Erratum page 23 in the proceedings: this is not Philippe Baveye talk

***Significance of round-off error propagation in two codes used in environmental modeling*! ...**

... but (page 37)

Symbolic Computation: Recent Progress and New Frontiers

Paul Zimmermann

INRIA Lorraine/LORIA

Abstract

Symbolic computation and computer algebra systems are usually known to be either very slow, or memory expensive. However, some specific symbolic computation problems have received in the last years new algorithmic solutions, which enabled to push further the limits of what is doable within a reasonable amount of time and space. Some noticeable examples are polynomial factorisation, lattice reduction, Gröbner basis computation. We will present a few such algorithms, together with a state-of-the-art of what problems computer algebra systems can (or cannot) solve, and for each problem what the current frontiers are.

Computer Algebra Systems

- **general purpose** (Maple, Mathematica, Macsyma, Reduce, MuPAD, Axiom, Derive):

  proprietary, slow, have known pitfalls

- **special purpose** (GNU MP, Pari, Magma, NTL, Linbox):

  free, fast, fewer bugs?

An Example

Cf. Fabienne Jézéquel talk "*Dynamical control of computations of multiple integrals*" (Tuesday, room Durand):

```
> f := exp(x) * cos(z) + sqrt(y) * sinh(z):
> int(int(int(f, x=2..5), y=1..3), z=2..3);
                        1/2                      1/2
2 exp(5) sin(3) + 3 3       exp(-3) + 3 3       exp(3) - 2 exp(2) sin(3)

                                                      1/2
     - exp(-3) - exp(3) - 2 exp(5) sin(2) - 3 3       exp(2)

            1/2
     - 3 3       exp(-2) + 2 exp(2) sin(2) + exp(2) + exp(-2)
```

```
> evalf(%, 50);
           -163.74566743673553320720440934473233062072278553592

> i := Int(Int(Int(f, x=2..5), y=1..3), z=2..3);
             3    3    5
            /    /    /
           |    |    |                 1/2
           |    |    |   exp(x) cos(z) + y    sinh(z) dx dy dz
           |    |    |
          /    /    /
            2    1    2
> st:=time(): evalf(i), time()-st;
                               -163.7456674, 30.790
```

Pitfalls?

```
[zimmerma@ecrouves scan]$ maple
    |\^/|     Maple 7 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) 2001 by Waterloo Maple Inc.
 \  MAPLE  /  All rights reserved. Maple is a registered trademark o
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> Digits:=3:
> a:=1.0: b:=9e-5:
> Rounding := 0: # same with -infinity
> a - b;
                                     1.0
```

You said slow?

```
> s:=numer(orthopoly[L](60, x)):
> st:=time(): readlib(realroot)(s), time()-st;
[[0, 1/8], [1/8, 1/4], [1/4, 1/2], [1/2, 3/4], [3/4, 1], [1, 3/2],
    ...
    [176, 184], [184, 192], [192, 208], [208, 224]], 48.539

[zimmerma@ecrouves scan]$ time ./usp < laguerre60
1: 0/8..1/8
...
60: 208..224
60 real root(s)
0.200u 0.000s 0:00.44 45.4%     0+0k 0+0io 45pf+0w
```

What size of problem can we solve in one minute?

What size of problem can we solve in one minute?

Computer Characteristics: Pentium II 366Mhz under RedHat 7.2, with gcc 2.95.2, 128MB of main memory, and 256MB of swap.

What kind of problems?

Arithmetics: integer multiplication, gcd, modular exponentiation, floating-point and interval computation.

Number Theory: primality testing, integer factorisation.

Univariate Polynomials: factorisation over $\mathbb{F}_p$ and $\mathbb{Q}$, approximation of complex roots, isolation of real roots.

Linear Algebra: minimal polynomial, determinant, lattice reduction.

Moore's law

The computer power increases by a factor of 2 every 18 months.

CONSEQUENCE: Assume we have a computer whose frequency automatically increases following Moore's law. Then a computation that started at $t = -\infty$ until now can be reproduced in ???

Moore's law

The computer power increases by a factor of 2 every 18 months.

CONSEQUENCE: Assume we have a computer whose frequency automatically increases following Moore's law. Then a computation that started at $t = -\infty$ until now can be reproduced in

only 18 months!

Moore's law vs Problem Size

Suppose we can solve in 2002 a problem with 10^{20} bit operations. How large can we solve in 2010, 2100?

Complexity	2002	2010	2100
n	10^{20}	10^{22}	10^{40}
$n \log n$	10^{18}	10^{20}	10^{38}
n^2	10^{10}	10^{11}	10^{20}
n^3	10^7	10^7	10^{13}
n^6	$2,000$	$4,000$	10^7
$e^{cn^{1/3} \log^{2/3} n}$	159	186	681
2^n	66	72	132
2^{2^n}	6.05	6.17	7.04

Arithmetics: Integer Multiplication

Multiplication of integers has complexity $O(n \log n \log \log n)$ (using FFT). Current multiprecision libraries like GNU MP (GMP) can multiply in one minute (and 87MB) two 21M-digit numbers:

```
ecrouves% ./intmul 21000000
x has 21000007 digits
y has 21000007 digits
x*y took 59040ms
```

Arithmetics: Integer GCD

Integer GCD has asymptotic complexity $O(M(n) \log n)$ where $M(n)$ is the multiplication complexity, *i.e.* $O(n \log^2 n \log\log n)$ using FFT. But the constant is so large that most libraries only implement quadratic algorithms.

Worst case for Euclid's algorithm: $\gcd(F_n, F_{n+1})$. Here with GMP:

```
ecrouves% ./intgcd 717745
Fib[717745] has 150000 digits
gcd(Fib[717745], Fib[717746]) took 12790ms

ecrouves% ./intgcd 1435490
Fib[1435490] has 300000 digits
gcd(Fib[1435490], Fib[1435491]) took 56410ms
```

Magma implements a subquadratic algorithm (but gains only a factor of 2):

```
> n := 2870981;
> x := Fibonacci(n);
> Ceiling(Log(x)/Log(10));
600000
> y := Fibonacci(n+1);
> time z := Gcd(x,y);
Time: 56.540
```

Arithmetics: Modular Exponentiation

Basic routine for RSA: compute $x^y \bmod m$ where both x, y and m have n digits. Theoretical asymptotic complexity is $O(nM(n))$, *i.e.* $\approx n^2 \log n$.

```
ecrouves% ./powm 4300
x has 4305 digits
y has 4306 digits
m has 4306 digits
x^y mod m took 58660ms
```

Arithmetics: Floating-Point Computations

Computation of $\exp(\pi)$ with the MPFR library [complexity $O(M(n)\log^2 n)$]:

```
[zimmerma@ecrouves scan]$ ./exp_pi 200000 > /tmp/exp_pi
mpfr_const_pi took 83770ms
mpfr_exp took 60500ms
mpfr_out_str took 4930ms

% cat /tmp/exp_pi
digits=200000 prec=664386
2.31406926327792690057290863679485473802661062426002119934450464
...
9930270981154056341199492439774176843224931910297732e1
```

Arithmetics: Interval Computations

```
> f := exp(x) * cos(z) + sqrt(y) * sinh(z):
> i:=int(int(int(f, x=2..5), y=1..3), z=2..3):
> codegen[optimize]([res=i]);
                                         1/2
t1 = exp(5), t2 = sin(3), t5 = 3   , t6 = exp(-3), t9 = exp(3),

    t12 = exp(2), t15 = sin(2), t20 = exp(-2), res = 2 t1 t2

     + 3 t5 t6 + 3 t5 t9 - 2 t12 t2 - t6 - t9 - 2 t1 t15 - 3 t5 t12

     - 3 t5 t20 + 2 t12 t15 + t12 + t20
```

With MPFI 1.0 (cf. talk of Nathalie Revol), modified to include the sin function:

```
[zimmerma@ecrouves scan]$ time ./ex_mpfi 33000
Using 109627 bits
res=[-1.637456674...8300366317e2,
     -1.637456674...8300366316e2]
58.690u 0.050s 0:58.92 99.6%    0+0k 0+0io 69pf+0w
```

Number Theory

New: PRIME is in P (Agrawal, Kayal, Saxena, Aug. 2002), but complexity is $O(\log^{12} n)$ (proven) or $O(\log^6 n)$ (under conjecture)...

ECPP is believed to be $O(\log^4 n)$. With Morain's ECPP-V6.4.5a:

```
ecrouves% time xrunecpp -f p290 -C p290.cert
Working on 1415132764414474869524500144421965690639918680434754208024734\
3365460383949194096118887526376260141289807260452962259273084893220429 14\
836340003691320909724764365042424859088382828895499502881883260433667686\
909076137633236197675088250191255604277652097861010838659146460381197447\
3290309003941
...
% N_56=1061
% Proofs: [0][1][2][3][4][5][6][7][8][9][10][11][12][13][14][15]...
% Total time is        58.820000 s
This number is prime
```

Checking the certificate

```
ecrouves% time xcheckcertif -f p290.cert
  0:     43 y       0.60s
...
 57:     -1 y       0.00s

% Total time is       7.72 s
```

```
ecrouves% ./xcheckcertif -f /tmp/cert -v # 572 digits
% Each line has the following shape:
%
% i: D y/n time
% where i is the rank of the certificate,
% D the discriminant
% y if the certificate is correct and n otherwise,
% time is the time needed
  0:    872 y       3.59s
  1:    643 y       3.40s
  2:   1659 y       3.18s
...
 75:     -1 y       0.00s

% Total time is      58.73 s
```

Integer Factorisation

Factor is $O(\exp^{c(\log n)^{1/3}(\log\log n)^{2/3}})$ (Number Field Sieve, Pollard 1988) where the best known $c = \frac{(92+26\sqrt{13})^{1/3}}{3} \approx 1.902$:

```
[zimmerma@ecrouves scan]$ gp
               GP/PARI CALCULATOR Version 2.1.4 (released)
             i686 running linux (ix86 kernel) 32-bit version
             (readline v4.2 enabled, extended help available)
? n=885684319731911762861578039559042994165374347516240073;
? factorint(n,14)
...
time = 1mn, 1,370 ms.
%7 =
[812628396211489583584572751 1]

[1089900776124748047286366823 1]
```

Polynomials: Finding Roots over $\mathbb{F}_p$

Over $\mathbb{F}_p$, the basic operation is $x^p \bmod f$. Von zur Gathen challenge: sparse polynomials $x^n + x + 1$ modulo $p_n = \mathrm{nextprime}(\lfloor \pi 2^n \rfloor)$. With NTL, which implements a $O(n^{1/2} M^2(n))$ algorithm:

```
[zimmerma@ecrouves scan]$ ./findroots 300
[44345865210347378455039303391189968223765785066\
   52416409072300108508446099286502968211870461
30758603506275503746589713211169440873493694454 6\
   45258674074103901708973359612323901854790 61
92964876706540865620696382535550171675005618279 3\
   99524218260671312125405000812443564149310 6
30510953905334047895910560391997620336661238326 8\
   08317192405330107151222925263879548622548 65] 61.2
```

Magma faster for sparse poly.: 34s for $n = 300$, 59s for $n = 450$.

Shoup: dense polynomials $F_n = \sum a_i x^{n-i}$ modulo $p_n = \mathrm{nextprime}(\lfloor \pi 2^{n-2} \rfloor)$ (JSC 1995), where $a_0 = 1, a_{i+1} = a_i^2 + 1$:

```
[zimmerma@ecrouves scan]$ ./findroots2 298
[2165715194265596704942858035325630055569013748l\
   798967577768513583081762652641723649497420] 62.02
```

Factoring over small fields

Factoring F_n mod 3 where F_n is Shoup's polynomial, with Magma [$O(nM(n))$]:

```
> time Shoup_mod3(4000);
[ 1, 3, 30, 201, 611, 645, 1083, 1426 ]
Time: 58.920
```

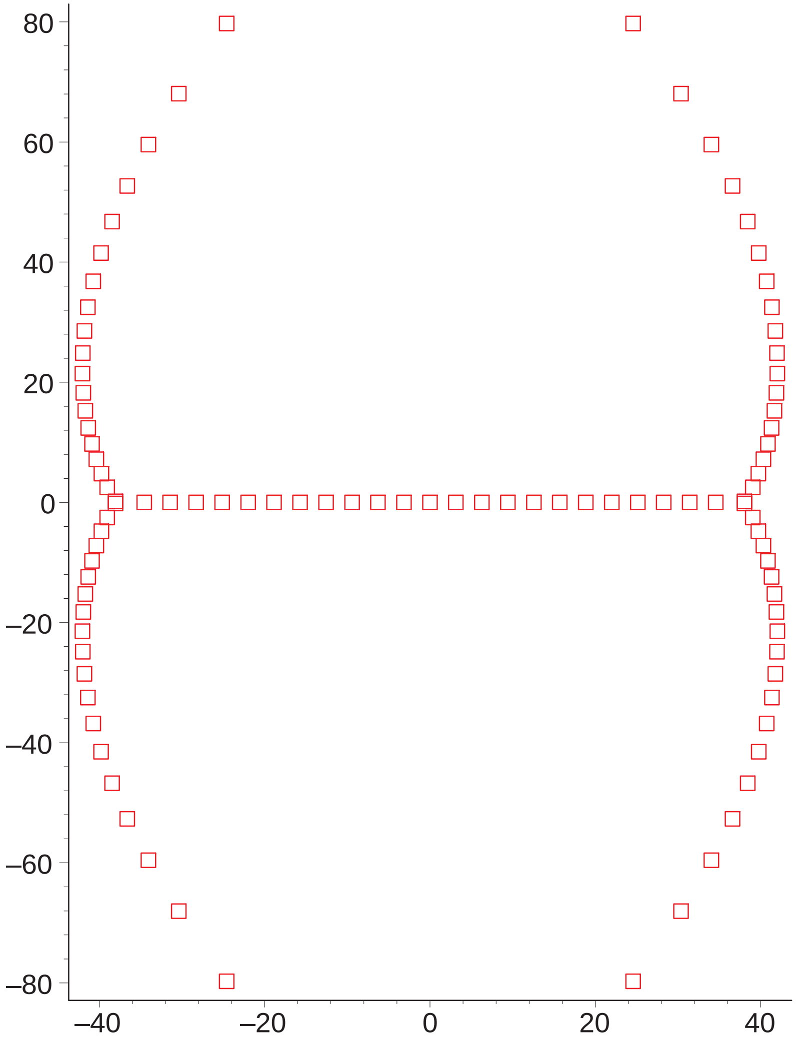
(NTL is much slower: 60s for $n = 1200$, and Pari/GP lies between: 17s for $n = 1200$.)

Factoring over $GF(2)$

Squaring a polynomial takes linear time! Factoring $x^n + x + 1$ takes $\tilde{O}(n^2)$. With Magma (NTL needs 400s):

```
> time VZG_mod2(21000);
[ 5, 22, 28, 31, 72, 156, 1510, 1855, 6898, 10423 ]
Time: 60.110
```

Complex Roots



Let S_n be the truncated Taylor expansion of order n of $\sin x$ at $x = 0$.

Approximating all complex roots of S_n with MPSolve (Bini, Fiorentino):

```
[zimmerma@ecrouves MPSolve2.0]$ time ./unisolve < sin317
(0, 0)
(-0.130007172520032279e3, -0.667510698581253538e2)
...
(0.130007172520032279e3, -0.667510698581253538e2)
60.110u 0.040s 1:00.80 98.9%    0+0k 0+0io 188pf+0w
```

Isolating Real Roots

Uspensky's algorithm based on Descartes' rule (Rouillier, Z., 2002, $O(n^3)$ conjectured):

```
[zimmerma@ecrouves scan]$ time ./usp < sin449
initial interval is -0.1024e4..0.1024e4
1: 0..0
2: -168..-167
...
109: 167..168
109 real root(s)
58.240u 0.010s 0:58.81 99.0%    0+0k 0+0io 47pf+0w
```

Polynomial Factoring

First polynomial algorithm: Lenstra, Lenstra, Lovácz (1982), but impractical [$\mathcal{O}(n^{10})$], even of worst cases for Zassenhaus' exponential algorithm!

van Hoeij (2000): new polynomial algorithm based on LLL, very practical, implemented in NTL.

Factoring $L_n L_{n+1}$ with NTL

$$L_0 = 1, \quad L_1 = 1 - x, \quad nL_n = (2n - 1 - x)L_{n-1} - (n-1)L_{n-2}$$

```
[zimmerma@ecrouves src]$ ./ZZXFacTest < L205_L206
...
*** end SFFactor.  degree sequence:
206 205
total time: 59.38
```

Linear Algebra

Arne Storjohann (ISSAC'02) proved that $A^{-1}b$ and $\det(A)$ over $K[x]^{n\times n}$ have arithmetic complexity $\tilde{O}(n^\theta \deg A)$ (Las Vegas model), using "High-Order Lifting".

Seems to extend well to the integer case: $\tilde{O}(n^\theta)$ bit complexity?

Trefethen's Challenge

Let A be a $n \times n$ sparse matrix with $A_{i,i}$ the ith prime, and $A_{i,j} = 1$ for $|i - j| = 2^k$.

$$
\begin{bmatrix}
2 & 1 & 1 & & 1 & & & & 1 & & & & & & \\
1 & 3 & 1 & 1 & & 1 & & & & 1 & & & & & \\
1 & 1 & 5 & 1 & 1 & & 1 & & & & 1 & & & & \\
 & 1 & 1 & 7 & 1 & 1 & & 1 & & & & 1 & & & \\
1 & & 1 & 1 & 11 & 1 & 1 & & 1 & & & & 1 & & \\
 & 1 & & 1 & 1 & 13 & 1 & 1 & & 1 & & & & 1 & \\
 & & 1 & & 1 & 1 & 17 & 1 & 1 & & 1 & & & & 1 \\
 & & & 1 & & 1 & 1 & 19 & 1 & 1 & & 1 & & & \\
1 & & & & 1 & & 1 & 1 & 23 & 1 & 1 & & 1 & & \\
 & 1 & & & & 1 & & 1 & 1 & 29 & 1 & 1 & & 1 & \\
 & & 1 & & & & 1 & & 1 & 1 & 31 & 1 & 1 & & 1 \\
 & & & 1 & & & & 1 & & 1 & 1 & 37 & 1 & 1 & \\
 & & & & 1 & & & & 1 & & 1 & 1 & 41 & 1 & 1 \\
 & & & & & 1 & & & & 1 & & 1 & 1 & 43 & 1 \\
 & & & & & & 1 & & & & 1 & & 1 & 1 & 47
\end{bmatrix}
$$

The Challenge: for $n = 20000$, compute $B = A^{-1}$, then $B_{1,1}$ (rational with numerator/denominator of $100,000$ digits each). Solved by LinBox in about 2 years of CPU time.

```
[zimmerma@ecrouves blackbox]$ time ./load-det 2500 test.matrix
Massey...Done
Determinant is 665 (mod 65521)
58.180u 0.060s 0:58.30 99.8%    0+0k 0+0io 210pf+0w
```

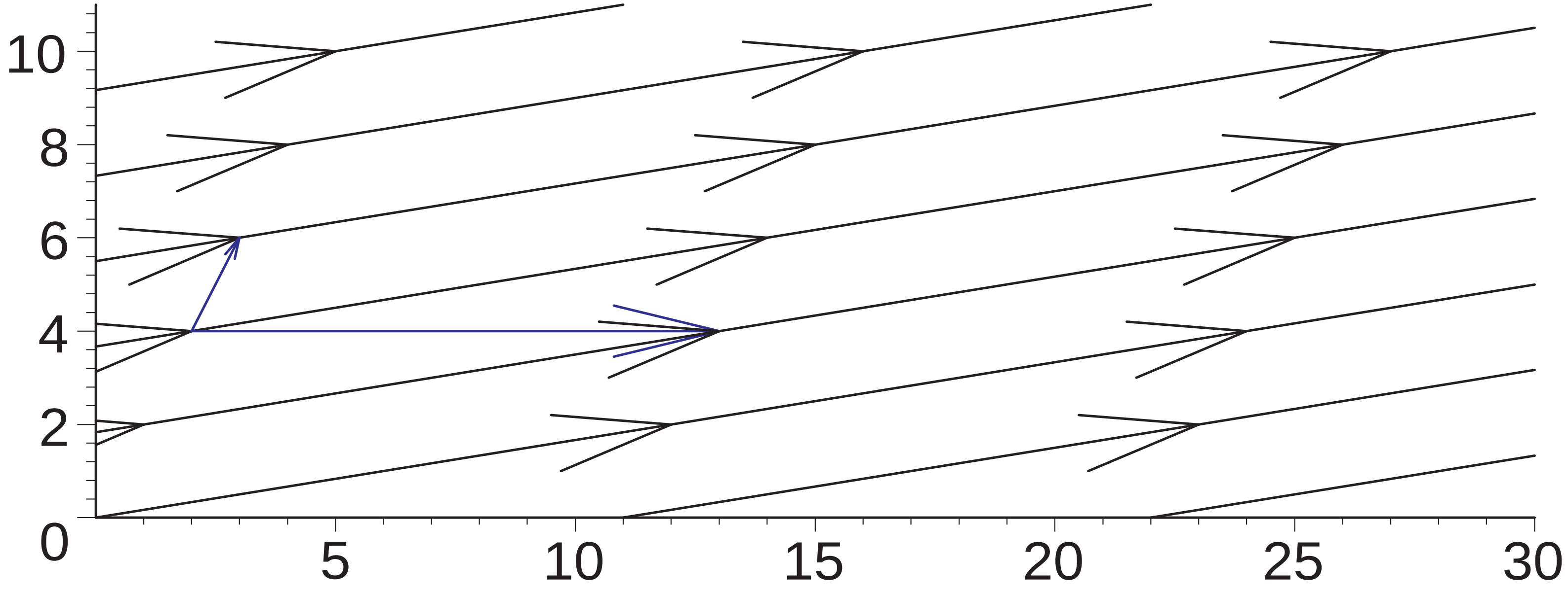
Minimal Polynomial

Compute the minimal polynomial of A mod 65521 (Linbox using black box algorithms):

```
$ time linbox-0.1.2/examples/blackbox/example 3300
Minimal polynomial...done
58.730u 0.030s 0:58.79 99.9%    0+0k 0+0io 194pf+0w
```

Lattice reduction

Problem: given a set of n vectors in $\mathbb{Z}^n$, find a basis of short vectors spanning the same space



Complexity: $\mathcal{O}(n^5 \log^2 A)$ (theoretical) $\mathcal{O}(n^4 \log^2 A)$ (experimental)

New developments: Koy and Schnorr (2002): saves a factor n.

Applications

Knapsack problem: find $\sum_{i=1}^{n} \epsilon_i b_i = S \bmod M$.

```
[zimmerma@ecrouves scan]$ time ./lll 350 < in
...
norm(B[1])=2.000000e+00
smallest norm=2.000000e+00
largest norm=1.100000e+01
59.910u 0.170s 1:02.19 96.6%    0+0k 0+0io 47pf+0w
```

Summary

Problem	1 minute	record
IntMul	21M dig.	$\approx 10^{10}$
IntGcd	600K dig.	$\approx 10^{8}$
Float	200K dig.	$\approx 10^{7}$
PowMod	4300 dig.	$\approx 10^{5}$
IntPrime	290 dig.	5020 dig. (13 weeks, 1.3Ghz Athlon)
IntCheckPrime	572 dig.	$\approx 10^{5}$ dig.
IntFactor	54 dig.	158 dig. (52 months 800Mhz Athlon)

Problem	1 minute	record
RootMod (sparse)	450	$\approx 10^4$
RootMod (dense)	300	$\approx 10^4$
RootMod (dense, GF(3))	4000	$\approx 10^5$
RootMod (dense, GF(2))	21000	$\approx 10^7$
ComplexRoots	317	$\approx 10^4$
RealRoots	449	$\approx 10^4$
Factor	411	$\approx 10^4$
LinDep (sparse, $\mathbb{F}_2$)		$\approx 10^7$
Det (sparse, $\mathbb{F}_{65521}$)	2500	$\approx 10^5$
MinPoly (sparse, $\mathbb{F}_{65521}$)	3300	$\approx 10^5$
LLL	350	10^3

Conclusion

- computer algebra tools can tackle large problems

Conclusion

- computer algebra tools can tackle large problems
- ... but you have to use special-purpose software

Conclusion

- computer algebra tools can tackle large problems
- ... but you have to use special-purpose software
- ... and choice of software is crucial!

Conclusion

- computer algebra tools can tackle large problems
- ... but you have to use special-purpose software
- ... and choice of software is crucial!
- for several problems, the new frontier is memory

Many thanks to the following software (and their authors): GNU MP 4.1, Pari-2.1.4 (and 2.2.4 alpha), Magma V2.8-3, NTL 5.3, Linbox 0.1.2, ECPP-V6.4.5a, MPFR 2.0.1, MPFI 1.0.