Introduction to Computer Algebra

Prof. Dr. Wolfram Koepf
Department of Mathematics
University of Kassel

koepf@mathematik.uni-kassel.de

http://www.mathematik.uni-kassel.de/~koepf

Yaounde, Cameroon
March 22, 2005

UNIKASSEL
VERSITÄT

History of Computer Algebra

- Some years after the first programming languages like Fortran or Algol 60 were designed, the first computer algebra systems were developed.
- Physicists were the first ones who were interested in symbolic computations done by a computer to save lengthy hand computations and to avoid mistakes.
- In the 1960s the programming language LISP was especially well suited for this purpose.

History of Computer Algebra

- **1968**: *Reduce*, Anthony Hearn, physicist, LISP-based. Oldest system, still on the market!
- **1970/1992**: *Scratchpad, Axiom*, IBM, LISP. Strongly typed system based on mathematical structures. Now free version available.
- **1971**: *Macsyma*, MIT, LISP. Now as free system *Maxima* still on the market.
- **1978**: *mumath*, David Stoutemyer, LISP. First system designed for mini-computers. System was later replaced by *Derive*.

History of Computer Algebra

- **1980**: *Maple*, University of Waterloo, C.
  First C-based system. Small kernel, mainly programmed in Maple language.
- **1988**: *Mathematica*, Stephen Wolfram, physicist, C.
  Best-selling system. First system which combined symbolics, numerics, graphics and a nice user interface.
- **1989**: *Derive*, David Stoutemyer, LISP.
  mumath-successor. PC-system, mainly used in education.
- **1993**: *MuPAD*, Benno Fuchssteiner, C.
  Object oriented computer algebra system.

On-line Demonstration of Maple

- In this talk I will use the computer algebra system *Maple* to show you the capabilities of such systems.

- If you have any question, please don't hesitate to interrupt me and ask! It is much easier to answer your questions directly when they evolve.

- Let us start with the *Maple* demonstration.

Euclidean Algorithm

To compute the greatest common divisor of a and b, we can use the following recursive algorithm:

- $\gcd(a, b) := \gcd(|a|, |b|)$ if $a < 0$ or $b < 0$
- $\gcd(a, b) := \gcd(b, a)$ if $a < b$
- $\gcd(a, 0) := a$ (stop condition)
- $\gcd(a, b) := \gcd(b, a \text{ mod } b)$

Modular Powers

As further example, we consider the fast computation of modular powers. To compute the modular power $a^n \pmod{p}$ efficiently, one tries to replace the exponent n by $n/2$ (divide and conquer algorithm):

- $a^0 \pmod{p} := 1$
- $a^n \pmod{p} := (a^{n/2} \pmod{p})^2 \pmod{p}$ if n is even
- $a^n \pmod{p} := (a^{n-1} \pmod{p} \cdot a) \pmod{p}$ if n is odd

Fermat's Little Theorem

- For every $p \in \mathbb{P}$ and $a \in \mathbb{Z}$ one has
$$a^p \equiv a \pmod{p}\ .$$
- Using modular powers, one can efficiently check *Fermat's Little Theorem*.
- Fermat Test: If this relation is not fulfilled for some $a \in \mathbb{Z}$, then p cannot be a prime!
- Modular powers are also used in modern cryptosystems like RSA.

Cryptography

- Assume A wants to send a secret message M securely to B.
- Then A and B agree upon a known encryption function E with decryption function D.
- A must have an encryption key e.
- $E_e(M)$ is called the cryptogram of message M.
- B must have a decryption key d.
- Of course $D_d(E_e(M)) = M$.

Asymmetric Cryptography

- In 1976 Diffie and Hellman invented asymmetric cryptography, also called public key cryptography.
- Here A and B have different keys, they both make their encryption keys e public, but keep their decryption keys d private.
- The security of such a system depends on the difficulty to find d from e or D_d from E_e.
- For this purpose one uses that some mathematical problems are much more difficult than their inverses. Such functions are called one way functions.

RSA Cryptosystem

- In the RSA cryptosystem (Rivest, Shamir, Adleman 1978) the message M is supposed to be a large integer.
- B chooses two 100-digit primes p and q.
- B sets $n := p \cdot q$ and $\varphi := (p-1)(q-1)$.
- B chooses her public key e relatively prime to φ.
- Public Key: Both e and n are public.

RSA Cryptosystem

- Private Key: Next B can compute her private key d such that $e \cdot d \equiv 1 \pmod{\varphi}$.

- For security reasons p, q and φ are deleted.

- The RSA encryption and decryption functions are given by

$$E_e(M) = M^e \pmod{n} \quad \text{and} \quad D_d(C) = C^d \pmod{n} .$$

- The cryptographic equation $D_d(E_e(M)) = M$ follows from Fermat's Little Theorem.