The Complexity of Accurate Floating Point Computation

or

Can we Compute Eigenvalues In Polynomial Time?

James Demmel
Mathematics and Computer Science
UC Berkeley

Joint work with
Plamen Koev, Yozo Hida, Ben Diament

W. Kahan, Ming Gu, Stan Eisenstat, Ivan Slapničar,
Krešimir Veselić, Zlatko Drmač

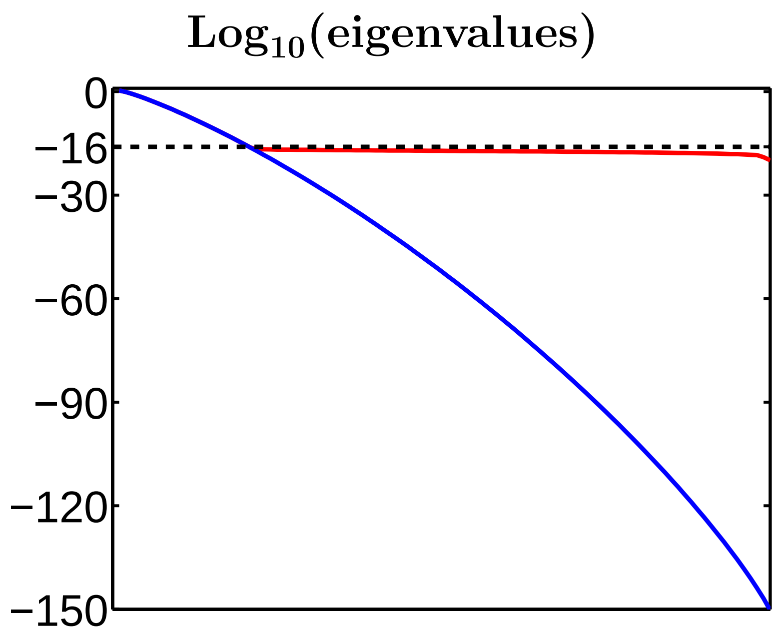
Supported by NSF and DOE

Goal

- Compute $y = f(x)$ with floating point data x accurately and efficiently
- $f(x)$ may be
  - Rational function
  - Solution of linear system $Ay = b$
  - Solution of eigenvalue problem $Ay = \lambda y$ ...
- Accurately means with guaranteed relative error $e < 1$
  - $|y_{\text{computed}} - y| \le e \cdot |y|$
  - $e = 10^{-2}$ means 2 leading digits of y_{computed} correct
  - $y_{\text{computed}} = 0 = y$ must be exact
- Efficiently means in "polynomial time"
- Abbreviation: CAE means "Compute Accurately and Efficiently"

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 digit arithmetic



- Cost of Old algorithm in high enough precision $= O(n^3 D^2)$ where $D = \#$ digits $= \log(\lambda_{\max}/\lambda_{\min}) = \log \text{cond}(A) = 150$ decimal digits
- Cost of New algorithm $= O(n^3 \log D)$
- When D large, new algorithm exponentially faster
- New algorithm exploits structure of Cauchy matrices

Example: Adding Numbers in Traditional Model of Arithmetic

- $fl(a \otimes b) = (a \otimes b)(1 + \delta)$ where roundoff error $|\delta| \le \epsilon \ll 1$
- How can we lose accuracy?
  - OK to multiply, divide, add positive numbers
  - OK to subtract exact numbers (initial data)
  - Accuracy may only be lost when subtracting approximate results:

```
    .12345xxx
-   .12345yyy
-------------
    .00000zzz
```

- Thm: In Traditional Model it is impossible to add $x + y + z$ accurately
  - Proof: $\forall$ algorithms $\exists$ inputs x, y and z and errors δ that make error large

Example: Adding Numbers in Bit Model of Arithmetic

- $x = m \cdot 2^e$ where m=mantissa and e=exponent are integers
- $fl(x + y)$ is correctly rounded result
- Algorithm for $S = \sum_{i=1}^{n} x_i$

  Sort so $|x_1| \geq |x_2| \geq \cdots \geq |x_n|$
  $S = 0$
  for $i = 1$ to n
  $\quad S = S + x_i$

- Thm: Suppose each x_i has b-bit mantissa and S has B-bit mantissa, where $b < B \leq 2b$. Then
  - If $n \leq 2^{B-b} + 1$, then S accurate
  - If $n \geq 2^{B-b} + 3$, then S may be completely wrong (wrong sign)
- Ex: x_i double ($b = 53$), S extended ($B = 64$) $\Rightarrow n \leq 2^{11} + 1 = 2049$

Structure of Results (1)

- Classes of rational expressions (matrices whose entries are expressions) that we can CAE depends strongly on Model of FP Arithmetic

  1. Traditional Model (TM for short):
     $fl(a \otimes b) = (a \otimes b)(1 + \delta)$ where $|\delta| \leq \epsilon \ll 1$
     no over/underflow
  2. Bit model: inputs are $m \cdot 2^e$, with "long exponents" e (LEM for short)
  3. Bit model: inputs are $m \cdot 2^e$, with "short exponents" e (SEM for short)

  4. Other models have been proposed (not today)
     (a) Blum/Shub/Smale
     (b) Cucker/Smale
     (c) Pour-El/Richards

Structure of Results (2)

- **Classes of expressions (matrices) that we can CAE are described by factorizability properties of expressions (minors of matrices)**

$$\mathbf{TM} \subsetneq \mathbf{LEM} \underset{\neq ?}{\subseteq} \mathbf{SEM}$$

- **New algorithms can be exponentially faster than conventional algorithms that just use high enough precision**
- **Cost(CAE in LEM) related to Cost(using symbolic computing)**
- **Cost(CAE in SEM) related to Cost(using integers)**

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 ...
    - Proof: $\det(A) = \pm \prod_i U_{ii} = \pm \prod_i R_{ii} = \prod_i \lambda_i = \cdots$
- 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; $O(n^3)$ needed
  - Singular values of A (SVD): Eigenvalues of $A^T A$
    - Proof: Compute $A = LDU$ with complete pivoting, then SVD of LDU
- Similar result for QR, pseudoinverse via minors of $\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}$, etc.
- Examine which expressions (minors) we can CAE

Accurate Singular values of any rank-revealing $A = LDU^T$

- Rank-revealing $\equiv$ D diagonal, L and U well-conditioned
- Algorithm 1: Find eigenvalues of

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

  by performing bisection on $\lambda \hat{D} - Z^{-1} Z^{-T}$
- Algorithm 2: Two applications of one-sided Jacobi, matrix multiplication

Outline of Remainder of Talk

1. **What we can do in Traditional Model (TM)**
2. **What we can do in Bit Model (SEM and LEM)**

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

- To maintain accuracy, avoid subtracting intermediate results
- Cauchy: $C(i,j) = 1/(x_i + y_j)$
- Fact 1: $\det(C) = \Pi_{i<j}(x_j - x_i)(y_j - y_i) / \Pi_{i,j}(x_i + y_j)$
- 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

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

  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!

Cost of Accuracy in TM (1)

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy								
TP Cauchy								
Vandermonde								
TP Vandermonde								
Confluent Vandermonde								
TP Confluent Vandermonde								
Vandermonde 3 Term Orth. Poly.								
Generalized Vandermonde								
TP Generalized Vandermonde								

GENP/PP/CP = Gaussian Elimination with No/Partial/Complete Pivoting
SVD = Singular Value Decomposition
NENP = Neville Elimination (bidiagonal factorization) with No Pivoting

Cost of Accuracy in TM (2)

TP = Totally Positive (all minors nonnegative)

Matrix Type	
Cauchy	$C_{ij} = 1/(x_i + y_j)$
TP Cauchy	$x_i \nearrow$, $y_j \nearrow$, $x_1 + y_1 > 0$
Vandermonde	$V_{ij} = x_i^{j-1}$, x_i distinct
TP Vandermonde	$0 < x_i \nearrow$
Confluent Vandermonde	if some x_i coincide, differentiate rows of V
TP Confluent Vandermonde	$0 < x_i \nearrow$
Vandermonde 3 Term Orth. Poly.	$V_{ij} = P_j(x_i)$, P_j orthogonal polynomial from 3-term recurrence
Generalized Vandermonde	$G_{ij} = x_i^{\lambda_j + j - 1}$, λ_j nonnegative increasing integer sequence
TP Generalized Vandermonde	$0 < x_i \nearrow$

Cost of Accuracy in TM (3)
Known results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3		n^2
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3		n^2
Vandermonde								
TP Vandermonde								
Confluent Vandermonde								
TP Confluent Vandermonde								
Vandermonde 3 Term Orth. Poly.								
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Exploit $\det(C) = \Pi_{i<j}(x_j - x_i)(y_j - y_i) / \Pi_{ij}(x_i + y_j)$

Cost of Accuracy in TM (4)
Known results + New Results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde								
TP Vandermonde								
Confluent Vandermonde								
TP Confluent Vandermonde								
Vandermonde 3 Term Orth. Poly.								
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Do GECP, apply new SVD algorithm

Cost of Accuracy in TM (5)
Known results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde	n^2							n^2
TP Vandermonde	n^2	n^3						n^2
Confluent Vandermonde								
TP Confluent Vandermonde								
Vandermonde 3 Term Orth. Poly.								
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Björck-Pereyra

Cost of Accuracy in TM (6)

Known results + New Results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2
Vandermonde	n^2						n^3	n^2
TP Vandermonde	n^2	n^3					n^3	n^2
Confluent Vandermonde								
TP Confluent Vandermonde								
Vandermonde 3 Term Orth. Poly.								
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Vandermonde = Cauchy × DFT

Cost of Accuracy in TM (7)

Known results + New Results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde	n^2						n^3	n^2
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2
Confluent Vandermonde								
TP Confluent Vandermonde								
Vandermonde 3 Term Orth. Poly.								
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Special case of TP Generalized Vandermonde

Cost of Accuracy in TM (8)
Known results + New Results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2
Confluent Vandermonde								
TP Confluent Vandermonde								
Vandermonde 3 Term Orth. Poly.								
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Can't add $x + y + z$ accurately

Cost of Accuracy in TM (9)
Known results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2
Confluent Vandermonde	n^2							n^2
TP Confluent Vandermonde	n^2	n^3		n^3				n^2
Vandermonde 3 Term Orth. Poly.								
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Higham

Cost of Accuracy in TM (10)
Known results + New Results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2
TP Confluent Vandermonde	n^2	n^3		n^3				n^2
Vandermonde 3 Term Orth. Poly.								
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Can't add $x + y + z$ accurately

Cost of Accuracy in TM (11)
Known results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2
TP Confluent Vandermonde	n^2	n^3		n^3				n^2
Vandermonde 3 Term Orth. Poly.	n^2							
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Higham

Cost of Accuracy in TM (12)

Known results + New Results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2
TP Confluent Vandermonde	n^2	n^3		n^3				n^2
Vandermonde 3 Term Orth. Poly.	n^2						n^3	
Generalized Vandermonde								
TP Generalized Vandermonde								

Proof: Poly_Vand(x) = Cauchy(x,y) $\times$ Poly_Vand(y)

Choose y as roots of Orth Poly $\Rightarrow$ Poly_Vand(y) = diagonal $\times$ orthogonal

Cost of Accuracy in TM (13)

New Results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2
TP Confluent Vandermonde	n^2	n^3		n^3				n^2
Vandermonde 3 Term Orth. Poly.	n^2						n^3	
Generalized Vandermonde	No	No	No	No	No	No	No	No
TP Generalized Vandermonde								

Proof: Can't add $x + y + z$ accurately

Cost of Accuracy in TM (14)

New Results

Matrix Type	det(A)	A^{-1}	Any minor	GENP	GEPP	GECP	SVD	NENP
Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
TP Cauchy	n^2	n^2	n^2	n^2	n^2	n^3	n^3	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2
TP Confluent Vandermonde	n^2	n^3		n^3				n^2
Vandermonde 3 Term Orth. Poly.	n^2						n^3	
Generalized Vandermonde	No	No	No	No	No	No	No	No
TP Generalized Vandermonde	$\Lambda n + n^2$	$\Lambda n^2 + n^3$	exp	Λn^2	Λn^2	exp	exp	Λn^2

- $G_{ij} = x_i^{\lambda_j+j-1}$, $0 \le \lambda_i \nearrow$
- $\Lambda = (\lambda_1 + 1) \cdot (\lambda_2 + 1)^2 \cdots (\lambda_n + 1)^2$
- Exponential speedup over previous best algorithm: $n^{\lambda_1+\cdots+\lambda_n}$
- Proof: Divide-and-conquer to evaluate Schur polynomials (see MacDonald)

Other examples in Traditional Model

- Diagonal * Totally Unimodular (TU) * Diagonal
  - TU $\Leftrightarrow$ each minor $\in \{0, \pm 1\}$
  - Poincaré: Signed incidence matrix on graph $\Rightarrow$ TU
  - Includes 2nd centered difference approximations to Sturm-Liouville equations and elliptic PDEs on uniform meshes
  - One-line change to GECP makes it accurate
- Sparse matrices with
  - Acyclic sparsity patterns, Cost $= O(n^3)$
  - Particular sparsity and sign patterns ("Total Sign Compound") Cost $= O(n^4)$
- Other Totally Positive matrices (but cost not always poly)

- What do these matrices have in common?

Traditional Model - What we can do

- Goal: evaluate homogeneous polynomial $f(x)$ accurately on domain $\mathcal{D}$
- Property A: $f = \Pi_m f_m$ where each factor f_m satisfies one of
  1. f_m of the form x_i, $x_i - x_j$ or $x_i + x_j$, or
  2. $|f_m|$ bounded away from 0 on $\mathcal{D}$
- Conjecture 1: f satisfies Prop. A iff $f(x)$ can be evaluated accurately
- Conjecture 2: f satisfies Prop. A iff $f(x)$ has a ***relative perturbation theory***:
  - relative error in output = O($\kappa_{rel}\cdot$ relative error in input)
  - $\kappa_{rel} = O(1/\min \frac{|x_i \pm x_j|}{|x_i|+|x_j|}) = O(1/$ smallest relative gap among inputs)
  - Tiny outputs often well conditioned
  - Relative perturbation theory justifies computing them!
- Intuition:
  - Everything works if $f(x)$ has factors only of forms
    x_i, $x_i - x_j$, $x_i + x_j$, positive stuff
  - Otherwise, $\forall$ algorithms $\exists$ inputs, errors that make relative error large

Bit Models of Arithmetic

- Inputs of form $x = m \cdot 2^e$, e and m integers
- size(x) = # bits used to represent x = #bits(m) + #bits(e)
- Can evaluate any rational expression accurately
  - Convert to poly/poly, using high enough precision
  - Question is cost
- Cost depends strongly on #bits(e)
  - Short Exponent Model (SEM)
    * #bits(e) = O(log(#bits(m)))
    * Equivalent to integer arithmetic
    * Can CAE many problems
  - Long Exponent Model (LEM)
    * #bits(e) and #bits(m) independent
    * Natural model for algorithm design
    * Like symbolic algebra, which is much harder

Differences between Short and Long Exponent Models - 1

- **SEM and integer arithmetic "equivalent"**
  - Represent $m \cdot 2^e$ as integer with $\#\text{bits} = \#\text{bits}(m) + e \approx \#\text{bits}(m) + 2^{\#\text{bits}(e)} = \text{poly}(\#\text{bits}(m))$
  - Any minor of any SEM matrix A computable accurately in poly time
    * Put all A_{ij} over common denominator
    * Compute each numerator, denominator exactly
    * Compute minor using Clarkson's Algorithm
  - Can do accurate linear algebra in polynomial time
- **LEM and integer arithmetic not equivalent**
  - $\Pi_{i=1}^{n}(1 + x_i)$ can have exponentially more bits if x_i LEM than SEM
  - Getting arbitrary bit of $\Pi_{i=1}^{n}(1 + x_i)$ as hard as permanent
  - Testing if an LEM matrix is singular may not be in NP
  - For efficiency, matrices need structure

Differences between Short and Long Exponent Models - 2

- Cond(A) in LEM can be exponentially larger than in SEM
  - SEM: $\log \text{cond}(A)$ is poly(size(A))
    - * Conventional algorithms using $\log \text{cond}(A)$ bits are polynomial
  - LEM: $\log \text{cond}(A)$ can be exponential in size(A)
    - * $\text{cond}(\text{diag}(2^e, 1)) = 2^e \approx 2^{2^{\#\text{bits}(e)}}$
    - * 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

Which FP Expressions can we CAE in the Long Exponent Model (LEM)?

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

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

where

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

and

$$\text{size}(r) = \#\text{bits to write down } r$$

- Theorem: We can CAE r in time poly(size(r))
  - Compute each monomial $\alpha_{ij} \cdot x_1^{e_{ij1}} \cdots x_k^{e_{ijk}}$, exactly
  - Compute $p_i(x_1, ..., x_k)$ by sorting and adding monomials, rounding
  - Compute $p_i(x_1, ..., x_k)^{e_i}$ by repeated squaring, rounding
  - Compute $\prod_{i=1}^{n} p_i(x_1, ..., x_k)^{e_i}$ by multiplying, rounding
- 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 size(r) $= O(\text{poly}(n))$
- Thm: Suppose $A_n(x)$ is polyfactorable. Then in the LEM we can CAE LU with pivoting, A^{-1}, singular values.

Cost comparison of LEM to symbolic algebra

- Cost(Accurate evaluation in Long Exponent Model) $\geq$ Cost(deciding if symbolic expression $\equiv 0$)
- Proof idea: Simulate symbolic algebra using numbers with large exponents
  - 2^a and 2^b are like indeterminates x and y, because a and b can be extracted from $2^a \cdot 2^b = 2^{a+b}$
  - Given $p(X_1, ..., X_n)$, $\exists$ numbers $x_1, ..., x_n$ such that $p \equiv 0$ iff $p(x_1, ..., x_n) = 0$
  - Cost(Accurate evaluation of p) $\geq$ Cost(deciding if $p \equiv 0$)
- Example: determinant of A each entry of which is rational

Summary of differences between Arithmetic Models

- **What can we CAE in LEM that we could not in TM?**
  - **Rational Expressions**
    - **LEM: anything in factored form can be computed accurately in polynomial time**
      - ***Not* det A where each A_{ij} independent: size is $n!$**
    - **TM: factors limited to being**
      - **x_i, $x_i + x_j$, $x_i - x_j$, or**
      - **bounded away from 0**
  - **Matrix computations**
    - **Take any $A(x)$ that we can CAE in TM, substitute $x_i = p_i(y)$**
    - **Green's matrices (inverses of tridiagonals, represented as $A_{ij} = x_i y_j$)**
- **What can we CAE in SEM that we could not in LEM?**
  - **Rational matrices with arbitrary polynomial-sized entries**

Open Questions

- **Are there FP expressions that we provably cannot CAE in LEM?**
  - $\Pi_{i=1}^{n}(1 + x_i) - \Pi_{j=1}^{n}(1 + y_j)$
  - **Determinant of general matrix**
  - **Determinant of tridiagonal matrix**
- **What changes if we have sign information?**
  - **We have accurate algorithms for all TP matrices, but not efficient**
  - **How big a class of TP matrices can we do efficiently?**
- **Differential equations**
  - **Only simplest ones understood (eg M-matrices)**
  - **What about other discretizations?**
  - **Conjecture: Accuracy depends only on geometry, not material properties**
- **Exploit sparsity for efficiency**
- **What about nonsymmetric eigenproblem?**

Conclusions

- **We have identified many classes of floating point expressions and matrix computations that permit**
  - **Accurate solutions: relative error < 1**
  - **Efficient solutions: time = poly(input size)**
- **Explored 3 natural models of arithmetic**
  - **Traditional Model (TM)**
  - **Long Exponent Model (LEM)**
  - **Short Exponent Model (SEM)**
- **New efficient algorithms for each**
- **TM $\overset{\subsetneq}{\neq}$ LEM $\overset{\subsetneq}{\neq ?}$ SEM**
- **Lots of open problems**
- **See `www.cs.berkeley.edu/~demmel` for more information**