

The Complexity of
Accurate Floating Point Computation
and Symbolic Computing

or

Can we do Numerical Linear Algebra
In Polynomial Time?

James Demmel
Mathematics and Computer Science
UC Berkeley

Joint work with
Ben Diamant, W. Kahan, Plamen Koev
Ming Gu, Stan Eisenstat, Ivan Slapničar,
Kresimir Veselić, Zlatko Drmač

Supported by NSF and DOE

Goals

- Def: **Accurate** floating point (FP) computation means with guaranteed relative error < 1
 - $10^{-2} \equiv 2$ digits, $10^{-16} \equiv 16$ digits, ...
 - zero must be exact
- Def: **Efficient** computation of an expression means in time poly(size of the expression, size of the input)
- Def: **CAE** means “compute accurately and efficiently”
- Goal: Understand cost of accurate FP computation
 - What FP expressions can we CAE?
 - Are there FP expressions that we cannot CAE?
 - For what structured matrices (i.e. with FP expressions as entries) are there matrix computations that we can CAE?
 - * LU, QR, Inv, Pinv, SVD, Eig, ...

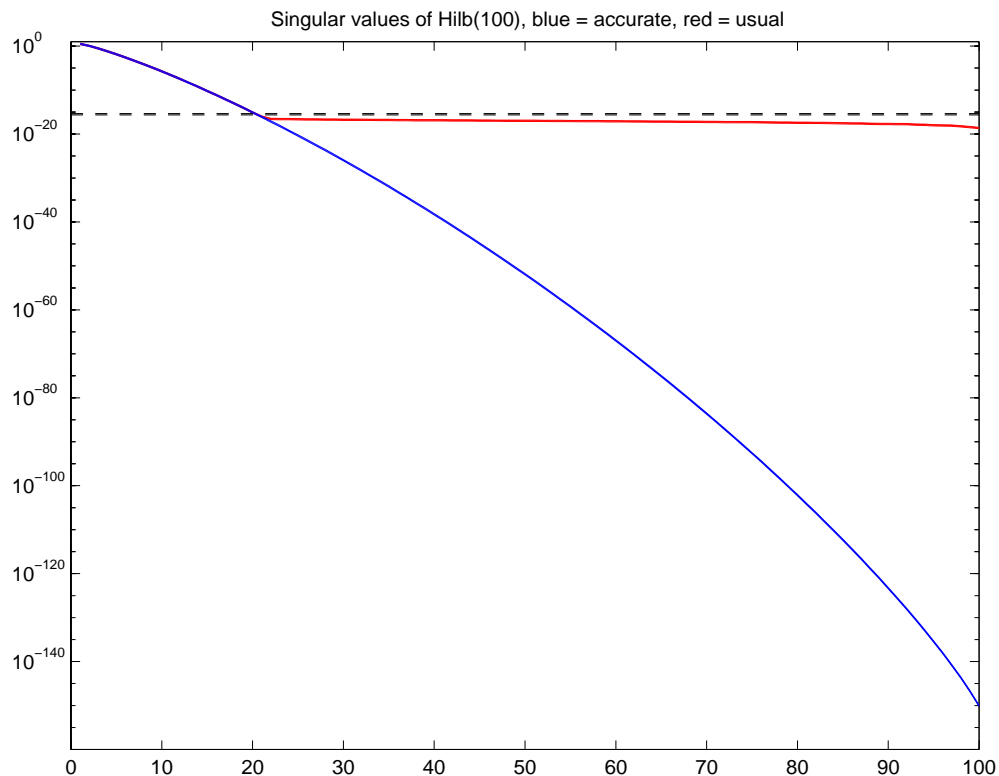
Structure of Results

- Classes of FP expressions/matrices that we can CAE depends strongly on
 - **Set of Numbers**
 1. Positive real numbers
 2. Real numbers
 3. Complex numbers
 - **Model of FP Arithmetic**
 1. Traditional (“ $1 + \delta$ ”) Model:
 $fl(a \otimes b) = (a \otimes b)(1 + \delta)$, $|\delta|$ tiny
 2. Bit model (IEEE) $f \cdot 2^e$, with “large exponents”:
“natural” model for algorithms, analysis
 3. Bit model (IEEE) $f \cdot 2^e$, with “small exponents”:
integers in disguise, well understood
 4. Blum/Shub/Smale (BSS) model (not today)
- Classes described by factorizability properties of expressions/minors of matrices
- New algorithms can be exponentially faster than conventional algorithms that just use enough precision
- $\text{Cost}(\text{CAE in Bit model with large exponents}) \geq \text{Cost}(\text{“symbolic computing”})$
- There are FP expressions that we cannot CAE

Example: 100 by 100 Hilbert Matrix

$$H(i, j) = 1/(i + j - 1)$$

- Eigenvalues range from 1 down to 10^{-150}
- **Old algorithm**, **New Algorithm**, both in 16 digits



- $D = \log(\lambda_1/\lambda_n) = \log \text{cond}(A)$ (**here $D = 150$ digits**)
- “Cost” of Old algorithm = $O(n^3 D^2)$
- “Cost” of New algorithm = $O(n^3)$
- New algorithm cost independent of condition number

Why bother computing tiny eigenvalues accurately?

- Aren't they too ill-conditioned?
- NO - they are determined accurately by the data
 - Hilbert: $H(i, j) = 1/(i + j - 1)$
 - Cauchy: $C(i, j) = 1/(x_i + y_j)$
 - Data are x_i and y_j , not $1/(x_i + y_j)$
 - Changing x_i or y_j in d -th digit changes each eigenvalue in d -th digit
(for Hilbert, analogous for Cauchy)
 - Hilbert matrix is nearly perfectly conditioned!
 - Algorithm works for Cauchy matrices
 - * complex numbers
 - * $1 + \delta$ model
- In some applications, tiny eigenvalues matter most
 - Quantum mechanics: want lowest energy levels
 - Elasticity: want lowest frequencies of vibration
 - Large eigenvalues are artifacts of discretization
 - Computational geometry: what is $\text{sign}(\det(A))$?
- Good to get accurate results if they cost about as much as inaccurate results

Central Role of Minors

- Being able to CAE $\det(A)$ is necessary for CAE
 - $A = LU$ with pivoting
 - $A = QR$
 - Eigenvalues λ_i of A
 - Related factorizations ...
 - * Proof: $\det(A) = \pm \prod_i U_{ii} = \pm \prod_i R_{ii} = \prod_i \lambda_i = \dots$
- Being able to CAE all minors of A is sufficient for CAE
 - A^{-1}
 - * Proof: Cramer's rule
 - * Only need $n^2 + 1$ minors
 - $A = LU$ or $A = LDU$ with pivoting
 - * Proof: Each entry of L, D, U a quotient of minors
 - * Only need $O(n^2)$ or $O(n^3)$ minors
 - Singular values
 - * Proof: new SVD algorithm for $A = LDU$ with pivoting
- Similar result for QR, pseudoinverse via $\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}$, etc.
- Examine which expressions (minors) we can CAE

How do we CAE $A = L \cdot D \cdot U$ for a Hilbert (or Cauchy) Matrix?

- How can we lose accuracy in computing?

- $1 + \delta$ model: $fl(a \otimes b) = (a \otimes b)(1 + \delta)$, $|\delta| \leq \epsilon \ll 1$
- OK to multiply, divide, add positive numbers
- OK to subtract exact numbers (initial data)
- Cancellation when subtracting approximate results dangerous:

$$\begin{array}{r} .12345\mathbf{xxx} \\ - .12345\mathbf{yyy} \\ \hline .00000\mathbf{zzz} \end{array}$$

- Cauchy: $C(i, j) = 1/(x_i + y_j)$
- Fact 1: $\det(C) = \prod_{i < j} (x_j - x_i)(y_j - y_i) / \prod_{i, j} (x_i + y_j)$
 - No bad cancellation \Rightarrow good to most digits
- Fact 2 : Each minor of C also Cauchy
- Fact 3 : Each entry of L, D, U is a (quotient of) minors
- Change inner loop of Gaussian Elimination from

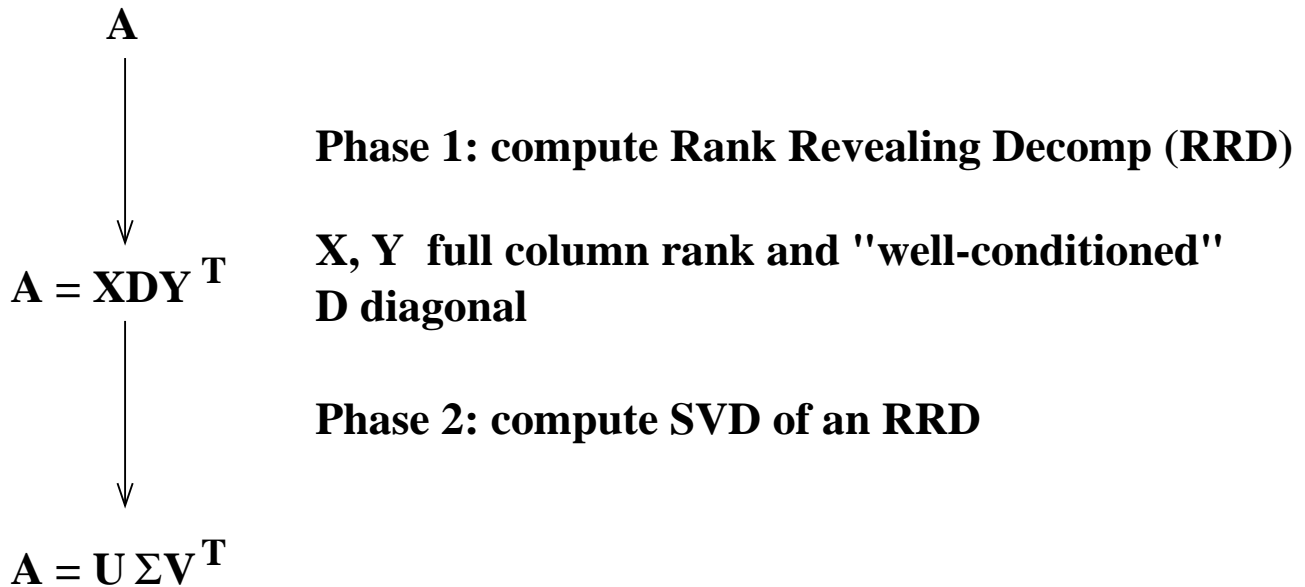
$$C(i, j) := C(i, j) - C(i, k) * C(k, j)$$

to

$$C(i, j) := C(i, j) * (x_i - x_k) * (y_j - y_k) / (x_k + y_j) / (x_i + y_k)$$

- Each entry of L, D, U accurate to most digits!

SVD Algorithm



- **Examples of RRDs:**
 - $A = U\Sigma V^T$, SVD itself
 - $A = QDR$, QR decomposition with pivoting
 - $A = LDU$, Gaussian Elimination with “complete” pivoting (GECF)
- **Phase 1: GECF via (implicitly) computing accurate minors of A**
 - Depends on structure of A (e.g. Cauchy)
- **Phase 2: Works for any RRD in $O(n^3)$**
 - Independent of structure of A

Accurate SVD of $A = XDY^T$

- SVD is $A = U\Sigma V^T$
- Many accurate algorithms, here is simplest:

1. Compute SVD of $DY^T = U_1\Sigma_1V_1^T$
using one-sided Jacobi

2. Multiply $W = XU_1$

3. Compute SVD of $W\Sigma_1 = U\Sigma_2V_2^T$
using one-sided Jacobi

4. Multiply $V = V_1V_2$

- To guarantee efficiency, find eigenvalues of

$$\begin{aligned} \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} &= \frac{1}{2^{1/2}} \cdot \begin{bmatrix} L & L \\ U^T & -U^T \end{bmatrix} \cdot \begin{bmatrix} D & 0 \\ 0 & -D \end{bmatrix} \cdot \begin{bmatrix} L^T & U \\ L^T & -U \end{bmatrix} \cdot \frac{1}{2^{1/2}} \\ &\equiv Z \cdot \hat{D} \cdot Z^T \end{aligned}$$

by performing bisection on $\lambda\hat{D} - Z^{-1}Z^{-T}$

Why roundoff is harmless

- $1 + \delta$ model: $fl(a \otimes b) = (a \otimes b)(1 + \delta)$
- We want $A = U\Sigma V^T$,
 $U = [u_1, \dots, u_n]$, $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$
- But we compute $\bar{A} = \bar{U}\bar{\Sigma}\bar{V}^T$,
 $\bar{U} = [\bar{u}_1, \dots, \bar{u}_n]$, $\bar{\Sigma} = \text{diag}(\bar{\sigma}_1, \dots, \bar{\sigma}_n)$

Absolute
(additive)
Perturbations

vs.

Relative
(multiplicative)
Perturbations

$$\bar{A} = A + \sigma_{\max} \cdot E$$

$$\bar{A} = (I + E)A$$

$$\|E\| \ll 1$$

$$|\sigma_i - \bar{\sigma}_i| \leq \|E\| \cdot \sigma_{\max}$$

$$|\sigma_i - \bar{\sigma}_i| \leq \|E\| \cdot \sigma_i$$

Conventional error bound

Our error bound

Sturm-Liouville Equations

- $-(p(x)y')' + q(x)y = \lambda r(x)y$ on $[0,1]$, $p, q, r > 0$
- **Discretize: second centered differences on uniform grid**
 - **Get** $Tx = \lambda D_r x$
 - $D_r = \text{diag}(r_i)$
 - $T = \text{tridiag}(-p_{i-.5}, p_{i-.5} + p_{i+.5} + h^2 q_i, p_{i+.5})$
 - **Don't even form T !**
- **Rewrite $Tx = \lambda D_r x$ as $GG^T y = \lambda y$ (SVD of G)**
 - **Factor $T = LD_p L^T + h^2 D_q$ where**

$$D_p = \text{diag}(p_{1.5}, \dots, p_{n-.5}) \quad , \quad D_q = \text{diag}(q_1, \dots, q_n)$$

$$L = \begin{bmatrix} 1 & & & \\ -1 & \ddots & & \\ & \ddots & 1 & \\ & & & -1 \end{bmatrix}$$

$$T = U \cdot K^2 \cdot U^T \quad \text{where} \quad U = [L, I] \quad \text{and} \quad K = \text{diag}(D_p, h^2 D_q)$$

$$- G = D_r^{-1/2} U K$$

- **Thm: (Poincaré) U is totally unimodular,**
i.e. all minors $\in \{+1, -1, 0\}$
- **Theorem: High relative accuracy from GE by changing**

$$G(j, k) := G(j, k) - G(j, i) * G(i, k)$$

to

$$\text{if } G(j, k), G(j, i), G(i, k) \text{ all } \neq 0 \text{ then } G(j, k) := 0$$

$$\text{else } G(j, k) := G(j, k) - G(j, i) * G(i, k)$$

Other examples in $1 + \delta$ model

- Sturm-Liouville trick extends to elliptic PDE
 - Second centered differences on uniform square meshes
 - Irregular boundaries ok
 - Still get SVD of
 $G = \text{diagonal} \cdot \text{totally unimodular} \cdot \text{diagonal}$
 - Cost = $O(n^3)$
- Sparse matrices with
 - particular sparsity patterns, Cost = $O(n^3)$
 - particular sparsity and sign patterns, Cost = $O(n^4)$
- Real Vandermonde, other unit displacement rank
 - Reduce to Cauchy
 - Cost = $O(n^3)$
- Totally positive matrices (but cost not always poly)
 - More later ...

- What do these matrices have in common?

$1 + \delta$ Model with Complex Numbers - What we CAN do

- **Def:** $1 + \delta$ allowable expressions may only
 - Multiply and divide
 - Add or subtract initial data
 - **Proof:** All $1 + \delta$ factors can be factored out
- **Def:** A family $A_n(x)$ of n -by- n rational matrices is **fully polyfactorable** if each minor $r(x)$ is a $1 + \delta$ allowable expression of at most $\text{poly}(n)$ factors x_i and $(x_j \pm x_k)$
- **Thm:** Suppose $A_n(x)$ is fully polyfactorable. Then in the $1 + \delta$ model we can CAE LU with pivoting, A^{-1} , singular values.
- Enough for all examples so far (Cauchy, real Vandermonde, Sturm-Liouville, ...)
- Common conditioning analysis
 - Relative uncertainty ϵ in input only magnified by computing $x_i \pm x_j$, yielding $\epsilon(|x_i| + |x_j|)/|x_i \pm x_j|$
 - Condition number is largest $(|x_i| + |x_j|)/|x_i \pm x_j|$
 - Hilbert matrix very well conditioned

$1 + \delta$ Model - What we CANNOT do

- **Thm: Can't compute $x + y + z$ accurately in $O(1)$ time**
 - Assume only \pm , deterministic rounding, exact compare, branch
 - Without branches, show $s = x + y + z + x \cdot f_x + y \cdot f_y + z \cdot f_z$ where f_a a polynomial in rounding errors δ_i 's
 - So if $x + y + z = 0$, need $0 = x \cdot f_x + y \cdot f_y + z \cdot f_z$
 - Show (x, y, z) cannot be orthogonal to (f_x, f_y, f_z) for all independent choices of x, y, z, δ_i 's,
 - Reduce case with branches to case without
- Will see $x + y + z$ is easy in bit model...

Adding Sign Restrictions to $1 + \delta$ Model

- Consider real numbers with fixed signs
 - Fixed signs \Rightarrow can add $x + y + z$ accurately if all positive
 - Real \Rightarrow can compute $x^2 + y^2$ accurately
- New classes of matrices
- Total Sign Compound Matrices (Brualdi and Shader)
 - each matrix entry $+x_{ij}$, $-x_{ij}$ or 0 (all $x_{ij} \geq 0$)
 - each minor a sum of like-signed terms or zero
- Totally Positive Matrices
 - Each minor nonnegative
 - Many examples depending rationally on positive parameters (Karlin)
 - Structure theorems (Brenti, Aiseen, Schoenberg, Whitney, Erdrei) show that all TP matrices can be built from simpler ones via matmul, inversion, Schur complementation, subsetting
 - \implies Can compute any minor accurately in $1+\delta$ model
 - Possibly expensive!
 - Concentrate on CAE TP Generalized Vandermonde matrices

Facts about TP Generalized Vandermonde Matrices

- **TP Vandermonde Matrix:**

$$V = \begin{bmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ 1 & x_2 & \dots & x_2^{n-1} \\ & & \ddots & \\ 1 & x_n & \dots & x_n^{n-1} \end{bmatrix},$$

where $x_1 > x_2 > \dots > x_n > 0$.

- **TP Generalized Vandermonde Matrix:**

$$G_\lambda = \begin{bmatrix} x_1^{\lambda_1} & x_1^{1+\lambda_2} & \dots & x_1^{n-1+\lambda_n} \\ x_2^{\lambda_1} & x_2^{1+\lambda_2} & \dots & x_2^{n-1+\lambda_n} \\ & & \ddots & \\ x_n^{\lambda_1} & x_n^{1+\lambda_2} & \dots & x_n^{n-1+\lambda_n} \end{bmatrix},$$

where

- $x_1 > x_2 > \dots > x_n > 0$
- $\lambda_n \geq \lambda_{n-1} \geq \dots \geq \lambda_0$

- **Def:** $\lambda = (\lambda_n, \lambda_{n-1}, \dots, \lambda_0)$ is **partition** of $|\lambda| = \lambda_n + \dots + \lambda_0$
- **Def:** **Young Diagrams** \equiv partitions:

$$\lambda = (4, 2, 1) = \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$

- **Def:** **Schur Function** $s_\lambda(x_1, \dots, x_n) = \det(G_\lambda) / \det(V)$.
- **Thm:** s_λ is a polynomial with positive integer coefficients depending only on λ (MacDonald)

More facts about TP Generalized Vandermonde Matrices

● **Recall:** $\det(G) = \det(V) \cdot s_\lambda(x_1, \dots, x_n)$

● **Example:**

$$\det \begin{pmatrix} 1 & x_1^2 & x_1^4 \\ 1 & x_2^2 & x_2^4 \\ 1 & x_3^2 & x_3^4 \end{pmatrix} =$$

$$= \det \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{pmatrix} \cdot (2x_1x_2x_3 + x_1^2x_2 + x_1x_2^2 + x_1^2x_3 + x_1x_3^2 + x_2^2x_3 + x_2x_3^2)$$

● $\det(V) = \prod_{i>j}(x_i - x_j)$ is **CAE**


● s_λ **computable accurately** ($x_i > 0$), question is **cost**.

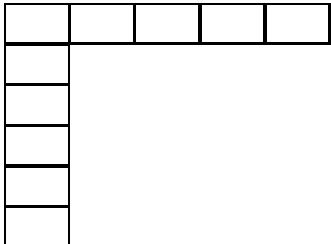
● **Theorem:**

$$s_\lambda(x_1, \dots, x_n, y_1, \dots, y_m) = \sum_{\mu < \lambda} s_\mu(x_1, \dots, x_n) s_{\lambda/\mu}(y_1, \dots, y_m)$$

Allows recursive computation and Divide-and-Conquer approach.

● **Some s_λ 's are CAE:**

$\lambda = (1, 1, 1, \dots, 1) =$ 

and $\lambda = (m, 1, 1, \dots, 1) =$ 

| Type of Matrix | $\det(A)$ | A^{-1} | Any minor | GEPP / GENP | Bidiag Decomp | GECP | SVD |
|--|--------------------|--------------------|-----------|--------------------|--------------------|----------|----------|
| Cauchy | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ | $O(n^3)$ | $O(n^2)$ | $O(n^3)$ | $O(n^3)$ |
| Vandermonde | $O(n^2)$ | NONE | NONE | NONE | $O(n^2)$ | NONE | $O(n^3)$ |
| Totally Positive (TP) Vandermonde | $O(n^2)$ | $O(n^3)$ | EXP | $O(n^3)$ | $O(n^2)$ | EXP | $O(n^3)$ |
| TP Generalized Vandermonde, $ \lambda = \lambda_1 + O(1)$ | $O(\lambda_1 n^2)$ | $O(\lambda_1 n^3)$ | EXP | $O(\lambda_1 n^3)$ | $O(\lambda_1 n^2)$ | EXP | EXP |
| TP Generalized Vandermonde, any λ | $f(n, \lambda)$ | EXP | EXP | EXP | EXP | EXP | EXP |

$$f(n, \lambda) = O(n^{1+\log \lambda_1 + \log \lambda_2 + \dots + \log \lambda_n}) \text{ vs. } O(n^{|\lambda|}) = O(n^{\lambda_1 + \lambda_2 + \dots + \lambda_n})$$

(Symmetrica)

Models of FP Arithmetic

- **Traditional $1 + \delta$ model**

- $fl(a \otimes b) = (a \otimes b)(1 + \delta)$
- **No over/underflow**
- $|\delta| \leq \epsilon < 1$, otherwise *arbitrary*
- **Enough for many examples, but limited**

- **Bit model**

- **Inputs of form $f \cdot 2^e$, e and f integers**
- **Can accurately compute arbitrary rational FP expressions**
- **Question is cost**

- **BSS model (not today)**

Bit Model

- Inputs of form $f \cdot 2^e$, e and f integers
- $\text{size}(X) = \#$ bits used to represent X
- $\text{size}(f \cdot 2^e) = \# \text{bits}(f) + \# \text{bits}(e)$
- Can evaluate any rational expression accurately
 - Convert to poly/poly, using high enough precision
 - Question is cost
- Cost depends strongly on $\#$ exponent bits
 - Small Exponents (SE)
 - * $\# \text{bits}(e) = O(\log(\# \text{bits}(f)))$
 - * Equivalent to integer arithmetic
 - * Can CAE many problems
 - Large Exponents (LE)
 - * $\# \text{bits}(e)$ and $\# \text{bits}(f)$ independent
 - * “Natural” model for algorithm design
 - * Algorithms work for any input magnitudes
 - * Like symbolic algebra, which is much harder

Differences between SE and LE bit models - 1

- Recall definitions for size of $f \cdot 2^e$
 - Small Exponents (SE): $\#bits(e) = O(\log(\#bits(f)))$
 - Large Exponents (LE): $\#bits(e), \#bits(f)$ indep.
- SE and “integer arithmetic” equivalent
 - Represent $f \cdot 2^e$ as integer, not pair (f, e)
 - $\#bits(f \cdot 2^e) = \#bits(f) + e \approx \#bits(f) + 2^{\#bits(e)} = \text{poly}(\#bits(f))$
- LE and “integer arithmetic” not equivalent
 - $2^{\#bits(e)}$ exponentially larger than $\#bits(e)$
- # bits in FP expressions much bigger for LE than SE
 - SE: $\text{size}(x \cdot y) \leq \text{size}(x) + \text{size}(y)$
 - LE: $\text{size}(x \cdot y) \leq \text{size}(x) \cdot \text{size}(y)$
 - $$\begin{aligned} \text{size}(x \cdot y) &= \text{size}((2^{x_1} + 2^{x_2} + 2^{x_3}) \cdot (2^{y_1} + 2^{y_2} + 2^{y_3})) \\ &= \text{size}\left(\sum_{i,j=1}^3 2^{x_i+y_j}\right) \\ &= \text{size}(x) \cdot \text{size}(y) \\ &= 9, \text{ not } 6 \end{aligned}$$
- LE can encode symbolic algebra
 - $2^a \cdot 2^b = 2^{a+b}$ can “remember” a and b
 - Some symbolic algebra problems are very hard

Differences between SE and LE bit models - 2

- Recall definitions for size of $f \cdot 2^e$
 - **Small Exponents (SE):** $\#bits(e) = O(\log(\#bits(f)))$
 - **Large Exponents (LE):** $\#bits(e)$, $\#bits(f)$ **indep.**
- **Cond(A) in LE can be exponentially larger than in SE**
- **SE:** $\log \text{cond}(A)$ is **poly(size(A))**
- * **Conventional algorithms using $\log \text{cond}(A)$ bits are polynomial**
- * **Proof on next slide**
- **LE:** $\log \text{cond}(A)$ **can be exponential in size(A)**
- * $\text{cond}(\text{diag}(2^e, 1)) = 2^e \approx 2^{2^{\#bits(e)}}$
- * **Conventional algorithms using $\log \text{cond}(A)$ bits are not polynomial**
- **$\log \log \text{cond}(A)$ is lower bound on complexity of any FP algorithm**
 - **# bits needed to print out exponent of answer**

Small Exponent (SE) Bit Model

- Consider set of FP expressions of form P/Q , P and Q polynomials
 - P or $Q = \sum_i \alpha_i \cdot x_1^{e_{i1}} \cdots x_k^{e_{ik}}$
 - Let N bound sizes of P and Q
 - * $N \geq \# x_i, \# \text{ nonzero } \alpha_i, \# \text{bits}(\alpha_i), e_{ij}$
 - $F \geq \# \text{ bits in each } x_j$
- **Fact:** $\# \text{ bits}(P) = \text{poly}(N, F)$, same for $\# \text{ bits}(Q)$
 - Computing P and Q costs $\text{poly}(N, F)$
- Consider n -by- n matrix $A(x_1, \dots, x_k)$ with entries P_{ij}/Q_{ij}
 - Suppose $N = \text{poly}(n)$
 - Can compute each P_{ij}, Q_{ij} in time $\text{poly}(n, F)$
 - Can put over common denominator in time $\text{poly}(n, F)$
- **Conclusion:** $\log \text{cond}(A) = \text{poly}(n, F)$
 - Conventional algorithms with $\log \text{cond}(A)$ bits are ok
- **Thm (Clarkson, 1992)** Can CAE any minor in time $O(\text{poly}(n, F))$
 - Can CAE $A = LU$ with pivoting, A^{-1} , A^+ , singular values of such matrices via minors

Large Exponent (LE) Bit Model

- Several obstacles to CAE
 - Clarkson's Thm does not work
 - $N \geq$ size of polynomial P
 - $F \geq$ # fraction bits in any value
 - $E \geq$ # exponent bits in any value
 - # bits (P) *not* poly(N, F, E)
 - **Example:** #bits($\prod_{i=1}^N (1 + x_i)$) can grow like 2^N
- **Thm:** It is PP-Hard to compute an arbitrary bit of $\prod_{i=1}^N (1 + x_i)$
 - Reduce permanent to finding coeff C of $\prod_{i=1}^n u_i v_i$ in $\prod_{i,j=1}^n (u_i v_j A_{ij} - 1)$ where $A_{ij} = 0, 1$
 - Reduce C to finding a few bits of $\prod_{i=1}^{n^2} (1 + 2^{n_i})$
- **Conjecture:** We cannot CAE

$$\prod_{i=1}^N (1 + x_i) - \prod_{j=1}^N (1 + y_j)$$

- **Justification:** If many x_i and y_j the same, exponentially many leading bits of the two products can cancel. It seems hard to compute the first bit that does not cancel without finding the exponentially many that do.

But isn't $\det(A) = 0$ in NP?

- Not necessarily in LE model
- What is witness for singularity of A ? A null vector
- But null vector for a matrix with LE entries can have exponentially many bits
 - Even for a tridiagonal

Sparse Arithmetic: Computing in Large Exponent Model

- How much work does it take to compute $(B + s) - B$, where $B \gg s$?
- How many bits does it take to represent $B + s$?
- In usual “dense arithmetic” model, depends on relative sizes of B and s
- Storing $2^e + 1$ takes e fraction bits
- In “sparse arithmetic”, $B + s$ stored as (B, s) , exponentially smaller
- Widely used double-double arithmetic is special case of sparse arithmetic (Priest, Shewchuk, Bailey, Briggs, Bohlender, Kahan, Dekker, Pichat ...)
- Same idea very practical for higher precision
 - Store number as array of “nonoverlapping” FP numbers
 - Used in some exact inner product implementations

Which FP Expressions can we CAE in the Large Exponent Model?

- Def: $r(x)$ is in **factored form** if

$$r(x) = \prod_{i=1}^n p_i(x_1, \dots, x_k)^{e_i}$$

where

$$p_i(x_1, \dots, x_k) = \sum_{j=1}^t \alpha_{ij} \cdot x_1^{e_{ij1}} \cdots x_k^{e_{ijk}}$$

and

$$\text{size}(r) = \max(n, t, \log |e_i|, \text{size}(\alpha_{ij}), \text{size}(x_i), \text{size}(e_{ijk}))$$

- Thm (mostly Priest) We can CAE r in time $\text{poly}(\text{size}(r))$
 - Compute $p_i(x_1, \dots, x_k)$ exactly, with sparse arithmetic
 - CAE $p_i(x_1, \dots, x_k)^{e_i}$ by repeated squaring
 - CAE $\prod_{i=1}^n p_i(x_1, \dots, x_k)^{e_i}$ in conventional FP
- Def: A family $A_n(x)$ of n -by- n rational matrices is **polyfactorable** if each minor $r(x)$ is in factored form of size $\text{size}(r) = O(\text{poly}(n))$
- Thm: Suppose $A_n(x)$ is polyfactorable. Then in the Large Exponent model we can CAE LU with pivoting, A^{-1} , singular values.

Accurate and Efficient Matrix Computations in Large Exponent Model

- **Thm:** Suppose $A_n(x)$ is **polyfactorable**.
Then in the Large Exponent model we can CAE LU with pivoting, A^{-1} , singular values.
- **Recall Thm:** Suppose $A_n(x)$ is **fully polyfactorable**.
Then in the $1 + \delta$ model we can CAE LU with pivoting, A^{-1} , singular values.
- What matrices can we do in LE model that we could not do in $1 + \delta$ model?
 - Complex Vandermonde matrices, not just real
 - Green's matrices
 - * inverses of tridiagonals
 - * represented as $A_{ij} = x_i y_j, i < j$
 - Any fully polyfactorable matrix where we substitute polynomials
 - * Substitute $x_i = p_i(z), y_j = q_j(z)$ in $C_{ij} = 1/(x_i + y_j)$
- What matrices can we do in SE model that we could not do in LE model?
 - A lot
 - Any matrix whose entries are reasonable rational functions of data
 - Matrix of n^2 FP entries

Cost comparison to symbolic algebra

- **Cost(Accurate evaluation in Large Exponent Model) \geq Cost(symbolic expression $\equiv 0$?)**
- **Proof idea: Simulate symbolic algebra using large exponents**
 - **Cost(Accurate evaluation of p) \geq Cost($p \equiv 0$?)**
 - **Suppose $p(x_0, \dots, x_{m-1})$ a symbolic polynomial with max degree $D - 1$, integer coeffs of max # bits $B - 1$**
 - **Let $X_i = 2^{B \cdot D^i}$**
 - **Then $p(X_0, \dots, X_{m-1}) = 0$ iff $p \equiv 0$**
 - * **Idea: bits in typical term $\alpha \cdot x_1^{e_1} \cdots x_{m-1}^{e_{m-1}}$ of p do not “overlap” so cannot cancel in sum**
- **Example: determinant of A each entry of which is rational**

Open Questions

- Are there FP expressions that we provably cannot CAE in Large Exponent (LE) model?
 - $\prod_{i=1}^n (1 + x_i) - \prod_{j=1}^n (1 + y_j)$
 - Determinant of general matrix
 - Determinant of tridiagonal matrix
 - Such an example could distinguish LE from SE
 - Conversely, are there fast algorithms for these?
- What does symbolic computing complexity tell us about CAE in LE model, and vice versa?
- What changes if we have sign information?
 - We have accurate algorithms for all Totally Positive (TP) matrices, but not efficient
 - How big a class of TP matrices can we do efficiently?
 - Generalized Vandermondes \equiv Schur Functions
- What changes if we have complex data?
- Differential equations
 - Only simplest ones understood
 - What about other discretizations?
 - Conjecture: Accuracy depends only on geometry, not material properties
- What about nonsymmetric eigenproblem?
- Blum/Shub/Smale model

Conclusions

- We have identified many classes of floating point expressions and matrix computations that permit
 - Accurate solutions: relative error < 1
 - Efficient solutions: time = poly(size)
- Classes depend strongly on model of FP arithmetic
 - $1+\delta$ model: most limited classes, but still interesting
 - Bit model with Large Exponents: strictly more, but limited by complexity of symbolic algebra
 - Bit model with Small Exponents: strictly more again, since not “really” FP, just integer arithmetic
- Need new algorithms for Large Exponents
 - Conventional algorithms with enough precision would be exponentially slower
 - New algorithm much faster in practice!
- Lots of open problems
- Reports available
 - www.cs.berkeley.edu/~demmel/NASC.ps
to appear Contemporary Mathematics soon
 - SIMAX, v. 21, n. 2, pp 562–580, 1999
 - Lin. Alg. Appl., vol 299, issue 1-3, pp 21–80, 1999
 - These slides: www.cs.berkeley.edu/~demmel/ISSAC2001_2.ps