Exact linear algebra tools for computer assisted proofs

Clément PERNET
LIG, INRIA - Grenoble Université

Dagstuhl Seminar,
November 19, 2009

Introduction

Computer algebra:

- Symbolic manipulations
- Computing exactly with algebraic quantities

Introduction

Computer algebra:

- Symbolic manipulations
- Computing exactly with algebraic quantities

Exact computations:

- $\mathbb{Z}, \mathbb{Q}$ $\Rightarrow$variable size
- $\mathbb{Z}_p$, GF(p^k) $\Rightarrow$fixed size, dedicated arithmetic
- $K[X]$ for $K = \mathbb{Z}, \mathbb{Z}_p, \ldots$.

Computer algebra for computer assisted proofs

Assisting mathematical research:

- simplification of formula
- finding analytical solutions
- manipulating complex mathematical objects (modular forms, abelian groups...)

Experimental mathematics:

- Testing conjectures
- Building large test sets
- Asymptotic matters! (*the larger the test set, the higher the confidence*)

Experimentations
$\Downarrow$
counterexample,
or improved confidence
$\Downarrow$
reformulation
$\Downarrow$
...
$\Downarrow$
solution

Outline

1. Tools and software
2. Application of exact computations for proofs
   - Example 1: graph isomorphism conjecture
   - Example 2: algebraic number theory
3. Hot topics

Plan

1. Tools and software
2. Application of exact computations for proofs
   - Example 1: graph isomorphism conjecture
   - Example 2: algebraic number theory
3. Hot topics

Few ingredients for exact computations

Floating point arithmetic: `float, double`

- Prominent on most architectures: `fma`, SSE, GPU, ...
- Used for:
  - Finite fields: `NTL`, `LinBox`,...
  - Lattice reductions: `fpLLL`
- Exact computations using the mantissa only, or controlled approximation

Few ingredients for exact computations

Floating point arithmetic: `float, double`

- Prominent on most architectures: `fma`, SSE, GPU, ...
- Used for:
  - Finite fields: `NTL`, `LinBox`,...
  - Lattice reductions: `fpLLL`
- Exact computations using the mantissa only, or controlled approximation

Reduction to building block routines

Integer/Polynomial multiplication: Karatsuba, Toom-Cook, FFT

Matrix multiplication: BLAS, Strassen,

$\Rightarrow$ Block recursive reduction algorithms

Tools and software for exact computations

Specialized libraries

finite fields: NTL, Givaro, Lidia, ...

multiprecision integers: GMP, MPIR

polynomials: NTL, Givaro, zn_poly ...

Tools and software for exact computations

Specialized libraries

finite fields: NTL, Givaro, Lidia, ...

multiprecision integers: GMP, MPIR

polynomials: NTL, Givaro, zn_poly ...

Middleware: e.g. LinBox

- generic
- focused on algorithms

Tools and software for exact computations

Specialized libraries

finite fields: NTL, Givaro, Lidia, ...

multiprecision integers: GMP, MPIR

polynomials: NTL, Givaro, zn_poly ...

Middleware: e.g. LinBox

- generic
- focused on algorithms

End-user multi-purpose software

- Maple, Mathematica, MuPad, ... (closed source)
- Sage, Pari, Maxima, ... (open source)

Tools and software for exact computations

Specialized libraries

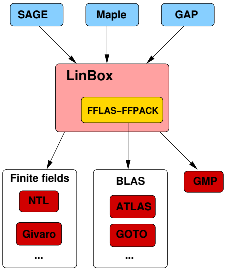
finite fields: NTL, Givaro, Lidia, ...

multiprecision integers: GMP, MPIR

polynomials: NTL, Givaro, zn_poly ...

Middleware: e.g. LinBox

- generic
- focused on algorithms



End-user multi-purpose software

- Maple, Mathematica, MuPad, ... (closed source)
- Sage, Pari, Maxima, ... (open source)

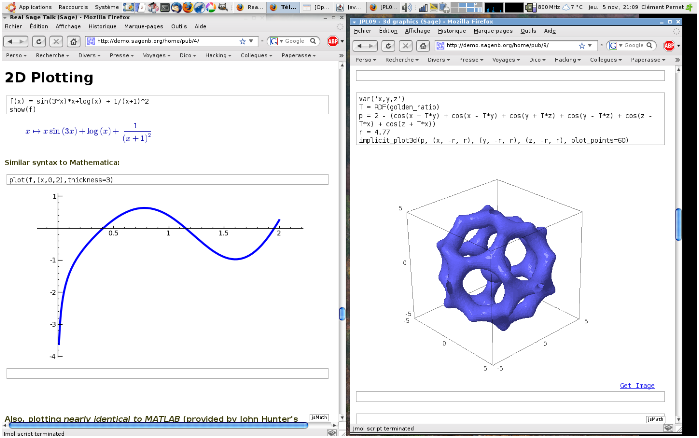
Sage: open source mathematics software

- symbolic computations, geometry, statistics, numerical computations, ...

```
-------------------------------------------------------------------------
| Sage Version 4.2, Release Date: 2009-10-24                            |
| Type notebook() for the GUI, and license() for information.           |
-------------------------------------------------------------------------
sage: R.<x>=Pol<tab>
Polyhedron                    PolynomialQuotientRingElement
Polynomial                    PolynomialRing
PolynomialQuotientRing
sage: R.<x>=PolynomialRing(GF(2))
sage: P=x^2+1
sage: P.parent()
Univariate Polynomial Ring in x over Finite Field of size 2 (using NTL
sage: P.factor()
(x + 1)^2
```

- Linux, MacOS X, Solaris, Windows (port in progress)
- x86, x86_64, PPC,

Sage: graphic interface



- Web integrated
- LaTeX typesetting
- shared worksheet
- 2D, interactive 3D plot
- Interactive applets

Sage: a distribution

- A distribution of the best specialized libraries and software for mathematics (over 70 packages)

Arithmetic	GMP, MPFR, Givaro, MPFI
Commutative algebra	PolyBoRi, SINGULAR (libSINGULAR)
Linear Algebra	LinBox, M4RI, IML, fpLLL
Crypto	GnuTLS, PyCrypto
Integer factorization	FlintQS, ECM
Group theory	GAP
Combinatorics	Symmetrica, sage-combinat
Graph theory	NetworkX
Number theory	PARI, NTL, Flint, mwrank, eclib
Numerical computation	GSL, Numpy, Scipy, ATLAS
Symbolic computation	Maxima, Sympy, Pynac
Statistics	R
User interface	Sage Notebook, jsmath, Moin wiki, IPython
Graphics	Matplotlib, Tachyon, libgd, JMol
Networking	Twisted
Database	ZODB, SQLite, SQLAlchemy, Python pickle
Programming language	Python, Cython (compiled)

Sage

A proper source library

- Over 1M line of code
- Python, Cython (compiled Python)

A development model, based on academic research:

- Over 150 contributors, about 50 per release
- Any new piece of code is proposed, then reviewed by a referee, before getting accepted in the next release
- Automated regression tests, and documentation (Python doctests)

Trust in mathematical result computed by a software?

- high quality code standards
- source inspection!

Plan

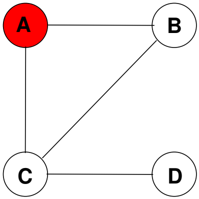
1. Tools and software

2. Application of exact computations for proofs
   - Example 1: graph isomorphism conjecture
   - Example 2: algebraic number theory

3. Hot topics

Graph isomorphism conjecture

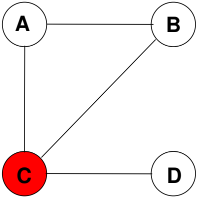
Random Walk in a graph



1-walk

Graph isomorphism conjecture

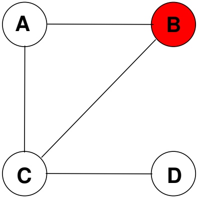
Random Walk in a graph



1-walk

Graph isomorphism conjecture

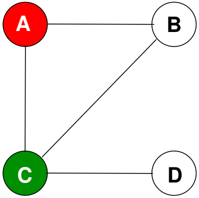
Random Walk in a graph

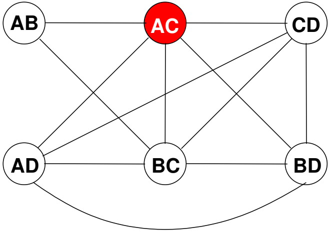


1-walk

Graph isomorphism conjecture

Random Walk in a graph



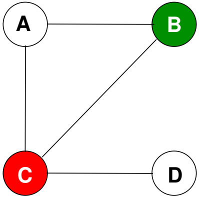


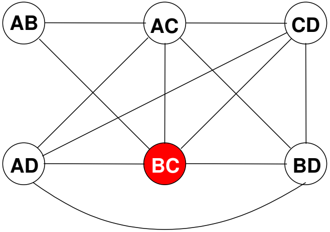
2-walk

- symmetric squares of a graph X :
  $\Rightarrow$the graph $X^{\{2\}}$ of every $\binom{n}{2}$ pairs of vertices
- 2-walk in $X \equiv$ 1-walk in $X^{\{2\}}$

Graph isomorphism conjecture

Random Walk in a graph



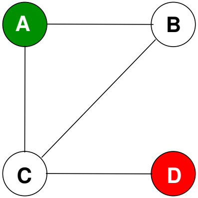


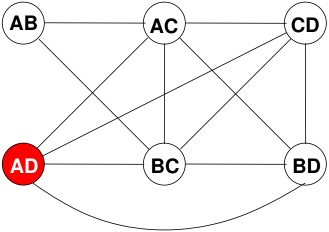
2-walk

- symmetric squares of a graph X :
  $\Rightarrow$the graph $X^{\{2\}}$ of every $\binom{n}{2}$ pairs of vertices
- 2-walk in $X \equiv$ 1-walk in $X^{\{2\}}$

Graph isomorphism conjecture

Random Walk in a graph



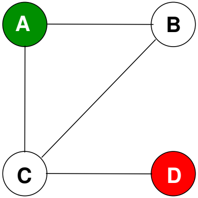


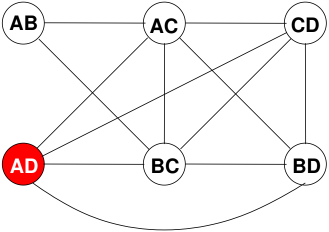
2-walk

- symmetric squares of a graph X :
  $\Rightarrow$the graph $X^{\{2\}}$ of every $\binom{n}{2}$ pairs of vertices
- 2-walk in $X \equiv$ 1-walk in $X^{\{2\}}$

Graph isomorphism conjecture

Random Walk in a graph





2-walk

- symmetric squares of a graph X :
  $\Rightarrow$ the graph $X^{\{2\}}$ of every $\binom{n}{2}$ pairs of vertices
- 2-walk in $X \equiv$ 1-walk in $X^{\{2\}}$

Application :

- Modeling Hamiltonian systems in quantum mechanics
- Graph Isomorphisms

Graph isomorphism

Problem

$\texttt{Graph-isomorphism} \in P$?

Graph isomorphism

Problem

$\texttt{Graph-isomorphism} \in P$?

[Audenaert,& al. 2007] : the spectrum of a symmetric power of the graph determines its isomorphism class ???

Graph isomorphism

Problem

$\mathtt{Graph\text{-}isomorphism} \in P$?

[Audenaert,& al. 2007] : the spectrum of a symmetric power of the graph determines its isomorphism class ???

Experiments: symmetric powers of families of strongly regular graphs

Graph isomorphism

Problem

$\mathtt{Graph\text{-}isomorphism} \in P$?

[Audenaert,& al. 2007] : the spectrum of a symmetric power of the graph determines its isomorphism class ???

Experiments: symmetric powers of families of strongly regular graphs

- $k = 2$: wrong ([Godsil, Royle & al. 2006])
- $k = 3$: true up to 29 edges (70 cases, $n = 3654$)
- $k = 3$: true up to 36 edges (36 510 cases, $n = 7140$)

$\Rightarrow$ 588 CPU hours

Graph isomorphism

Problem

$\texttt{Graph-isomorphism} \in P$?

[Audenaert,& al. 2007] : the spectrum of a symmetric power of the graph determines its isomorphism class ???

Experiments: symmetric powers of families of strongly regular graphs

- $k = 2$: wrong ([Godsil, Royle & al. 2006])
- $k = 3$: true up to 29 edges (70 cases, $n = 3654$)
- $k = 3$: true up to 36 edges (36 510 cases, $n = 7140$)
  $\Rightarrow$588 CPU hours

Compute characteristic polynomials over Z

Sparse 0-1 matrices $\Rightarrow$Black-box model

Black Box linear algebra

Black Box linear algebra

- Matrices viewed as linear operators
- algorithms based on matrix vector apply only $\Rightarrow$cost $E(n)$

Black Box linear algebra

- Matrices viewed as linear operators
- algorithms based on matrix vector apply only $\Rightarrow$cost $E(n)$

Structured matrices: Fast apply (e.g. $E(n) = \mathcal{O}(n \log n)$)

Sparse matrices: Fast apply and no fill-in

Black Box linear algebra

- Matrices viewed as linear operators
- algorithms based on matrix vector apply only $\Rightarrow$cost $E(n)$

Structured matrices: Fast apply (e.g. $E(n) = \mathcal{O}(n \log n)$)

Sparse matrices: Fast apply and no fill-in

$\Rightarrow$

- Iterative methods
- No access to coefficients, trace, no elimination
- Matrix **multiplication** $\Rightarrow$ Black-box **composition**

Black box linear algebra

Minimal polynomial: [Wiedemann 86]

⇒adapts numerical iterative Krylov/Lanczos methods

⇒$\mathcal{O}\left(dE(n) + n^2\right)$ operations

Rank, Det, Solve: [Kaltofen & Saunders 90, Chen& Al. 02]

⇒reduced to minimal polynomial and preconditioners

Black box linear algebra

Minimal polynomial: [Wiedemann 86]

$\Rightarrow$adapts numerical iterative Krylov/Lanczos methods

$\Rightarrow\mathcal{O}\left(dE(n) + n^2\right)$ operations

Rank, Det, Solve: [Kaltofen & Saunders 90, Chen& Al. 02]

$\Rightarrow$reduced to minimal polynomial and preconditioners

$\Rightarrow\mathcal{O}\tilde{}\left(nE(n)\right)$ operations

Black box characteristic polynomial

The method of multiplicities:

1. Compute the minimal polynômial over $\mathbb{Z}$
2. Factor it: $P_{\text{min}} = \prod_i P_i^{m_i}$
3. Determine the e_i s.t. $P_{\text{char}} = \prod_i P_i^{e_i}$

Black box characteristic polynomial

The method of multiplicities:

1. Compute the minimal polynômial over $\mathbb{Z}$
2. Factor it: $P_{\text{min}} = \prod_i P_i^{m_i}$
3. Determine the e_i s.t. $P_{\text{char}} = \prod_i P_i^{e_i}$
   - $\text{rank}(P_i(A)) = n - e_i \deg(P_i)$
   - Index calculus:

$$\sum_{j=1}^{k} \log_g(P_j(\lambda)) e_j = \log_g(\det(\lambda I - A)) \mod (p-1)$$

for several λ's

Outline

1. Tools and software
2. Application of exact computations for proofs
   - Example 1: graph isomorphism conjecture
   - Example 2: algebraic number theory
3. Hot topics

The problem

Clay Math Institute, \$1M challenge:

Problem (Conjecture Birch Swinnerton-Dyer)

A "method" to determine whether any equation of the type $Y^2 = X^3 + aX + b$ has an infinity of rational solutions

The problem

Clay Math Institute, \$1M challenge:

Problem (Conjecture Birch Swinnerton-Dyer)

A "method" to determine whether any equation of the type $Y^2 = X^3 + aX + b$ has an infinity of rational solutions

Experimentations

$\Downarrow$

Counterexample,
or increased confidence

$\Downarrow$

Reformulation

$\Downarrow$

...

$\Downarrow$

Solution

Need:

- Database of modular forms
- Compute tables as large as possible

Computing with modular forms

Action of Hecke Operators over the space

- Decomposition into Frobenius invariant factors
- Compute
  - Characteristic polynomials over $\mathbb{Z}_p$
  - Matrix kernel over $\mathbb{Q}$

Computing with modular forms

Action of Hecke Operators over the space

- Decomposition into Frobenius invariant factors
- Compute
  - Characteristic polynomials over $\mathbb{Z}_p$
  - Matrix kernel over $\mathbb{Q}$

“Mathematics is the art of reducing any problem to linear algebra” *W. Stein*

Dense exact linear algebra algorithmic

General considerations:

- No instability
- Variable size

$\Rightarrow$reduce the manipulation of large integer as much as possible

Dense exact linear algebra algorithmic

General considerations:

- No instability
- Variable size

$\Rightarrow$reduce the manipulation of large integer as much as possible

Example: computing the determinant over $\mathbb{Z}$

Method	Complexity
naive Gauss over $\mathbb{Q}$	$\mathcal{O}(\exp(n))$
Gauss mod det	$\mathcal{O}\left(n^6\right)$
Gauss mod p + RNS	$\mathcal{O}\left(n^4\right), \mathcal{O}\left(n^{\omega+1}\right)$
p-adic lifting	$\mathcal{O}\left(n^3\right), \mathcal{O}^{\sim}\left(n^{\omega}\right)$

Dense exact linear algebra algorithmic

Characteristic polynomial over $\mathbb{Z}_p$:

1890 $\mathcal{O}\left(n^4\right)$

1937 $\mathcal{O}\left(n^3\right)$

1985 $\mathcal{O}\left(n^\omega \log n\right)$

2007 $\mathcal{O}\left(n^\omega\right)$

Dense exact linear algebra algorithmic

Characteristic polynomial over $\mathbb{Z}_p$:

1890 $\mathcal{O}\left(n^4\right)$

1937 $\mathcal{O}\left(n^3\right)$

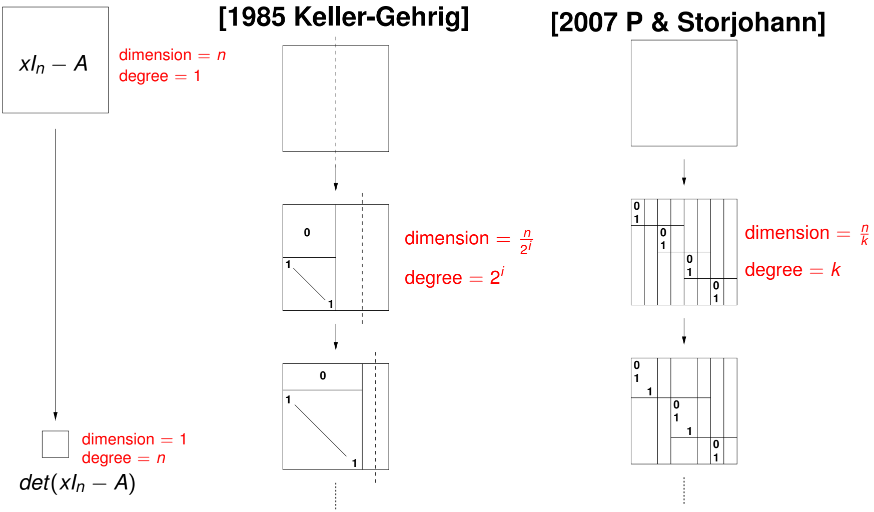
1985 $\mathcal{O}\left(n^\omega \log n\right)$

2007 $\mathcal{O}\left(n^\omega\right)$

$\Rightarrow$ reduction to matrix multiplication

- theoretical interest: asymptotic complexity
- practical interest: extremely optimized

Dense exact linear algebra algorithmic



Matrix multiplication over $\mathbb{Z}_p$: a building block

Principle:

- **Delayed** modular reduction
- Floating point arithmetic (`fma`, SSE2, ...)

$$\lambda(p-1)^2 < 2^M$$

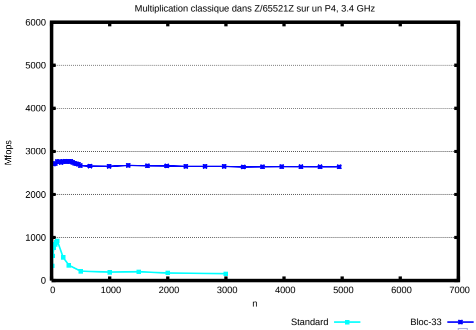
Matrix multiplication over $\mathbb{Z}_p$: a building block

Principle:

- **Delayed** modular reduction
- Floating point arithmetic (`fma`, SSE2, ...)

$$\lambda(p-1)^2 < 2^M$$

- Cache optimization



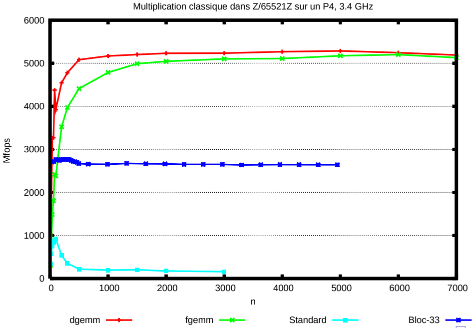
Matrix multiplication over $\mathbb{Z}_p$: a building block

Principle:

- **Delayed** modular reduction
- Floating point arithmetic (`fma`, SSE2, ...)

$$\lambda(p-1)^2 < 2^M$$

- Cache optimization $\Rightarrow$ rely on existing numerical BLAS



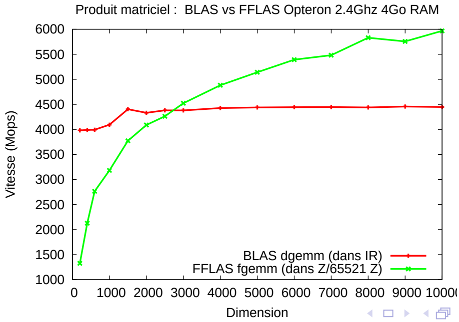
Matrix multiplication over $\mathbb{Z}_p$: a building block

Principle:

- **Delayed** modular reduction
- Floating point arithmetic (`fma`, SSE2, ...)

$$\left(\frac{1+3^l}{2}\right)^2 \left\lceil \frac{\lambda}{2^l} \right\rceil (p-1)^2 < 2^M \text{ for } l \text{ recursive levels}$$

- Cache optimization $\Rightarrow$ rely on existing numerical BLAS
- Sub-cubic algorithm (Strassen-Winograd)



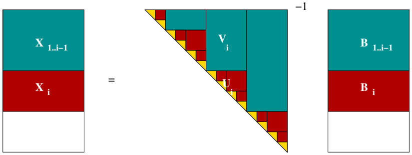
Other dense linear algebra routines

- Reductions to matrix multiplication
- Bounds for the delayed modular reduction

Other dense linear algebra routines

- Reductions to matrix multiplication
- Bounds for the delayed modular reduction

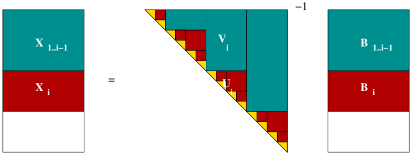
⇒Cascade block algorithms



Other dense linear algebra routines

- Reductions to matrix multiplication
- Bounds for the delayed modular reduction

⇒Cascade block algorithms



	n	*1000*	*2000*	*3000*	*5000*	*10 000*
TRSM	$\frac{ftrsm}{dtrsm}$	1,66	1,33	1,24	1,12	1,01
LQUP	$\frac{lqup}{dgetrf}$	2,00	1,56	1,43	1,18	1,07
INVERSE	$\frac{inverse}{dgetrf+dgetri}$	1.62	1.32	1.15	0.86	0.76

Characteristic polynomial:

n	500	5000	15 000
LinBox	0.91s	4m44s	2h20m
magma-2.13	1.27s	15m32s	7h28m

Plan

Solve exactly

Linear System solving over $\mathbb{Q}$:

p-adic Lifting $\mathcal{O}\left(n^3\right), \mathcal{O}^{\sim}\left(n^{\omega}\right)$ bit complexity

- All the precision for *the same price*
- Regardless of the condition number

In practice:

- applied on stiff problems directly?
- or combined in an iterative refinement when necessary?

Not enough primes

Not enough floating point primes

Probabilistic algorithms require

- to pick randomly a prime in a **large** set
- **large** primes ($> 2n^2$)

Not enough primes

Not enough floating point primes

Probabilistic algorithms require

- to pick randomly a prime in a **large** set
- **large** primes ($> 2n^2$)

Now

- BLAS + floating point arithmetic
  $\Rightarrow$limited to 20ish bit primes.
- only 73586 primes of 20 bits

Not enough primes

Not enough floating point primes

Probabilistic algorithms require

- to pick randomly a prime in a **large** set
- **large** primes ($> 2n^2$)

Now

- BLAS + floating point arithmetic
  $\Rightarrow$limited to 20ish bit primes.
- only 73586 primes of 20 bits

Workarounds?

- RNS based larger prime field implementation
- Fault tolerant algorithms: tolerating failures of probabilistic algorithms

Example: integer minimal polynomial

begin
  for $i = 1..B$ **do**
    Pick a random prime p_i;
    $P_i = \text{MinPoly}(A) \mod p_i$ using Wiedemann alg;
  end
  $P = \text{RNS}(P_1, ..., P_B)$;
end

Problem

How to reconstruct P when some P_i's may be erroneous, using $B + r$ primes.

Arithmetic Error correcting code [MandelBaum76]

- Approx. r-detector, $\lfloor \frac{r}{2} \rfloor$-corrector
- Natural decoding by Extended Euclidean algorithm