[I]

# A brief history of exact linear algebra with LinBox

**David Saunders - U of Delaware**

**July 28, 2005**

Abstract:
We present a review of the developments in algorithms and implementation techniques which have brought us to a state where integer and rational matrix problems can be solved for a remarkable range of types and sizes of matrix. Among the problems for which there is now sophisticated high performance software are linear system solution, rank, determinant, Smith Normal Form, minimal and characteristic polynomials.

. . .

[I]NSF grants 0098284 and 0112807

## Outline

- Algorithmic developments for dense matrices.
- Algorithmic developments for sparse and structured matrices (blackbox methods).
- survey of applications.
- predictions and open problems.

## The software package LinBox

- How it came into being
- What it can do
- How it does it
  - What algorithms
  - What software design techniques
- Where it may evolve in the future
- What can it do for you?

## Some important other examples

- Linear algebra in Magma: particularly the "Meat Axe" null space basis method.
- Linear algebra in Maple and Mathematica, the whole range of computations. Dense methods, variety of representations (to save memory). Emphasis on capability over performance. Small problems done well in very general setting.
- The Maple/NAG interface giving floating point numeric computation with arbitrary but fixed precision.
- Sparse linear algebra: Montgomery's implementation of Block Lanczos algorithm for finding null space vector of huge sparse matrix over GF(2). Used in cryptography, breaking an RSA key.
- Faugère's F5 applied to polynomial systems, Grobner bases.

LinBox may make these claims:

- It is the only systematic implementation of blackbox methods for sparse and structured systems.
- It is an important example of BLAS applied to dense linear algebra over finite fields.
- It is well organized to serve as "middleware" between tools (BLAS, GMP, NTL, Givaro) and applications (Gap, Maple, LieAtlas, Singular)

LinBox goal: High performance methods for integer and rational matrix problems, yet generic for (sparse and dense) matrices over arbitrary finite fields and rings (primarily because we need this for the modular methods applied to integer and rational problems)

## 70s - Era of algebraic complexity

Algebraic complexity - count field operations.

- good for matrix of floating point. (Coincides with development of numeric linear algebra)
- good for matrix over small finite field.
- bad for integer or rational matrix.

Let S(n) = algebraic complexity to solve an $n \times n$ system.

Let M(n) = algebraic complexity to multiply $n \times n$ matrices.

Known: $O(n^2) \subseteq O(S(n)) \subseteq O(M(n)) \subset O(n^3)$ [Strassen 69, Pan 78]

Open: $O(n^2) \neq O(S(n))$? $O(S(n)) \neq O(M(n))$?

Note: Dense matrices. Memory used is essentially fixed at $O(n^2)$.

## 80's - Facing the bit complexity

The problem:
Given $A, b$, integer $n \times n$ matrix and $n$-vector, find rational vector $x : Ax = b$.

Suppose entries in A, b are bounded by $\beta$. Let $d = \log(\beta)$. The input size is $O(n^2 d + nd)$.

Det(A) is bounded by $(n^{1/2}\beta)^n$, storage size by $n(\log(n)/2 + d)$. By Cramer's rule, the output involves $n$ determinants as numerator and $\det(A)$ as denominator, Size of solution vector $x$ is $O(n^2(\log(n) + d)$, greater than size of $A, b$ in general!

---

Simplify: Let $\beta < n$, so $d < \log(n)$. Then, simply, let us ignore log factors.

Thus: When size of $A, b$ is $\mathrm{O}^{\sim}(n^2)$, size of $x$ is $\mathrm{O}^{\sim}(n^2)$.

## Direct Elimination Cost

During Gaussian elimination we compute about $(1/3)n^3$ quotients of minors. Sizes averaging $\mathrm{O}^{\sim}(n)$.

Thus: The size of intermediate storage is $\mathrm{O}^{\sim}(n^3)$. The time cost with standard integer arithmetic is $\mathrm{O}^{\sim}(n^5)$.

For example:

$10000 \times 10000$ matrix of $\{0, 1, -1\}$. Initial memory: 100 Megabytes.

Intermediate memory need: $10^{12}$ bytes $=$ 1 Terabyte.

Run time: $10^{20}$ cycles $=$ 300 years at 10 GHz speed.

## [Dixon 82]

Given $n \times n$ matrix $A$, vector $b$ over $\boldsymbol{Z}$, solve $Ax = b$. Choose a prime $p$ near $2^{32}$ (wordsize prime)

Compute $LU$ mod $p$. $O(n^3)$. Set $x_0 = U^{-1}L^{-1}b \pmod p$. Set $r_1 = (Ax_0 - b)/p$. $O(n^2)$

```
[ Hensel lifting - base p expansion of x ]
for $i = 1$ to $n\log_p(n)$ [Hadamard bound] do:
    Set $x_i = U^{-1}L^{-1}r_i \pmod p$. $O(n^2)$
    Set $r_{i+1} = (Ax - b)/p$. (Can be done in $O(n^2)$)
```

Thus: bit complexity of $\mathrm{O}^{\sim}(n^3)$, memory $\mathrm{O}^{\sim}(n^2)$.

$$\begin{array}{rrrr} Ax_0 = b(\text{mod } p), & A(x_0 + x_1 p) = b(\text{mod } p^2), & A(x_0 + x_1 p + x_2 p^2) = b(\text{mod } p^3), & \ldots \\ n^3, & n^2, & n^2, & \ldots \end{array}$$

Dixon's method Bit complexity of $O^{\sim}(n^3)$, memory $O^{\sim}(n^2)$: No worse than classic algebraic cost!

For example:

$10000 \times 10000$ matrix of $\{0, 1, -1\}$. Initial memory: 100 Megabytes.

Intermediate memory need: $10^8$ bytes = 100 Megabytes.

Run time: $10^{12}$ cycles = 17 minutes at 1 GHz speed.

Further work: output sensitive method. Early termination.

## Dense Matrix Rational Solving

| order | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 |
|---|---|---|---|---|---|---|---|---|
| Bottom up | 0.91 | 7.77 | 29.2 | 78.38 | 158.85 | 298.81 | 504.87 | 823.06 |
| Bottom up (use BLAS) | 0.11 | 0.60 | 1.61 | 3.40 | 6.12 | 10.09 | 15.15 | 21.49 |
| Top down | 0.03 | 0.20 | 0.74 | 1.84 | 3.6 | 6.03 | 9.64 | 14.31 |

All entries are randomly and independently chosen from $[-2^{20}, 2^{20}]$.

1. The *Bottom up* is implementation of Dixon lifting without calling BLAS in LinBox.
2. The *Bottom up (use BLAS)* is implemented by Zhuliang Chen the idea of FFLAS and mixture of Dixon lifting and the Chinese remainder algorithm.
3. The *Top down* is implemented by us using hybrid numeric/symbolic solver (in prep. for publication).

## Smith Form History

$n \times n$ dense matrices with constant size entries, using standard Mat Mul.

| Author(s) | Year | Time Cost | Method |
|---|---|---|---|
| Smith | 1861 | UNK | D |
| Kannan and Bachem | 1979 | Polynomial | D |
| Iliopoulos | 1989 | $O^{\sim}(n^5)$ | D, mod det |
| Hafner and McCurley | 1991 | $O^{\sim}(n^5)$ | D, mod det |
| Giesbrecht | 1995 | $O^{\sim}(n^4)$ | P, mod det |
| Storjohann | 1996 | $O^{\sim}(n^4)$ | D, mod det |
| Eberly, Giesbrecht and Villard | 2000 | $O^{\sim}(n^{3.5})$ | P, mod $s_n$ |
| Kaltofen and Villard | 2003 | $O^{\sim}(n^{3.33})$ | P, mod $D_n$ |
| Local at a prime p | Folklore | $O^{\sim}(n^3)$ | D, mod $p^e$ |
| Saunders, Wan | 2004 | $O^{\sim}(n^{3.5})$ | P, smooth/rough |

Last invariant factor: Pan 88, Abbot Bronstein Mulders 1999, Mulders Storjohann 1999

## Matrix with one million entries

$1000 \times 1000$, **dense**

$200 \times 5000$, dense

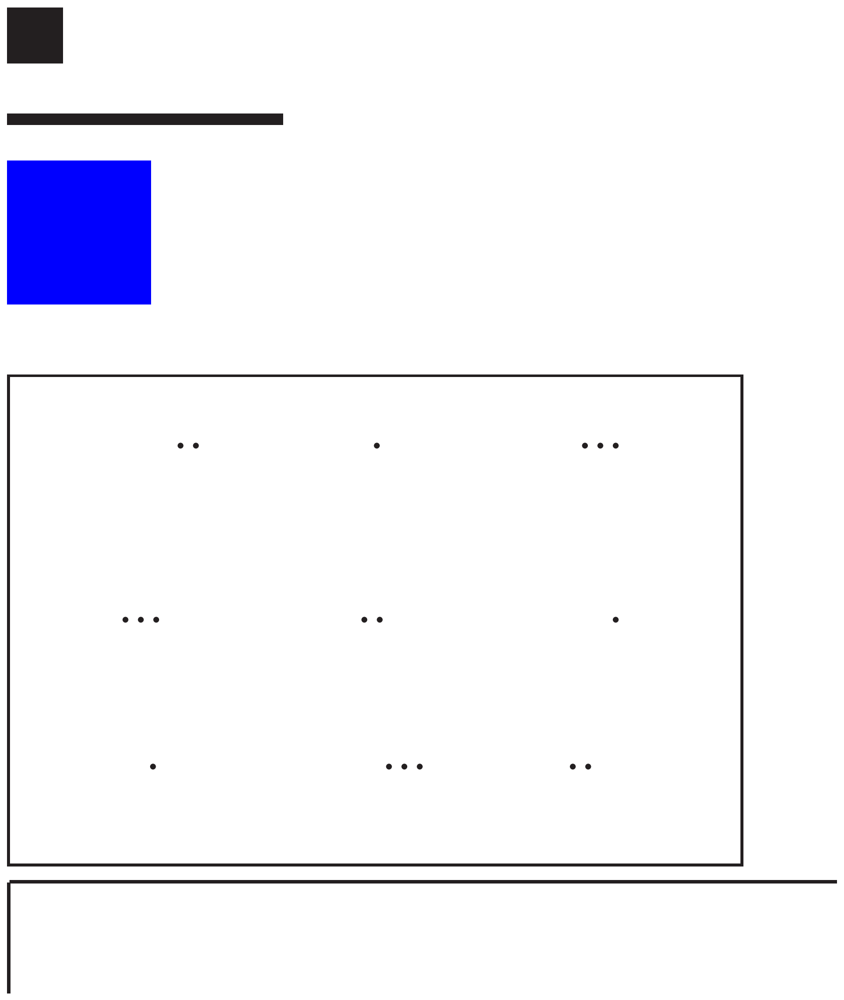
$2000 \times 2000$, $1/4$ *dense*

$10000 \times 12000$

100 per row, *sparse*

$100000 \times 100000$

10 per row, **sparse**



## measure sparsity as power of dimension

$n \times n$ matrix $\Rightarrow n^e$ nonzero entries

$n = 1000 \Rightarrow 10^6 = n^2$

$n = 2000 \Rightarrow 10^6 = n^{1.82}$

$n = 10000 \Rightarrow 10^6 = n^{1.5}$

$n = 100000 \Rightarrow 10^6 = n^{1.2}$

Gaussian elimination - Dense matrix: $n^3 = n * n^e$ arithmetic ops. $(e = 2)$
Gaussian elimination - Sparse matrix: $n \le \text{ops} \le n^3$

Time is anywhere from much less than $n * n^e$ to much more.
Memory need is anywhere from $n^e$ to $n^2 = n * n^e$

For sparse matrices, asymptotic analysis is much less helpful for understanding algorithm performance in practice.

## 90's - Era of Blackbox algorithms

the *minimal polynomial* of a matrix is the lowest degree monic polynomial $m(x) = \sum_i m_i x^i$, such that $m(A) = \sum_i m_i A^i = 0$.

Matrix *sequence* viewpoint (over $n^2$ dimensional vector space):

$$\begin{array}{cccccccc}
I, & A, & A^2, & A^3, & \ldots, & A^f, & A^{f+1}, & \ldots \\
m_0, & m_1, & m_2, & m_3, & \ldots, & m_f & & \\
 & m_0, & m_1, & m_2, & m_3, & \ldots, & m_f &
\end{array}$$

$m_A(A) = 0.$

Lanczos: *Vector* sequence viewpoint (over $n$ dimensional space)

$$\begin{array}{cccccccc} b, & Ab, & A^2b, & A^3b, & \ldots, & A^eb, & A^{e+1}b, & \ldots \\ m_0, & m_1, & m_2, & m_3, & \ldots, & m_e & & \\ & m_0, & m_1, & m_2, & m_3, & \ldots, & m_e & \end{array}$$

$m_{A,b}(A)b = \sum_i m_i A^i b = 0$ (and shifted: $\sum_i m_i A^{i+k} b = 0$)

Wiedemann: **Scalar** sequence viewpoint (over 1 dimensional space)

$$\begin{array}{cccccccc} u^Tb, & u^TAb, & u^TA^2b, & u^TA^3b, & \ldots, & u^TA^db, & u^TA^{d+1}b, & \ldots \\ m_0, & m_1, & m_2, & m_3, & \ldots, & m_d & & \\ & m_0, & m_1, & m_2, & m_3, & \ldots, & m_d & \end{array}$$

$u^T m_{u,A,b}(A)b = \sum_i u^T m_i A^i b = 0$ (and shifted)

$d \le e \le f$

## Wiedemann's method

Wiedemann's method is to *sparse* matrices as Gaussian elimination is to *dense* matrices.

**Berlekamp/Massey**: if scalar sequence satisfies a linear recurrence of degree $n$ or less, the minimal polynomial may be found from the first $2n$ sequence entries in time $O(n^2)$.

[**Wiedemann 88**]:

1. If vector $u$ is chosen at random, $m_{A,b} = m_{u,A,b}$ with high probability.

2. If vector $b$ is also chosen at random, $m_A = m_{A,b} = m_{u,A,b}$ with high probability.

**Consequences**: (a) Solve nonsingular system:

$$\begin{aligned} 0 &= m_{A,b}(A)b = m_0 b + m_1 Ab + \ldots + m_e A^e b \\ &= m_0 b + A(m_1 b + \ldots + m_e A^{e-1} b) \\ &\downarrow \\ b &= A(-1/m_0)(m_1 b + \ldots + m_e A^{e-1} b) \end{aligned}$$

(b) Compute minimal polynomial $m_A$ in $O(n^{1+e})$ for matrix with $O(n^e)$ nonzeros.

$$m_A(x) = (x-2)(x-3) = 6 - 5x + x^2.$$

$$\begin{array}{ccccc}
\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix} & \begin{pmatrix} 4 & 0 \\ 5 & 9 \end{pmatrix} & \begin{pmatrix} 8 & 0 \\ 19 & 27 \end{pmatrix} & \begin{pmatrix} 16 & 0 \\ 65 & 81 \end{pmatrix} \\
6 & -5 & 1 & & \\
 & 6 & -5 & 1 &
\end{array}$$

$$\begin{array}{ccccc}
\begin{pmatrix} 1 \\ 1 \end{pmatrix} & \begin{pmatrix} 2 \\ 4 \end{pmatrix} & \begin{pmatrix} 4 \\ 14 \end{pmatrix} & \begin{pmatrix} 8 \\ 46 \end{pmatrix} & \begin{pmatrix} 16 \\ 146 \end{pmatrix} \\
6 & -5 & 1 & & \\
 & 6 & -5 & 1 &
\end{array}$$

$$u = \begin{pmatrix} 1 & 1 \end{pmatrix}.$$

$$\begin{array}{ccccc}
(2) & (6) & (18) & (54) & (162) \\
6 & -5 & 1 & & \\
 & 6 & -5 & 1 & \\
 & & 6 & -5 & 1
\end{array}$$

$u = (1 \ \ 1)$.

$$\begin{array}{ccccc}
(2) & (6) & (18) & (54) & (162) \\
-3 & 1 & & & \\
 & -3 & 1 & & \\
 & & -3 & 1 &
\end{array}$$

$u = (1 \ \ 0)$.

$$\begin{array}{ccccc}
(1) & (2) & (4) & (8) & (16) \\
-2 & 1 & & & \\
 & -2 & 1 & & \\
 & & -2 & 1 &
\end{array}$$

But $u = (0 \ \ 1)$ works:

$$\begin{array}{ccccc}
(1) & (4) & (14) & (46) & (146) \\
-4 & 1 & & & \\
6 & -5 & 1 & & \\
 & 6 & -5 & 1 &
\end{array}$$

## Preconditioners

Let's use Wiedemann's minpoly algorithm to get the determinant. (This is also an example of a Las Vegas algorithm arising from Monte Carlo method).

1. $w_{u,A,b} | l_{A,b} | m_A$. [always]
2. if $w_{u,A,b}(0) = 0$, then $\det(A) = 0$. [always]
3. if $\deg(w_{u,A,b}) = n$, then $\det(A) = w_{u,A,b}(0)$. [always]
4. $\det(A) = \det(AB)/\det(B)$. [always]
5. If $A$ is nonsingular, vectors u, v, and matrix B are random variables, then $\deg(w_{u,AB,b}) = n$. [with high probability]

## More preconditioners

If $A$ is nonsingular, then with high probability, $\det(A) =$ constant coefficient of $w_{u,AB,v}$ for random $u, v, B$.

...when B is a Benes network matrix [Wiedemann 88].

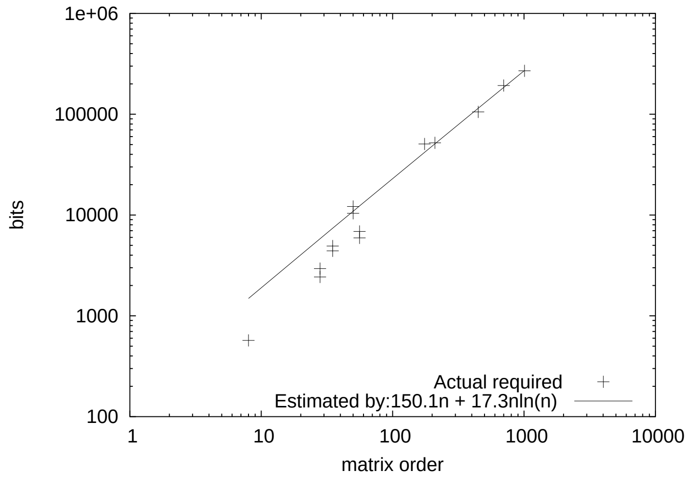
...when B is a Toeplitz matrix [Kaltofen, S 91].

...when B is a Butterfly matrix [Turner -]

...when B is a Sparse matrix [Wiedemann 88, Villard -]

...even when B is diagonal [Chen, Eberly, Kaltofen, S, Turner, Villard 02].

We may compute $m_A$, $c_A$, or $M$ by the Chinese remainder algorithm, and count the sign alternations. In this application thousands of word sized primes are needed.



We abstract the CRA process for convenience, separation of concerns, efficiency, and for parallel implementation.

```
class CRA {
// E, the early termination threshold, is the minimum number of
stable steps.
// L is the minimum total number of steps.
void initialize(int _E, int _L) : E(_E), L(_L) {};

// check termination condition
bool terminated()
{ return stableSteps ≥ E && totalSteps ≥ L;}

// add to the data the residues for one new modulus.
void progress (integer modulus, Vector residues) ;

// result residues mod the lcm of the moduli.
Vector result () ; };
```

## combine in a loop – parallelizable

```
template<class Function, class RandPrime>
Vector loop (Function residues, RandPrime genprime) {
   integer p;
   while( ! terminated() ) {
       genprime.randomPrime(p);
       progress(p, residues(p) );     }
    return result();
}
```

Also, can we can encapsulate a fast early termination strategy...

## Goal

$T$: time computing the characteristic polynomial at each $d$ bit prime. $N_1$: the number of $d$ bit primes required without early termination. $N_2$: the number of $d$ bit primes required with early termination. $N_2 \leq \mathbf{N_1}$.

No early termination + divide and conquer

$$\boxed{\mathrm{O}(T * \mathbf{N_1})} + \overbrace{\boxed{\mathrm{O}^{\sim}(d * \mathbf{N_1})} \cdots \boxed{\mathrm{O}^{\sim}(d * \mathbf{N_1})}}^{n}$$

Early termination + incremental

$$\boxed{\mathrm{O}(T * N_2)} + \overbrace{\boxed{\mathrm{O}^{\sim}((d * N_2)^{\mathbf{2}})} \cdots \boxed{\mathrm{O}^{\sim}((d * N_2)^{\mathbf{2}})}}^{n}$$

GOAL: Early termination + divide and conquer

$$\boxed{\mathrm{O}(T * N_2)} + \overbrace{\boxed{\mathrm{O}^{\sim}(d * N_2)} \cdots \boxed{\mathrm{O}^{\sim}(d * N_2)}}^{n}$$

## Fast Early Termination

A certificate, a random linear combination of residues is used to detect when the early termination happens.

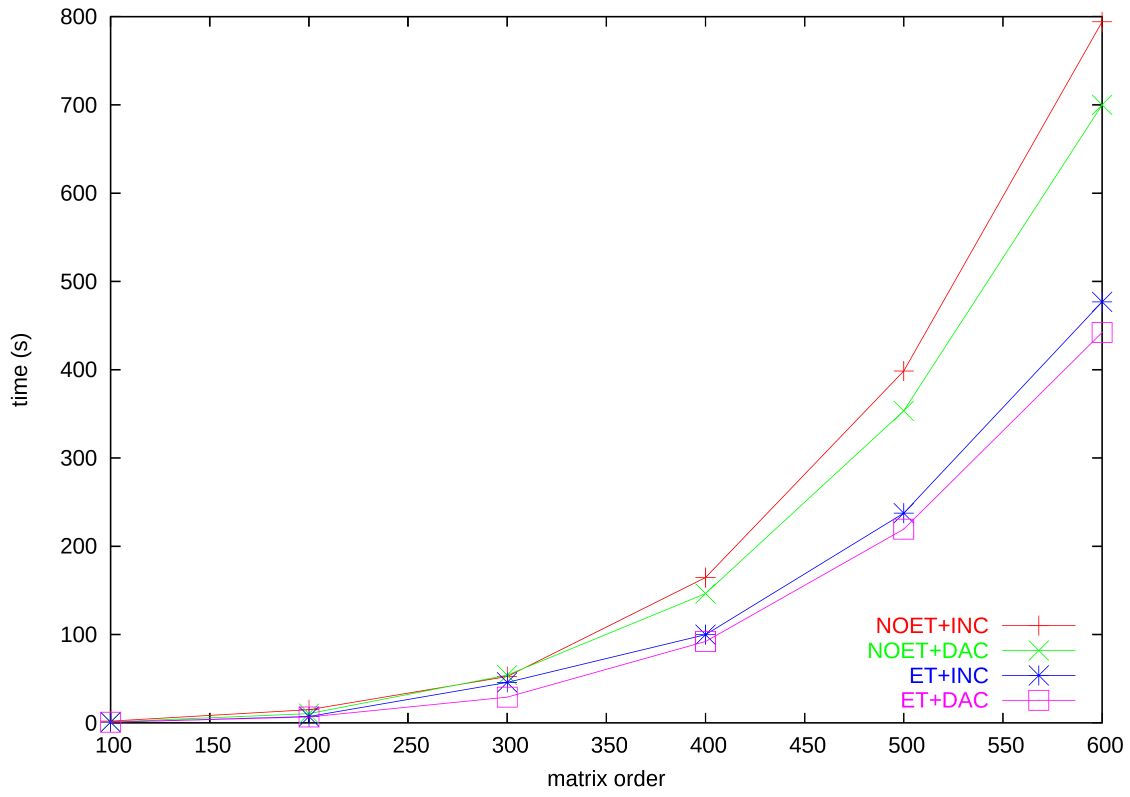
Solution: Early termination + divide and conquer + certificate

$$\boxed{\mathrm{O}(T * N_3)} + \overbrace{\boxed{\mathrm{O}^{\sim}(d * N_3)} \cdots \boxed{\mathrm{O}^{\sim}(d * N_3)}}^{n} + \boxed{\mathrm{O}^{\sim}((d * N_3)^{\mathbf{2}})}$$

Note: $N_3 \approx N_2$. $N_3$ may be slightly larger than $N_2$. because the certificate, a sum, can be as large as $nrV$ where $n$ is the number of summands, $r$ bounds the random coefficients and $V$ is the largest of the values being computed.

## Experimentation

We tested with “random” matrices. Each one is the product of a unit random lower triangle matrix, a random diagonal matrix, and a unit upper triangle matrix. And each entry is independently and uniformly chosen from $[0, 99]$.



## the small prime problem

...if field is large enough.

If diagonal preconditioner used, suppose

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \text{ then } AB = \begin{pmatrix} b_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & b_3 \end{pmatrix},$$

for random diagonal matrix $B$.

Over GF(2) it is not possible that the diagonal entries are distinct. Hence $\deg(m_{AB}) < 3$. A fortiori, $\deg(w_{u,AB,v}) < 3$.

But an an extension field is large enough. May use Zech log arithmetic in nonprime field for speed. [Dumas 2002]

**Next: Examples**

## Welker's homology boundary matrices

Problem: Homology of simplicial complexes in dimensions up to about 10.

Specific computation: Smith Normal form of $\{0, 1, -1\}$-boundary matrices.

Solution: Jean-Guillaume Dumas code (Gap package[Dumas, Heckenback, S, Welker 2003]/LinBox) for sparse matrix Smith form. [Dumas, S, Villard 2001.]

Example: $135135 \times 270270$ matrix, 5 entries per row ($n^{1.14}$). Smith form: 133991 one's, 220 three's, 924 zeroes. Time: 4 days.

## Krattenthaler: "Combinatorialists love determinants"

Favorite theorem: the number of $< \ldots >$ of size $n$ is NICE($n$).

NICE formula is (roughly) hypergeometric. For many $< \ldots >$ the $n$-th instance is a determinant. The nice formula arises if the determinant involves only small primes.

Example problem: Conjectured formula

$$\det(q^{\mathrm{maj}_B(\sigma\pi^{-1})}) = \prod_{i=1}^{n}(1-q^{2i})^{e(i)} \prod_{i=2}^{n}(1-q^{i})^{f(i)}$$

Done by Macsyma: $n = 1, 2, 3, 4$(hard) Done by LinBox: $n = 5$, matrix size is $2^n * n! = 3840$, entries are smallish powers of $q$.

We then deduce $e(i) = 2^{n-1}n!/i, f(i) = 2^n n!(i-1)/i$

Specific computation: Recent hybrid Smith form algorithm [Wan, S 2004]. (fastest way to get integer determinant in this case).

## Krattenthaler $\pi$ formula determinants

Another family of determinants generates formulas for $\pi$. Interesting for LinBox: Matrix is very sparse with mostly small (9 digit) entries, but a few entries are very large (1000 digits).

Blackbox for A = B + C, where B entries are int, C entries are GMP integers Ax = Bx + Cx, where C has few nonzeroes and Bx is fast to compute.

## Royle's graph adjacency matrices

Problem: Find two graphs with cospectral *symmetric cubes*.

Subproblem: Pairs of *strongly-regular* graphs with cospectral symmetric square have been found. Do any pairs of strongly-regular graphs have cospectral symmetric cubes?

Specific computation: Determinants of $A + \alpha I$ mod $p$, for 32548 matrices. Each of them is of order 7140 with 295680 nonzeros ($n^{1.4}$).

Ans: (Pernet and Dumas) No cospectral pairs for strongly-regular graphs with 36 or fewer vertices. Time cost: About one minute per determinant. Use blackbox determinant algorithm discussed above.

## Chandler's Toeplitz matrices

Problem: Smith forms of 0,1 Toeplitz matrices. Order about 10000. Incidence matrix of flats in projective spaces.

Richly structured Smith forms, but only one small prime occurring except in largest invariant factor. Easy for LinBox.

## Lie Atlas operator signatures

General topic: understanding symmetry

Specific question: For Weyl group E8, given lie algebra operator $\alpha$ constructed in a certain way, is $\alpha$ positive (semi)definite in every irreducible representation?

There are 112 representations. The largest is as $7168 \times 7168$ rational matrices. The operator $\alpha$ maps to matrix $A$ which is dense and has entries of length about 100 digits. But also $A$ has a representation as a product of 121 very sparse matrices, each with quite small entries.

In the other representations the structure is the same but the matrix order is smaller and the entry lengths are smaller.

Solution:
Method 1. LU plus CRA of diagonal entries - use A in dense form (construction a major cost).

Method 2. Minimal polynomial plus CRA of coefficients - use A in product form as a blackbox.

LU or minimal polynomial: about 12000 instances mod wordsize primes are needed.

[Adams, Saunders, Wan 2005]: Obtained complete computation for an operator $\alpha_1$ of low rank (verifies a difficult recent theorem). Partial solution for an operator $\alpha_2$ of full rank. Estimation that the order 7168 representation will take 2 cpu years for this operator by current methods.

Lie Atlas group has desire to compute signatures for many such operators.

## Summary of Sparse Smith Form algorithms

$n \times n$ Sparse matrices with constant size entries, using standard Mat Mul.

| Author(s) | Year | Time Cost | Method |
|---|---|---|---|
| Giesbrecht | 1996 | $O^{\sim}(n^4)$ | P, MinPoly |
| Dumas, Saunders, and Villard | 2000 | $O^{\sim}(n^4)$[a] | P, local Smith form |
| Eberly, Giesbrecht and Villard | 2000 | $O^{\sim}(n^{3.5})$ | P, M++, Bisection |
| Saunders and Wan | 2005 | $O^{\sim}(n^{3.5})$ | P, M+ |

Open problem: Memory efficient (Blackbox) method for local (mod $p^e$) Smith form.

[a] We ignore the time for factoring the valences.

M++ $SoftO(n^2)$ mem.

M+ $O(n^2)$ mem.

## Numeric/Symbolic Rational solver

[Wan 04] successive refinement using numeric solver.

Let $r = b$.

Repeat:

approximate: $x = A^{-1}r$ for some significant bits. (mod $1/2^{\alpha}$)

amplify: $r = (Ax - r)2^{\alpha}$

adjust: truncate $r$.

Dixon revisited:

$r = b$.

repeat

approximate: $x = A^{-1}r(\text{mod} p)$

amplify: $r = (Ax - b)/p$.

adjust: $r(\text{mod} p)$.

$$\begin{array}{llll} x_0, & x_0 + x_1/2^\alpha, & x_0 + x_1/2^\alpha + x_2/2^{2\alpha}, & \ldots \\ n^e, & \quad n^e, & \qquad\quad n^e, & \ldots \end{array}$$

Cost can be $\mathrm{O}^{\sim}(()n^{1+e})$, if numeric solver can get $\alpha$ bits of accuracy.

Next: A success story. After: then what?

## Trefethen's matrix

$$T_{i,j} = \begin{cases} i\text{-th prime}, & \text{if } i = j, \text{[diagonal of primes]} \\ 1, & \text{if } i - j \text{ is a power of 2[bands of 1's]} \\ 0, & \text{otherwise.[very sparse matrix]} \end{cases}$$

$$T_9 = \begin{bmatrix} 2 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 3 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 5 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 7 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 11 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 13 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 17 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 19 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 23 \end{bmatrix}.$$

$$T_{20000}\, X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

What $X_1$?

## Rational solve on Trefethen's matrix

2002

- CRA - Jean-Guillaume Dumas, (4 days, 180 procs)
- CRA and Dixon, Hensel lifting - William Turner ( est. similar effort )
- Dixon, Hensel lifting - Zhendong Wan (12.5 days, 1 proc, big mem )

2004

- Hybrid numeric/symbolic solver - Zhendong Wan (12.5 minutes, 1 proc, small mem )
  Days to minutes: A factor of 1440 speedup!
- Solution is quotient of integers having $10^5$ digits.

## Hybrid algorithms

Wan's rational solver is an example of a numeric-symbolic hybrid applied to a symbolic problem (rational solution to linear system).

Problem for integer systems is to get the numeric part to converge (numeric preconditioning issue). Poor luck so far.

The **converse problem** may be more important. Consider a (large, sparse) numeric linear system which is

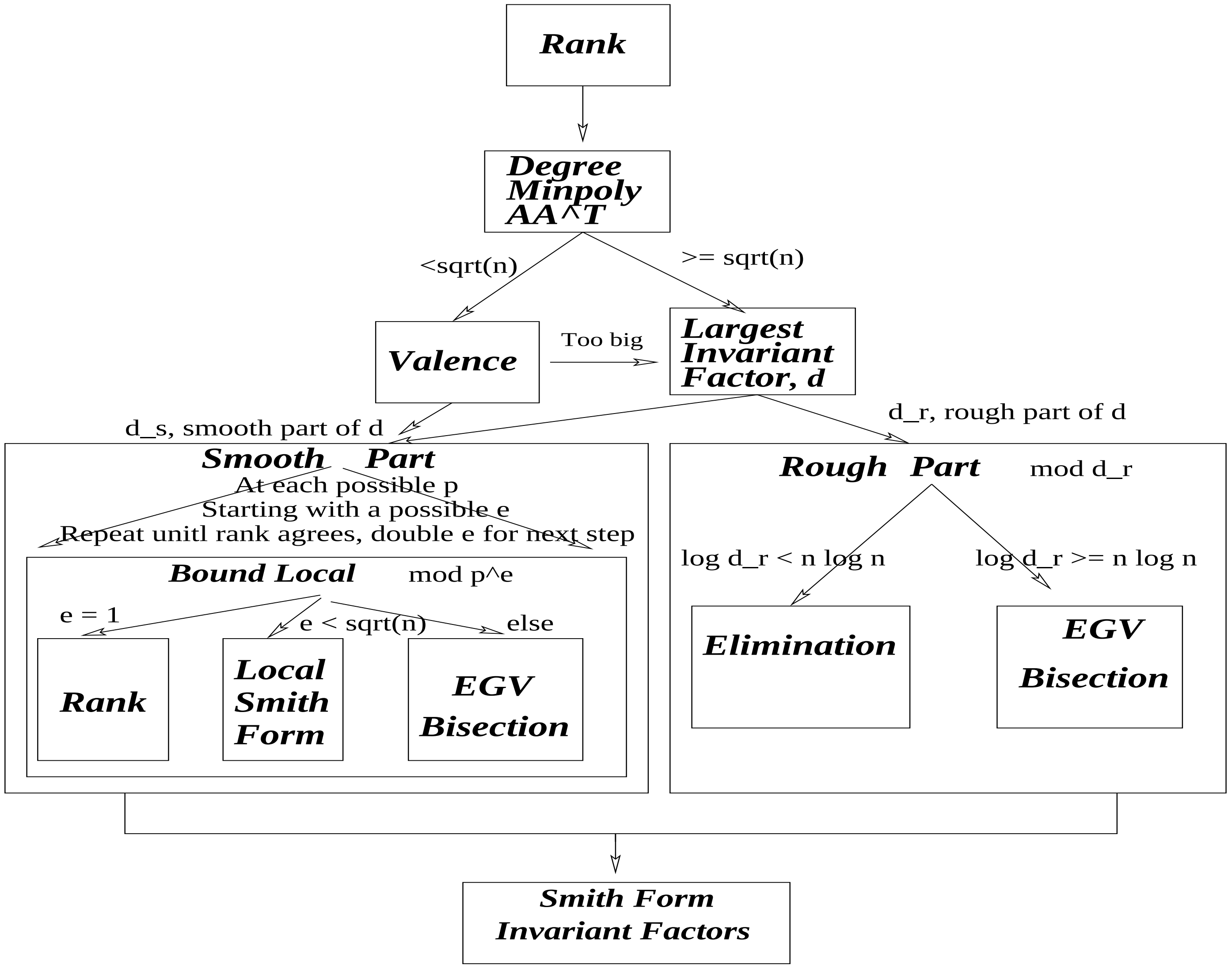
- too large or too nasty (fill in) for sparse direct solvers.
- unresponsive to iterative methods.

It may prove useful to solve it exactly. Relative to numeric iterative methods existing blackbox method is ”slow but sure”.

## Predictions

- Elimination/Blackbox hybrids (in particular, fuller implementation and use of block methods).
- Symbolic/Numeric hybrids
- Exact linear algebra applied to numeric problems.
- Linking of LinBox into general purpose systems such as Maple, Mathematica.
  Alternative: native reimplementations - not likely
- Mod p in O$^{\sim}$(()$n^{1+e}$) plus chinese remainder algorithm $\Rightarrow$ Integer problems in O$^{\sim}$(()$n^{2+e}$) time. Better?
- Extension to polynomial matrices.

# An engineered Smith form algorithm



Rank
Degree Minpoly AA^T
<sqrt(n)
>= sqrt(n)
Valence
Too big
Largest Invariant Factor, d
d_s, smooth part of d
d_r, rough part of d
Smooth Part
At each possible p
Starting with a possible e
Repeat unitl rank agrees, double e for next step
Bound Local mod p^e
e = 1
e < sqrt(n)
else
Rank
Local Smith Form
EGV Bisection
Rough Part mod d_r
log d_r < n log n
log d_r >= n log n
Elimination
EGV Bisection
Smith Form Invariant Factors