] A. Cayley, *The collected mathematical papers,* vols 1–12, Cambridge University Press, Cambridge, 1889.

[24] A. M. Cohen, J. H. Davenport, J. P. Heck, *An overview of computer algebra,* in A.M. Cohen (ed.), Computer algebra in industry, Wiley, Chichester, 1993.

[25] H. Cohen, *A course in computational algebraic number theory (3rd corrected printing),* Springer, New York, 1996.

[26] H. Cohen, *Advanced topics in computational number theory,* Springer, New York, 2000.

[27] H. Cohen, H. W. Lenstra, Jr., *Heuristics on class groups,* in D.V. Chudnovsky et al. (eds.), *Number theory, New York 1982,* Springer, Berlin, 1984.

[28] H. Cohen, H. W. Lenstra, Jr., *Heuristics on class groups of number fields,* in H. Jager (ed.), *Number theory, Noordwijkerhout, 1983,* Springer, Berlin, 1984.

[29] D. Cox, J. Little, D. O'Shea, *Ideals, varieties, and algorithms, second edition,* Springer, New York, 1997.

[30] J. H. Davenport, *On the integration of algebraic functions,* Springer, New York, 1981.

[31] W. Decker, L. Ein, F.-O. Schreyer, *Construction of surfaces in* $\mathbf{P}^4$*,* J. Algebraic Geometry **2** (1993), 185–237.

[32] W. Decker, T. de Jong, *Gröbner bases and invariant theory,* in [18], 61–89.

[33] W. Decker, F.-O. Schreyer, *Non-general type surfaces in* $\mathbf{P}^4$*: some remarks on bounds and constructions,* J. Symbolic Computation **29** (2000), 545–582.

[34] J. Dixmier, D. Lazard, *Minimum number of fundamental invariants for the binary form of degree 7,* J. Symbolic Computation **6** (1988), 113–115.

[35] B. Eick, E. A. O'Brien, *The groups of order 512,* in B.H. Matzat et al. (eds.), *Algorithmic algebra and number theory,* 379–380, Springer, Berlin, 1999.

[36] D. Eisenbud, S. Popescu, *Gale duality and free resolutions of ideals of points,* Invent. Math. **136** (1999), 419–449.

[37] G. Ellingsrud, S.-A. Strømme, *The number of twisted cubic curves on the general quintic threefold,* Math. Scand. **76** (1995), 5–34.

[38] R. W. Floyd (ed.), *Proc. of ACM Symposium on Symbolic and Algebraic Manipulation (SYMSAM'66), Washington D.C.,* Comm. ACM **9** (1966), 574–643.

[39] F. Franklin, *On the calculation of the generating functions and tables of groundforms for binary quantics,* Amer. J. of Math. **3** (1883), 128–153.

[40] J. von zur Gathen, J. Gerhard, *Modern computer algebra,* Cambridge University Press, Cambridge, 1999.

[41] K. Gatermann, *Computer algebra methods for equivariant dynamical systems,* Springer, Berlin, 2000.

[42] K. Geddes, S. R. Czapor, G. Labahn, *Algorithms for computer algebra,* Kluwer Academic Publishers, Boston, 1992.

[43] G. Gonnet, *A study of iteration formulas for root finding, where mathematics, computer algebra and software engineering meet,* in these proceedings.

[44] L. Gonzalez-Vega, T. Recio, *Industrial applications of computer algebra: climbing up a mountain, going down a hill,* in these proceedings.

[45] P. Gordan, *Neuer Beweis des Hilbertschen Satzes über homogene Funktionen,* Nachrichten König. Ges. der Wiss. zu Gött., 1899, 240–242, English translation by M. Abramson in SIGSAM Bulletin **32**, Number 2 (1998), 47–48.

[46] R. W. Gosper, Jr., *Decision procedure for indefinite hypergeometric summation,* Proc. Nat. Acad. Sci. USA **75** (1978), 40–42.

[47] J. H. Grace, A. Young, *The algebra of invariants,* Cambridge University Press, Cambridge, 1903.

[48] G.-M. Greuel, *Applications of computer algebra to algebraic geometry, singularity theory and symbolic-numerical solving,* in these proceedings.

[49] J. Hammond, *On the solution of the differential equation of sources,* Amer. J. of Math. **5** (1883), 218–227.

[50] D. Hilbert, *Über die Theorie der algebraischen Formen,* Math. Ann. **36** (1890), 473–534.

[51] D. Hilbert, *Über die vollen Invariantensysteme,* Math. Ann. **42** (1893), 313–373.

[52] A. Hirschowitz, C. Simpson, *La resolution minimale de l'ideal d'un arrangement general d'un grand nombre de points dans* $\mathbf{P}^n$*,* Invent. Math. **126** (1996), 467–503.

[53] H. G. Kahrimanian, *Analytical differentiation by a digital computer,* Master's thesis, Temple University, Philadelphia, 1953.

[54] E. Kaltofen, *Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization,* SIAM J. Comp. **14** (1985), 469–489.

[55] E. Kaltofen, *Polynomial factorization,* in B. Buchberger et al. (eds.), *Computer algebra,* 95–113, Springer, Wien, 1982.

[56] E. Kaltofen, *Polynomial factorization 1982-1986,* in I. Simon (ed.), *Computers in mathematics,* 285–309, Marcel Dekker, New York, 1990.

[57] E. Kaltofen, *Polynomial factorization 1987–1991,* in D.V. Chudnovsky, R.D. Jenks (eds.), *Proceedings of LATIN'92, Sao Paulo,* 294–313, Springer, New York, 1992.

[58] S. Katz, *On the finiteness of rational curves on quintic threefolds,* Composito Math. **60** (1986), 151–162.

[59] W. Krandick, S. Rump (eds.), *Special issue on Validated numerical methods and computer algebra,* J. Symbolic Computation **24** (1997), 649–803.

[60] L. Kronecker, *Grundzüge einer arithmetischen Theorie algebraischer Grössen,* J. Reine Angew. Math. **92** (1882), 1–122.

[61] J. P. S. Kung, G.-C. Rota, *The Invariant Theory of Binary Forms,* Bull. Am. Math. Soc. **10** (1984), 27–85.

[62] L. A. Lambe (ed.), *Special issue,* J. Symbolic Computation **23** (1997), 445–623.

[63] P. L. Larcombe, *On Lovelace, Babbage and the origins of computer algebra,* in [99].

[64] A. K. Lenstra, H. W. Lenstra, Jr., *Algorithms in number theory,* in J. van Leeuwen (ed.), *Algorithms and complexity,*, Volume A, 673–716, Elsevier, Amsterdam, 1990.

[65] A. K. Lenstra, H. W. Lenstra, Jr. (eds), *The development of the number field sieve,* Springer, Berlin, 1993.

[66] A. K. Lenstra, H. W. Lenstra, Jr., L. Lovász, *Factoring polynomials with rational coefficients,* Math. Ann. **261** (1982), 515–534.

[67] H. W. Lenstra, Jr., *Factoring integers with elliptic curves,* Ann. of Math. **126** (1987), 649–673.

[68] H. W. Lenstra, Jr., *Flags and lattice basis reduction,* in these proceedings.

[69] A. Lorenzini, *The minimal resolution conjecture,* J. Algebra **156** (1993), 5–35.

[70] L. F. Menabrea, *Sketch of the analytical engine invented by Charles Babbage. With notes upon the memoir by the translator, Ada Augusta, Countess of Lovelace,* in P. Morrison and E. Morrison (eds.), *Charles Babbage and his calculating engines,* part II, Dover Publications, New York, 1961.

[71] W. F. Meyer, *Invariantentheorie,* in *Encyklopädie der mathematischen Wissenschaften, Erster Band,* Teubner, Leipzig, 1898–1904.

[72] H. M. Möller, *Gröbner bases and numerical analysis,* in [18], 159–178.

[73] J. F. Nolan, *Analytical differentiation on a digital computer,* Master's thesis, MIT, Cambridge, 1953.

[74] A. M. Odlyzko, H. J. J. te Riele, *Disproof of the Mertens conjecture,* J. Reine Angew. Math. **357** (1985), 138–160.

[75] P. J. Olver, Classical invariant theory, Cambridge University Press, Cambridge, 1999.

[76] S. R. Petrick (ed.), *Proc. of the Second Symposium on Symbolic and Algebraic Manipulation (SYMSAM'71),* Los Angeles, ACM Press, New York, 1971.

[77] C. Pomerance, *Analysis and comparism of some factoring algorithms,* in H.W. Lenstra, Jr., R. Tijdeman (eds.), *Computational methods in number theory,* 89–139, Mathematical Centre Tracts **154/155**, Math. Centrum, Amsterdam, 1982.

[78] C. Pomerance, *The quadratic sieve factoring algorithm,* in T. Beth et al. (eds.), *Advances in cryptology,* Springer, Berlin, 1985.

[79] S. Popescu, On smooth surfaces of degree ≥ 11 in $\mathbf{P}^4$, PhD thesis, Universität des Saarlandes, Saarbrücken, 1993.

[80] S. Popescu, K. Ranestad, Surfaces of degree 10 in the projective fourspace via linear systems and linkage, J. Algebraic Geometry **5** (1996), 13–76.

[81] R. Risch, *On the integration of elementary functions which are built up using algebraic operations,* Report SP-2801/002/00, System Development Corp., Santa Monica, 1968.

[82] R. Risch, *Further results on elementary functions,* Report RC-2402, IBM Corp., Yorktown Heights, 1969.

[83] R. Risch, *The problem of integration in finite terms,* Trans. A. M. S. **139** (1969), 167–189.

[84] R. Risch, *The solution of the problem of integration in finite terms,* Bull. A. M. S. **76** (1970), 605–608.

[85] R. L. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems,* Comm. ACM **21** (1978), 120–126.

[86] P. Roelse, *Factoring high-degree polynomials over* $\mathbf{F}_2$ *with Niederreiter's algorithm on the IBM SP2,* Math. Comp. **68** (1999), 869–880.

[87] F.-O. Schreyer, *Small fields in constructive algebraic geometry,* 221–228, in *Moduli of vector bundles, Sanda 1994,* 221–228, Dekker, New York, 1996.

[88] F.-O. Schreyer, F. Tognoli, *Constructions and investigations with small finite fields,* to appear in D. Eisenbud et al. (eds.), *Mathematical computations with Macaulay2,* Springer, New York.

[89] F. T. von Schubert, *De inventione divisorum,* Nova Acta Academiae Scientiarum Imperalis Petropolitanae **11** (1793), 172–182.

[90] I. E. Shparlinski, *Finite fields: theory and computations,* Kluwer Academic Publishers, Dordrecht, 1999.

[91] M. F. Singer, *Formal solutions of differential equations,* J. Symbolic Computation **10** (1990), 59–94.

[92] L. Smith, *Polynomial Invariants of Finite Groups,* A.K. Peters, Wellesley, 1995.

[93] B. Sturmfels, *Algorithms in invariant theory,* Springer, Wien, 1993.

[94] D. D. Swade, *Der mechanische Computer des Charles Babbage,* Spektrum der Wissenschaft, April 1993.

[95] J. J. Sylvester, *Tables of the generating functions and groundforms for the binary quantics of the first ten orders,* Amer. J. Math. **2** (1879), 223–251.

[96] J. J. Sylvester, *Tables of the generating functions and groundforms for the binary duodecimic, with some general remarks, and tables of the irreducible syzygies of certain quantics,* Amer. J. Math. **4** (1881), 41–61.

[97] B. Trager, *Integration of algebraic functions,* PhD thesis, MIT, Boston, 1984.

[98] S. M. Watt, H. Stetter (eds.), *Special issue on Symbolic numeric algebra for polynomials,* J. Symbolic Computation **26** (1998), 649–652.

[99] M. Wester (ed.), *Computer algebra systems. A practical guide,* Wiley, Chichester, 1999.

[100] H. Zassenhaus, *On Hensel factorization,* J. Number Theory **1** (1969), 291–311.

Department of Mathematics,
Universität des Saarlandes,
D-66041 Saarbrücken, Germany
E-mail address: `decker@math.uni-sb.de`