"There are three kinds of mathematicians,
those who can count and those who can't"

1. What is Computer Algebra ?

Mathematicians in the old period, say before 1850 A.D., solved the majority of mathematical problems by extensive calculations. A typical example of this type of mathematical problem solver is Euler. Even Gauss in 1801 temporarily abandoned his research in arithmetic and number theory in order to calculate the orbit of the newly discovered planetoid Ceres. It was this calculation much more than his masterpiece *Disquisitiones Arithmeticae* which became the basis for his fame as the most important mathematician of his time.

So it is not astonishing that in the 18th and beginning 19th centuries many mathematicians were real wizzards of computation. However, during the 19th century the style of mathematical research changed from quantitative to qualitative aspects. A number of reasons were responsible for this change, among them the importance of providing a sound basis for the vast theory of analysis. But the fact that computations gradually became more and more complicated certainly also played its rôle. This impediment has been removed by the advent of modern digital computers in general and by the development of program systems in computer algebra in particular. By the aid of computer algebra the capacity for mathematical problem solving has been decisively improved.

Even in our days many mathematicians think that there is a natural division of labor between man and computer: a person applies the appropriate algebraic transformations to the problem at hand and finally arrives at a program which then can be left to a "number crunching" computer. But already in 1844 Lady Augusta Ada Byron, countess Lovelace, recognized that this division of labor is not inherent in mathematical problem solving and may be even detrimental. In describing the possible applications of the *Analytical Engine* developed by Charles Babbage she writes:

"Many persons who are not conversant with mathematical studies imagine that because the business of [Babbage's Analytical Engine] is to give its results in numerical notation, the nature of its processes must consequently be arithmetical and numerical 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."

And indeed a modern digital computer is a "universal" machine capable of carrying out an arbitrary algorithm, i.e. an exactly specified procedure, algebraic algorithms being no exceptions.

An attempt at a definiton

Now what exactly is symbolic algebraic computation, or in other words *computer algebra*? In his introduction to (Buchberger et al. 1983) R. Loos gave the following attempt at a definition:

> *Computer algebra is that part of computer science which designs, analyzes, implements, and applies algebraic algorithms.*

While it is arguable whether computer algebra is part of computer science or mathematics, we certainly agree with the rest of the statement. In fact, in our view computer algebra is a special form of scientific computation, and it comprises a wide range of basic goals, methods, and applications. In contrast to numerical computation the emphasis is on computing with symbols representing mathematical concepts. Of course that does not mean that computer algebra is devoid of computations with numbers. Decimal or other positional representations of integers, rational numbers and the like appear in any symbolic computation. But integers or real numbers are not the sole objects. In addition to these basic numerical entities computer algebra deals with polynomials, rational functions, trigonometric functions, algebraic numbers, etc. That does not mean that we will not need numerical algorithms any more. Both forms of scientific computation have their merits and they should be combined in a computational environment. For instance, in order to compute an approximate solution to a differential equation it might be reasonable to determine the first n terms of a power series solution by exact methods from computer algebra before handing these terms over to a numerical package for evaluating the power series.

Summarizing, we might list the following characteristics of computer algebra:

(1) Computer algebra is concerned with computing in algebraic structures. This might mean in basic algebraic number domains, in algebraic extensions of such domains, in polynomial rings or function fields, in differential or difference fields, in the abstract setting of group theory, or the like. Often it is more economical in terms of computation time to simplify an expression algebraically before evaluating it numerically. In this way the expression becomes simpler and less prone to numerical errors.

(2) The results of computer algebra algorithms are exact and not subject to approximation errors. So, typically, when we solve a system of algebraic equations like

$$\begin{aligned} x^4 + 2x^2y^2 + 3x^2y + y^4 - y^3 &= 0 \\ x^2 + y^2 - 1 &= 0 \end{aligned}$$

we are interested in an exact representation $(\sqrt{3}/2, -1/2)$ instead of an approximative one $(0.86602\ldots, -0.5)$.

(3) In general the inputs to algorithms are expressions or formulas and one also expects expressions or formulas as the result. Computer algebra algorithms are capable of giving results in algebraic form rather than numerical values for specific evaluation points. From such an algebraic expression one can deduce how changes in the parameters affect the result of the computation. So a typical result of computer algebra is

$$\int \frac{x}{x^2 - a} dx = \frac{\ln \mid x^2 - a \mid}{2}$$

instead of

$$\int_0^{1/2} \frac{x}{x^2-1} dx = -0.1438\ldots$$

As a consequence one can build decision algorithms on computer algebra, e.g. for the factorizability of polynomials, the equivalence of algebraic expressions, the solvability of integration problems in a specific class of expressions, the solvability of certain differential equation problems, the solvability of systems of algebraic equations, the validity of geometric statements, the parametrizability of algebraic curves.

There is a variety of software systems for computer algebra. Some of the most widely used systems are *Mathematica*, *Maple*, *Macaulay*, *Magma*, *SageMath*, *Derive*, *Reduce*. Also *geogebra* relies heavily on computer algebra algorithms.

Topic of this course – Solving systems of algebraic equations

In this course we will not present the full scope of computer algebra, but we will concentrate on the theory of solving systems of polynomial or algebraic equations. In particular we deal with the theories of greatest common divisors, resultants, and Gröbner bases. If we can split one of the equations into factors, then we can replace our system by (several) simpler ones.

A system of polynomial or algebraic equations is of the form

$$\begin{aligned} f_1(x_1,\ldots,x_n) &= 0, \\ &\vdots \\ f_m(x_1,\ldots,x_n) &= 0, \end{aligned}$$

where the left hand sides are polynomials in n variables with coefficients in a field K; i.e., $f_i(x_1,\ldots,x_n) \in K[x_1,\ldots,x_n]$. The solutions are to be determined in $\overline{K}^n$, where $\overline{K}$ is the algebraic closure of K.

If n, the number of variables, is 1, then we need to compute and factor the greatest common divisor of the f_i's. The Euclidean algorithm can be employed for the computation of the GCD. If d, the degree of the equations, is 1, then we solve these linear equations by Gaussian elimination. But what are we to do if $n > 1$ and $d > 1$? This is the case for resultants and Gröbner bases.