SNAP:
A Maple 8 package
for arithmetic with numeric polynomials

Claude-Pierre Jeannerod
Laboratoire LIP
INRIA - ENS Lyon
France

George Labahn
Symbolic Computation Group
University of Waterloo
Canada

ICMS 2002

Introduction: Symbolic-Numeric Algorithms for Polynomials

Introduction: Symbolic-Numeric Algorithms for Polynomials

Data: Univariate polynomials with real floating-point coefficients

Introduction: Symbolic-Numeric Algorithms for Polynomials

Data: Univariate polynomials with real floating-point coefficients

Computations: Division, remainder, GCD, ...

Introduction: Symbolic-Numeric Algorithms for Polynomials

Data: Univariate polynomials with real floating-point coefficients

Computations: Division, remainder, GCD, ...

Model: IEEE 754 floating-point standard

Introduction: Symbolic-Numeric Algorithms for Polynomials

Data: Univariate polynomials with real floating-point coefficients

Computations: Division, remainder, GCD, ...

Model: IEEE 754 floating-point standard

Goal: Implement provably fast and numerically stable algorithms

Example: numerical coprimeness test

```
> with(SNAP):
> a := 0.1*z^2+1.5*z-0.2:
> b := 0.2*z^3+0.15:
> AreCoprime(a,b,z,0.5);
```

false

```
> AreCoprime(a,b,z,0.1);
```

true

Why look at symbolic-numeric problems?

- Linear systems theory
- Control theory
- Computer aided design
- ...

SNAP functionalities

AreCoprime
DistanceToCommonDivisors
DistanceToSingularPolynomials
EpsilonGCD
EuclideanReduction
IsSingular
QuasiGCD
Quotient
Remainder

SNAP functionalities

AreCoprime
DistanceToCommonDivisors
DistanceToSingularPolynomials
EpsilonGCD
EuclideanReduction
IsSingular
QuasiGCD
Quotient
Remainder

Division

SNAP functionalities

AreCoprime
DistanceToCommonDivisors
DistanceToSingularPolynomials
EpsilonGCD
EuclideanReduction
IsSingular
QuasiGCD
Quotient
Remainder

Division
Distance problems

SNAP functionalities

AreCoprime
DistanceToCommonDivisors
DistanceToSingularPolynomials
EpsilonGCD
EuclideanReduction
IsSingular
QuasiGCD
Quotient
Remainder

Division
Distance problems
Tests

SNAP functionalities

AreCoprime
DistanceToCommonDivisors
DistanceToSingularPolynomials
EpsilonGCD
EuclideanReduction
IsSingular
QuasiGCD
Quotient
Remainder

Division
Distance problems
Tests
Approximate GCDs

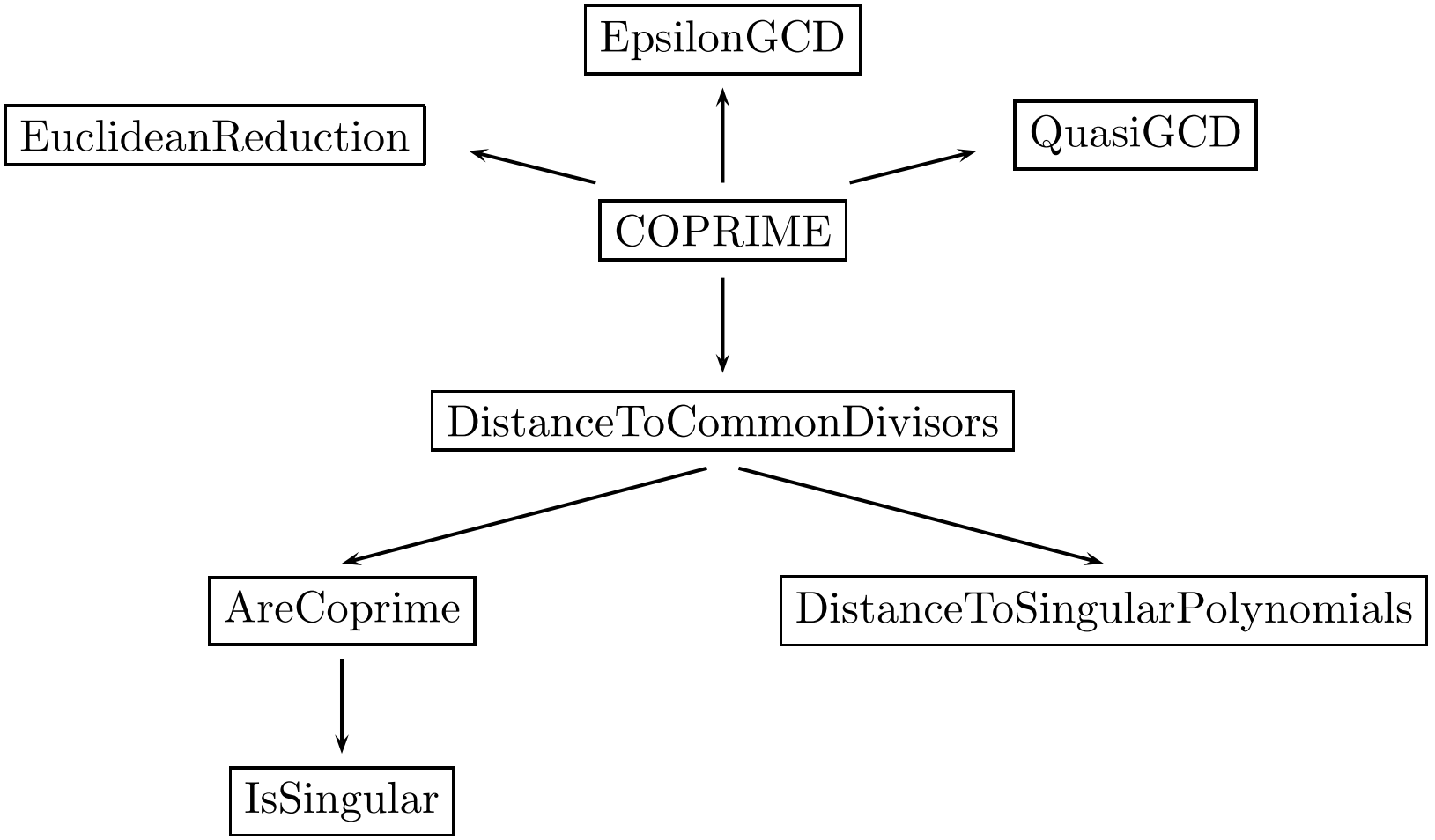
Algorithmic Structure of SNAP package

Algorithmic Structure of SNAP package

- Central procedure is COPRIME algorithm which solves :

$$\begin{aligned} a(z) \cdot s(z) + b(z) \cdot t(z) &\approx 1 \\ a(z) \cdot u(z) + b(z) \cdot v(z) &\approx z^{m+n-1} \end{aligned}$$

- Can use for efficient numerical coprimeness test
- Solved via a numerically stable Euclidean-like reduction algorithm
- Can use to often find certified approximate GCDs


EpsilonGCD
EuclideanReduction
QuasiGCD
COPRIME
DistanceToCommonDivisors
AreCoprime
DistanceToSingularPolynomials
IsSingular

Efficient numerical coprimeness test

Timings (in seconds) for polynomials of degrees m, n

$m = n + 1$	LinearAlgebra*	SNAP
50	0.33	0.34
100	1.30	1.34
250	8.6	8.7
500	43	33
10^3	307	126
$2 \cdot 10^3$	3360	515

Pentium III 800 MHz with 512MB of RAM, under Linux

* = solve a Sylvester linear system with `LinearAlgebra[LinearS olve]`.

DistanceToCommonDivisors($a(z), b(z), \epsilon$)

Given $a(z) = \sum_{i=0}^{m} a_i z^i$ and $b(z) = \sum_{i=0}^{n} b_i z^i$, the set

$$P = \{(p, q) \, : \, \deg p \leq m, \, \deg q \leq n, \, \deg \gcd(p, q) > 0\}$$

and a suitable norm $\| \cdot \|$, estimate the distance

$$\text{dist} = \inf_{P} \|(a, b) - (p, q)\| \quad ?$$

DistanceToCommonDivisors($a(z), b(z), \epsilon$)

Given $a(z) = \sum_{i=0}^{m} a_i z^i$ and $b(z) = \sum_{i=0}^{n} b_i z^i$, the set

$$P = \{(p, q) \ : \ \deg p \leq m, \ \deg q \leq n, \ \deg \gcd(p, q) > 0\}$$

and a suitable norm $\| \cdot \|$, estimate the distance

$$\text{dist} = \inf_{P} \|(a, b) - (p, q)\| \quad ?$$

Can solve this using linear algebra.

Norms

View $c \in \mathbb{C}[z]$, $c(z) = c_0 + .. + c_n z^n$ as the vector $\vec{c} = (c_0, .., c_n)^T$.

- Norm for $\mathbb{C}[z]$: $\quad ||c|| = ||\vec{c}||_1 = \sum_j |c_j|$
- Norm for $\mathbb{C}[z]^{r \times s}$: $\quad ||(c_{j,k})|| = ||(||c_{j,k}||)||_1 = \max_k \sum_j ||c_{j,k}||$.

Linear algebra formulation

- Sylvester matrix for $a(z)$ and $b(z)$:

$$S = \left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right]$$

Linear algebra formulation

- Sylvester matrix for $a(z)$ and $b(z)$:

$$S = \left[\begin{array}{ccc|ccc} a_0 & & & b_0 & & \\ a_1 & a_0 & & b_1 & b_0 & \\ a_2 & a_1 & & b_2 & b_1 & b_0 \\ a_3 & a_2 & & & b_2 & b_1 \\ & a_3 & & & & b_2 \end{array}\right]$$

- Well known: $\det S \neq 0 \iff \gcd(a, b) = 1$

$$\iff \text{can solve } a(z)s(z) + b(z)t(z) = 1.$$

Linear algebra formulation (cont'd)

Also well known: classical lower bound $\frac{1}{||S^{-1}||_1} \leq \text{dist}$.

Proof: $||S||_1 = ||(a,b)||$ where $||B||_1 = \max_{x \neq 0} \frac{||Bx||_1}{||x||}$

$$\implies \quad \text{dist} = \inf\{||S - S(p,q)||_1 : S(p,q) \text{ singular}\}$$

$$\geq \min\{||S - B||_1 : \ B \text{ singular}\} = \frac{1}{||S^{-1}||_1}$$

□

Note: $||S||_1$ can be computed via SVD.

A better lower bound [Beckermann & Labahn - 1998]

Sylvester matrix for $a(z)$ and $b(z)$ and its inverse:

$$S = \left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right] \qquad S^{-1} = \left[\begin{array}{ccccc} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{array}\right]$$

$$\frac{1}{\|S^{-1}\|_1} \leq \frac{1}{\kappa} \leq \text{dist} \quad \text{defined by } \kappa = \max(\textstyle\sum_i |*_i|, \sum_i |*_i|).$$

EuclideanReduction($a(z), b(z), \epsilon$**)**

EuclideanReduction($a(z), b(z), \epsilon$)

- Numerically stable euclidean reduction.
  - algorithm "jumps" remainders resulting from ill-conditioned subproblems
  - finds such ill-conditioned subsystems via estimating condition numbers
  - fast : quadratic rather than cubic
  - used to determine if two numeric polynomials are coprime
  - can be used for additional numerical GCD computations

Certified approximate GCDs (Rupprecht 's example)

$$a(z) = (z^4 - 1.000001)(z^3 - 3.000001z + 0.99999999)$$

$$b(z) = (z^4 - 0.9999999)(z^4 - 3.0000003z - 2.9999999)$$

`SNAP[EpsilonGCD](a,b,z)` with 10 digits yields the ϵ-GCD

$$z^4 + 5.198691325\,10^{-8} z^3 + 5.804634175\,10^{-7} z^2 + 9.873862873\,10^{-7} z - 1.000000584$$

with $\epsilon = 0.0006481143605.$

Satisfactory answers from Euclidean reduction

$$a(z) = z^6 - 12.4z^5 + 62.53z^4 - 163.542z^3 + 232.9776z^2 - 170.69184z + 50.18112$$

$$b(z) = z^5 - 17.6z^4 + 118.26z^3 - 372.992z^2 + 538.3333z - 274.09272$$

`SNAP[EuclideanReduction](a,b,z,tau=1e-8)` with 10 digits yields the reduced pair

$$0.2500000000z^2 - 0.8750003765z + 0.6600005057,$$
$$-0.90726410^{-7}z + 0.133529610^{-6}.$$

Euclidean algorithm

- Typical polynomial remainder sequence:

$$\begin{bmatrix} a & b \end{bmatrix} \xrightarrow{\times U^{(1)}} \begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix} \xrightarrow{\times U^{(2)}} \begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix} \xrightarrow{\times U^{(3)}} \begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$\begin{array}{cccccccc} 3 & 2 & & 2 & 1 & & 1 & 0 & & 0 & -\infty \end{array}$$

Euclidean algorithm

- Typical polynomial remainder sequence:

$$\begin{bmatrix} a & b \end{bmatrix} \xrightarrow{\times U^{(1)}} \begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix} \xrightarrow{\times U^{(2)}} \begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix} \xrightarrow{\times U^{(3)}} \begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$\begin{array}{cccccccc} 3 & 2 & & 2 & 1 & & 1 & 0 & & 0 & -\infty \end{array}$$

The first column of $U^{(1)}U^{(2)}U^{(3)}$ is $\begin{bmatrix} s(z) \\ t(z) \end{bmatrix}$ s.t. $as + bt = 1$.

- Fast but unstable!

Linear algebra interpretation

$$a(z) = 1 + \epsilon z + (1+\epsilon)\, z^2 + z^3 + z^4, \qquad b(z) = -2 + 3\, z - z^2 + z^3$$

$$S = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & -2 & 0 & 0 & 0 \\ \epsilon & 1 & 0 & 3 & -2 & 0 & 0 \\ 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 & 0 \\ 1 & 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 \\ 1 & 1 & 1+\epsilon & 0 & 1 & -1 & 3 \\ 0 & 1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Euclid's algorithm solves linear subsystems.

Linear algebra interpretation

$$a(z) = 1 + \epsilon z + (1+\epsilon)\, z^2 + z^3 + z^4, \qquad b(z) = -2 + 3\, z - z^2 + z^3$$

$$S = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & -2 & 0 & 0 & 0 \\ \epsilon & 1 & 0 & 3 & -2 & 0 & 0 \\ 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 & 0 \\ 1 & 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 \\ 1 & 1 & 1+\epsilon & 0 & 1 & -1 & 3 \\ 0 & 1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Euclid's algorithm solves linear subsystems.

Linear algebra interpretation

$$a(z) = 1 + \epsilon z + (1+\epsilon)\, z^2 + z^3 + z^4, \qquad b(z) = -2 + 3\, z - z^2 + z^3$$

$$S = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & -2 & 0 & 0 & 0 \\ \epsilon & 1 & 0 & 3 & -2 & 0 & 0 \\ 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 & 0 \\ 1 & 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 \\ 1 & 1 & 1+\epsilon & 0 & 1 & -1 & 3 \\ 0 & 1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Euclid's algorithm solves linear subsystems.

Linear algebra interpretation

$$a(z) = 1 + \epsilon z + (1+\epsilon)\, z^2 + z^3 + z^4, \qquad b(z) = -2 + 3\, z - z^2 + z^3$$

$$S = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & -2 & 0 & 0 & 0 \\ \epsilon & 1 & 0 & 3 & -2 & 0 & 0 \\ 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 & 0 \\ 1 & 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 \\ 1 & 1 & 1+\epsilon & 0 & 1 & -1 & 3 \\ 0 & 1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Euclid's algorithm solves linear subsystems.

Problem: what to do with ill-conditioned subsystems?

COPRIME algorithm [Beckermann & Labahn - 1998]

Look-ahead process: if $|U^{(2)}(0)|$ too small, jump over $\begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix}$:

$$\begin{bmatrix} a & b \end{bmatrix} \overset{\times U^{(1)}}{\longrightarrow} \begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix} \overset{\times U^{(2)}}{\longrightarrow} \begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix} \overset{\times U^{(3)}}{\longrightarrow} \begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

COPRIME algorithm [Beckermann & Labahn - 1998]

Look-ahead process: if $|U^{(2)}(0)|$ too small, jump over $\begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix}$:

$$\begin{bmatrix} a & b \end{bmatrix} \overset{\times U^{(1)}}{\longrightarrow} \begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix} \overset{\times U^{(2)}}{\longrightarrow} \begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix} \overset{\times U^{(3)}}{\longrightarrow} \begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

- Unstability related to the ill-conditioning of some $U^{(i)}$ at $z = 0$.
- Effective way of deducing $\begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix}$ from $\begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix}$.
- Skipping all the small $|U^{(i)}(0)|$ yields a small residual error.
- Provably stable (weak stability).

What about speed?

- At most $\max(m, n)$ jumps where $m = \deg a$ and $n = \deg b$.
- One jump of size s costs $O(s^3)$ flops.

$\hookrightarrow$ In most cases, $s \leq 3 \ll m+n$ and we have a $O((m+n)^2)$ algo.

$\hookrightarrow$ Worst-case: one large jump (i.e. $s \sim m+n$) costs $O((m+n)^3)$.

Summary

- COPRIME algorithm:
  - Fast in most cases
  - Proven weak (or forward) stability
  - Distance estimate + coprimeness test

Summary

- COPRIME algorithm:
  - Fast in most cases
  - Proven weak (or forward) stability
  - Distance estimate + coprimeness test
- Not only a numerical coprimeness test:
  - ϵ-GCD
  - quasi-GCD

  can be guaranteed from the last accepted basis $[a^{(k)}\ b^{(k)}]$.

Concluding remarks

- User's guide available at www.ens-lyon.fr/∼cpjeanne

Concluding remarks

- User's guide available at www.ens-lyon.fr/~cpjeanne

- Extensions:
  - Other kinds of approximate GCDs (≥ 6 definitions)
  - Complex polynomials, multivariate polynomials, systems of polynomials
  - Approximate factorization
  - Symbolic-numerics for matrix polynomials