Basic Domains of Interest used in Computer Algebra Systems

Lecture 2

We can compute with any finitely representable objects, at least in principle

- Objects from any algebraic system, and even some ad-hoc mixtures
- With any algorithms
  - All steps specified precisely
  - Terminating
- Some processes are not-necessarily terminating. E.g. L'hôpital's rule … use termination heuristics: time/space limits, losing some credibility for results ☹
- But we tend to concentrate on domains that occur in applied math, physical sciences, etc.

The Usual Operations

- Integer and Rational:
  - Ring and Field operations +- * exact quotient, remainder
- GCD, factoring of integers
- Approximation via rootfinding
- Polynomial operations
  - Ring, Field, GCD, factor
  - Truncated power series
  - Solution of polynomial systems
- Matrix operations (add determinant, resultant, eigenvalues, etc.)

More Operations

- Sorting (e.g. of monomials)
- Union (collections)
- Tests for zero
- Extraction of parts (polynomial degree, constant coefficient, leading coefficient)
- Conversion to different forms ("expand", express algebraic function in a minimal extension, "simplify")

Yet More Operations

Differentiate

Integrate

Limit

Prove

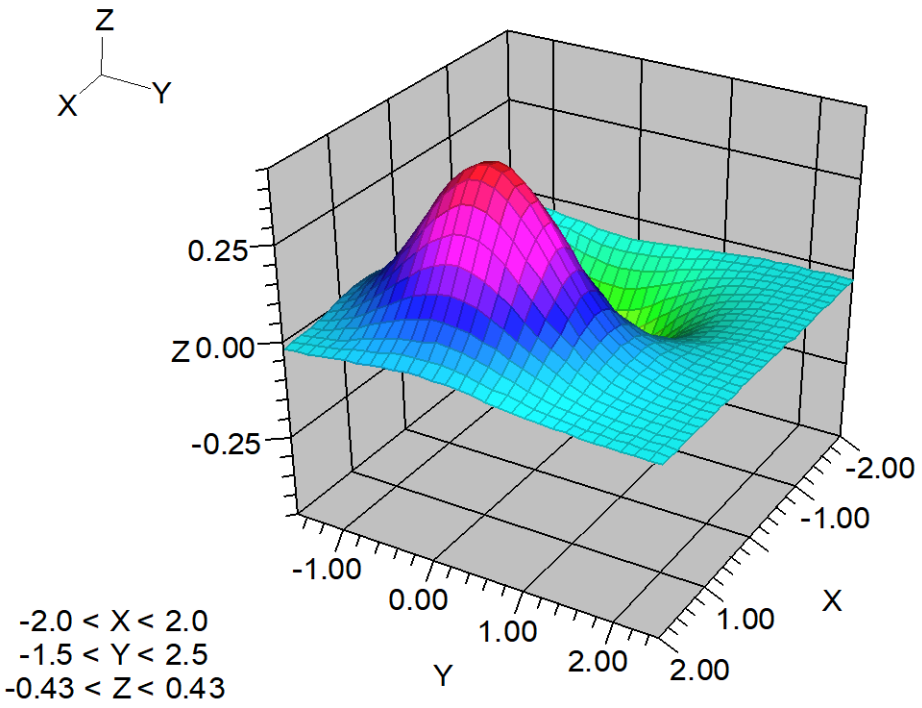
Find region in which (in)equalities hold

Confirm (as: steps in a proof)

Expand in series

Yet More Operations

Plot



plot3d(exp(-x^2-y^2)*x,x,-2,2,y,-1.5,2.5)

Yet More Operations

Typesetting

$$\text{factor}\left(\text{determinant}\left(\begin{bmatrix} 1 & x & x^2 \\ 1 & y & y^2 \\ 1 & z & z^2 \end{bmatrix}\right)\right) = (y - x)\,(z - x)\,(z - y)$$

Integer representations, operations

- The ring operations +*-
- Euclidean Domain: quotient & remainder, GCD
- UFD : factorization

Unfortunately computers don't do these operations directly

Addition modulo $2^{31}-1$ is rarely what we need.

How do we do arbitrary precision integer arithmetic? (If we could do this, we could build the rationals, and via intervals or some other construction, we could make reals)

Is it hard to do arbitrary integer (bignum) arithmetic?

- In spite of your knowledge of this subject, there are subtleties, especially in long division!
- You must choose fast algorithms (moderate size) or asymptotically optimal algorithms (large size): what's your target?

Who cares about integer arithmetic?

- You need fast long arithmetic to compute billions of digits of π e.g. DH Bailey's home page
- You need fast moderate-length arithmetic to play integer-factoring games.
- Arguably, there are sensitive geometric predicates that require very high precision (floats).
- You need all lengths to build a computer algebra system: without it your system tells lies.
- Every Common Lisp has bignums built in.

Some ideas just for representing integers

Integers are sequences of characters, 0..9.

Integers are sequences of words modulo 10^9 which is the largest power of 10 less than 2^{31}.

Integers are sequences of hexadecimal digits.

Integers are sequences of 32-bit words.

Integers are sequences of 64-bit double-floats (with 8 bits wasted).

Sequences are linked lists

Sequences are vectors

Sequences are stored in sequential disk locations

Yet more ideas

Integers are sequences of 64-bit double-floats (with 8 bits used to position the bits)

e.g. $2^{-300}+2^{300}$ takes 2 words

Integers are stored in redundant form a+b+…

Integer are stored modulo set of different primes

Integers are stored in p-adic form as a sequence of x mod p, x mod p^2, …

To be continued..

- Integers and more.