

I

# Symbolic Mathematical Computation: Is it right to be exact?

David Saunders - U of Delaware

September 24, 2007

Abstract:

A duality between exactness and approximation arises in many areas of human activity. Somewhat paradoxically, the uses of /em digital computers have lead to great strides in the mathematical understanding of *approximation*. At the same time, but somewhat less visibly, great strides have been made in understanding how to do *exact* mathematical computations as well. I will give a status report with examples linear algebra, computations with integer matrices.

---

<sup>1</sup>supported by NSF grant CCR0515197

A brief history of exact linear algebra

A brief history of exact linear algebra  
computation

A brief history of **sparse matrix** exact linear  
algebra computation

## Outline

- LinBox project and other implementation efforts.
- Prehistory.
- Algorithmic developments for dense matrices.
- Algorithmic developments for sparse and structured matrices (blackbox methods).
- survey of applications.
- predictions and open problems.

## Exact as opposed to what?

Linear algebra may be exact or approximate. Symbolic or numeric. In the numeric case one is concerned with accuracy, the issues of stability and convergence of methods.. In the exact case one is concerned with expression size, the problems of memory and runtime created by expression swell during computations.

Neither set of issues was paid much mind before the computer age.

## 19th century

All linear algebra was exact.

1801, Gauss described the “common” method of solving linear equations, also invented modular arithmetic.

Determinant preceded matrix as concept in 19th century.  
(Sylvester, Cauchy, Binet...)

Most important mathematics of linear algebra was developed such as canonical forms.

1861 Smith introduces normal form for integer matrix equivalence.

18xx Frobenius canonical form for rational matrix similarity.

1870 Jordan canonical form (over finite fields!) for matrix similarity.

In all, they produced the underpinnings so that the 20th century can view linear algebra as the simple, completely understood case to which you can reduce your favorite problem.

## First half of 20th century

The big thing is the modern algebra movement.

Abstract Algebra provides a useful mathematical framework for

1. modern modular algorithms (Smith form over  $Z_{p^e}$  helping compute Smith form over  $Z$ ), and
2. generic programming approach. Key abstractions: arithmetic domain of computation (field, ring), the blackbox (Linear operator on finite dimensional vector space, module).

## Second half of 20th century

The great story is the story of numeric linear algebra.

... of it's methods

exact, direct methods applied approximately (questions of stability)

approximate, iterative methods (questions of convergence)

... of it's fundamental role in scientific computing

## Second half of 20th century in Computer Algebra

The great stories are the stories of closed form integration, symbolic differential equations solving, and of polynomial manipulation.

60's: Slagle's "Saint", Moses' "Sin" programs. Richardson's elementary function integration algorithm. General purpose symbolic math systems REDUCE and Macsyma.

70's: Polynomial GCD and factoring. First significant improvement in GCD since Euclid.

80's: Second generation general purpose commercial symbolic math systems Maple, Mathematica, Magma (group theory)

90's: Greater attention to special purpose systems to obtain high performance for specific categories of symbolic math: Linear algebra (LinBox), Buchberger's algorithm. Also emergence of open source systems in the face of the high cost and high inertia of the 3 M's. (singular, gap, cocoa, pari, etc.)

## 21st Century...

00's Sage (Stein, Joyner), QEPCad/SACLIB(Collins, Brown)

Exact linear algebra draws from a half century of numeric (approximate) linear algebra developments, producing computational capability for problems that were (a) unthinkable by approximate methods, or (b) thinkable, but unsolvable due to computational stability problems.

## 70s - Era of algebraic complexity

Algebraic complexity - count arithmetic operations: ambition, distraction, uglification, and derision. (vs bit complexity - also consider the sizes of the operands)

Arithmetic complexity measures serve

- well for matrix of floating point. (Coincides with development of numeric linear algebra)
- well for matrix over small finite field.
- poorly 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:  $O(n^2) \neq O(S(n))$ ?  $O(S(n)) \neq O(M(n))$ ?

## 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  $O(n^2d + nd)$ .

$\text{Det}(A)$  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  $\text{det}(A)$  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  $O^\sim(n^2)$ , size of  $x$  is  $O^\sim(n^2)$ .

## Direct Elimination Cost

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

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

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  $\mathbf{Z}$ , solve  $Ax = b$ . Choose a prime  $p$  near  $2^{32}$  (wordsize prime)

Compute  $LU \bmod p$ .  $O(n^3)$ . 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  $O^\sim(n^3)$ , memory  $O^\sim(n^2)$ .

$$Ax_0 = b(\text{mod } p), \quad A(x_0 + x_1p) = b(\text{mod } p^2), \quad A(x_0 + x_1p + x_2p^2) = b(\text{mod } p^3), \quad \dots$$

$$n^3, \quad n^2, \quad n^2, \quad \dots$$

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.

$$A = \begin{bmatrix} 88 & 52 & -33 & -95 & 20 & 25 & -22 & 57 & 27 & \\ -82 & -13 & -68 & -20 & -61 & 94 & 45 & 27 & 8 & \\ -70 & 82 & -67 & -25 & -48 & 12 & -81 & -93 & 69 & \\ 41 & 72 & 22 & 51 & 77 & -2 & -38 & -76 & 99 & \\ 91 & 42 & 14 & 76 & 9 & 50 & -18 & -72 & 29 & \\ 29 & 18 & 16 & -44 & 31 & 10 & 87 & -2 & 44 & \\ 70 & -59 & 9 & 24 & -50 & -16 & 33 & -32 & 92 & \\ -32 & 12 & 99 & 65 & -80 & -9 & -98 & -74 & -31 & \\ -1 & -62 & 60 & 86 & 43 & -50 & -77 & -4 & 67 & \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -41/44 & 1 & 0 & 0 & 0 & 0 & 0 \\ -35/44 & 1357/390 & 1 & 0 & 0 & 0 & 0 \\ 41/88 & 1051/780 & 106351/156218 & 1 & 0 & 0 & 0 \\ 91/88 & -259/780 & 9569/156218 & 9469464/4130959 & 1 & 0 & 0 \\ 29/88 & 19/780 & 18271/156218 & -3315826/4130959 & -253468858/717012689 & 0 & 0 \\ 35/44 & -184/65 & -76218/78109 & 699563/590137 & 2646876953/2868050756 & -4126510 & 0 \\ -4/11 & 34/39 & 54004/78109 & -5192380/4130959 & 296262959/717012689 & 4423895 & 0 \\ -1/88 & -1351/780 & -69523/156218 & 1581304/4130959 & -2102895/84354434 & -1405315 & 0 \end{bmatrix}$$

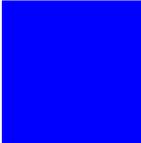
$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -41/44 & 1 & 0 & 0 & 0 & 0 & 0 \\ -35/44 & 1357/390 & 1 & 0 & 0 & 0 & 0 \\ 41/88 & 1051/780 & 106351/156218 & 1 & 0 & 0 & 0 \\ 91/88 & -259/780 & 9569/156218 & 9469464/4130959 & 1 & 0 & 0 \\ 29/88 & 19/780 & 18271/156218 & -3315826/4130959 & -253468858/717012689 & 0 & 0 \\ 35/44 & -184/65 & -76218/78109 & 699563/590137 & 2646876953/2868050756 & -4126510 & 0 \\ -4/11 & 34/39 & 54004/78109 & -5192380/4130959 & 296262959/717012689 & 4423895 & 0 \\ -1/88 & -1351/780 & -69523/156218 & 1581304/4130959 & -2102895/84354434 & -1405315 & 0 \end{bmatrix}$$

$$\frac{-163135032340604}{41702403318845}, \quad \frac{1179722492402710}{2024370527400749}$$

## Matrix with one million entries

1000 × 1000, **dense** 

200 × 5000, dense 

2000 × 2000, 1/4 

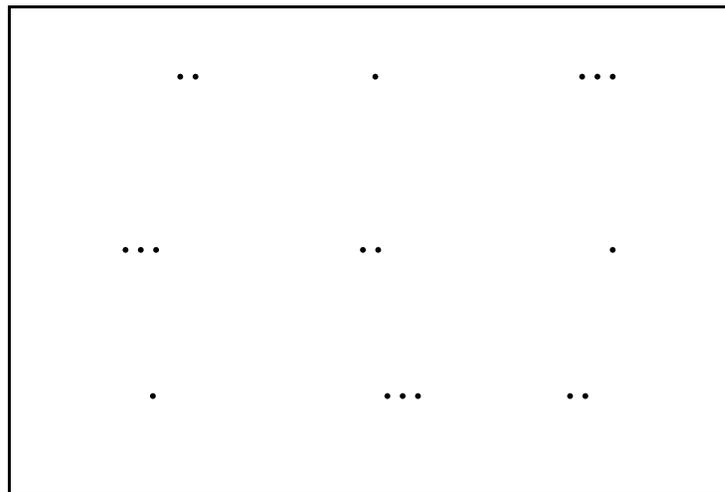
*dense*

10000 × 12000

100 per row, *sparse*

100000 × 100000

10 per row, **sparse**



## measure sparsity as power of dimension

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

$n = 1000 \Rightarrow 10^6 = n^2$ , all entries nonzero.

$n = 2000 \Rightarrow 10^6 = n^{1.82}$ , one in 4 entries nonzero.

$n = 10000 \Rightarrow 10^6 = n^{1.5}$ , one in 100 entries nonzero.

$n = 100000 \Rightarrow 10^6 = n^{1.2}$ , one in ten thousand is nonzero.

Gaussian elimination - Dense matrix:  $n^3 = n * n^e$  arithmetic ops.

( $e = 2$ )

Gaussian elimination - Sparse matrix:  $n \leq \text{ops} \leq n^3$

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

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

$$I, A, A^2, \dots, A^d, \dots$$

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

$$b, Ab, A^2b, \dots, A^e b, \dots$$

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

$$u^T b, u^T Ab, u^T A^2 b, \dots, u^T A^f b, \dots$$

The *minimal polynomial* of a matrix is the lowest degree monic polynomial is  $m_A(x) = \sum_i^d m_i x^i$ , such that

$$m_0 I + m_1 A + m_2 A^2 + \dots + m_d A^d = 0$$

The minimal polynomial of the vector sequence determined by  $A, b$  is  $m_{A,b}(x) = \sum_i^d m_i x^i$ , such that

$$m_0 I b + m_1 A b + m_2 A^2 b + \dots + m_d A^e b = 0$$

The minimal polynomial of the scalar sequence determined by  $u, A, b$  is  $m_{u,A,b}(x) = \sum_i^d m_i x^i$ , such that

$$m_0 u^T I b + m_1 u^T A b + m_2 u^T A^2 b + \dots + m_d u^T A^f b = 0$$

$f \leq e \leq d$ , even  $m_{u,A,b}(x) | m_{A,b}(x) | m_A(x)$

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

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

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

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

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

$$u = (1 \ 1).$$

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

$$u = (1 \ 0).$$

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

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,  $\det(A) = \text{constant}$  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].

## 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  $\text{GF}(2)$  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**

## Trefethen's matrix

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

$$T_9 = \begin{bmatrix} 2 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 3 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 5 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 7 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 11 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 13 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 17 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 19 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 23 \end{bmatrix} \cdot T_{20000} X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

What  $X_1$ ?

2	1	1	0	1	0	0	0	0	1
0	$5/2$	$1/2$	1	$-1/2$	1	0	0	0	$-1/2$
0	0	$22/5$	$4/5$	$3/5$	$-1/5$	1	0	0	$-2/5$
0	0	0	$71/11$	$12/11$	$-2/11$	$7/11$	1	1	$3/11$
0	0	0	0	$1439/142$	$159/142$	$127/142$	$-12/71$	1	$29/71$
0	0	0	0	0	$17850/1439$	$1388/1439$	$1324/1439$	1	$158/1439$
0	0	0	0	0	0	$148277/8925$	$8671/8925$	1	$9407/8925$
0	0	0	0	0	0	0	$95701/5113$	1	$4575/5113$
0	0	0	0	0	0	0	0	1	$61681783/2775329$

## 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".

## 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  $\langle \dots \rangle$  of size  $n$  is  $\text{NICE}(n)$ .

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

Example problem: Conjectured formula

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

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 nonzeros 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  $A + \alpha I \pmod p$ , 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.

## 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

