The Complexity of Accurate Floating Point Computation

or

Can we do Numerical Linear Algebra 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

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 (1)

- Classes of FP expressions/matrices that we can CAE depends strongly on Model of FP Arithmetic

  1. Traditional ("$1 + \delta$") Model (TM for short):
     $fl(a \otimes b) = (a \otimes b)(1 + \delta)$, $|\delta| \leq \epsilon \ll 1$
     no over/underflow
  2. Bit model: inputs are $f \cdot 2^e$, with "large exponents" (LEM for short):
     "natural" model for algorithms, analysis
  3. Bit model: inputs are $f \cdot 2^e$, with "small exponents" (SEM for short):
     integers in disguise, well understood
  4. Others have been proposed (not today)
     (a) Blum/Shub/Smale
     (b) Cucker/Smale
     (c) Pour-El/Richards

Structure of Results (2)

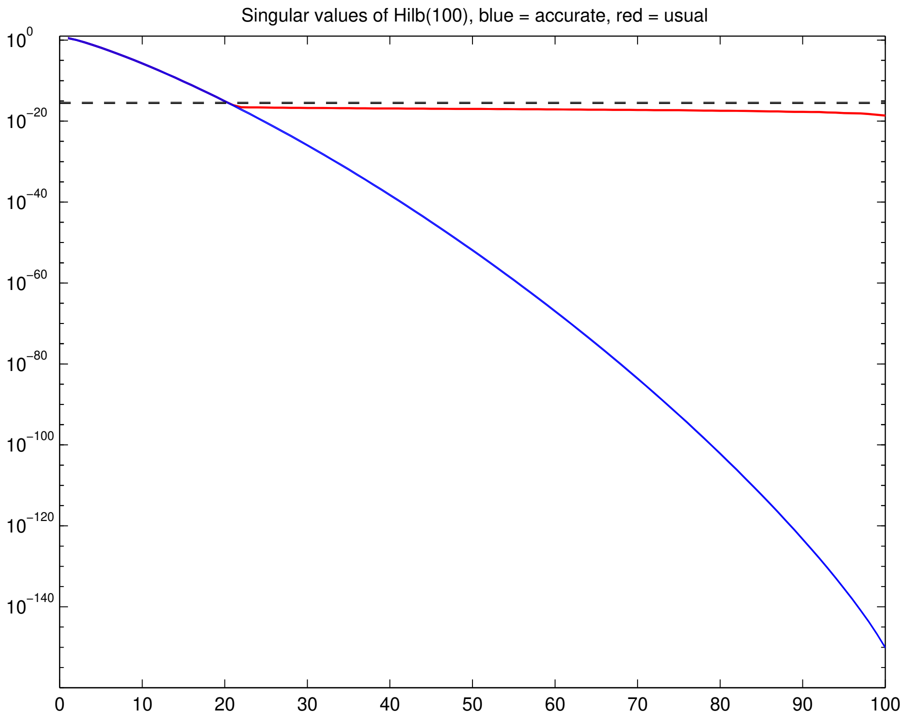
- Classes of expressions (matrices) that we can CAE are described by factorizability properties of expressions (minors of matrices)

$$\mathbf{TM} \underset{\neq}{\subset} \mathbf{LEM} \underset{\neq ?}{\subset} \mathbf{SEM}$$

- New algorithms can be exponentially faster than conventional algorithms that just use high enough precision
- Cost(CAE in LEM) $\geq$ Cost("symbolic computing")
- Many recent results (see Koev's talk too)

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



- $D = \log(\lambda_1/\lambda_n) = \log \mathrm{cond}(A)$ (here $D = 150$ digits)
- Cost of Old algorithm in high enough precision $= O(n^3 D^2)$
- Cost of New algorithm $= O(n^3)$ - independent of $\mathrm{cond}(A)$

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: $\det(A) = \pm \Pi_i U_{ii} = \pm \Pi_i R_{ii} = \Pi_i \lambda_i = \cdots$
- 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
    - Only need $O(n^2)$ or $O(n^3)$ minors
  - Singular values
    - Proof: Rank-revealing $A = LDU$, then SVD of LDU
- Similar result for QR, pseudoinverse via $\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}$, etc.
- Examine which expressions (minors) we can CAE

Accurate SVD of any rank-revealing $A = XDY^T$

- SVD is $A = U\Sigma V^T$
- Many accurate algorithms, here is simplest:
  1. Compute SVD of $DY^T = U_1 \Sigma_1 V_1^T$ using one-sided Jacobi
  2. Multiply $W = XU_1$
  3. Compute SVD of $W\Sigma_1 = U\Sigma V_2^T$ using one-sided Jacobi
  4. Multiply $V = V_1 V_2$
- To guarantee efficiency, find eigenvalues of

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

  by performing bisection on $\lambda \hat{D} - Z^{-1} Z^{-T}$
- Relative error $= O(\kappa(X) \cdot \kappa(Y))$

Why roundoff is harmless

- We want $A = U\Sigma V^T$ where $\Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_n)$
- But we compute $\bar{A} = \bar{U}\bar{\Sigma}\bar{V}^T$ where $\bar{\Sigma} = \mathrm{diag}(\bar{\sigma}_1, ..., \bar{\sigma}_n)$

Absolute (additive) Perturbations vs. **Relative (multiplicative) Perturbations**

$$\bar{A} = A + \sigma_{\max} \cdot E \qquad\qquad \bar{A} = (I + E)A$$

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

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

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

- How can we lose accuracy in computing?
  - TM: $fl(a \otimes b) = (a \otimes b)(1+\delta)$, $|\delta| \le \epsilon \ll 1$
  - OK to multiply, divide, add positive numbers
  - OK to subtract exact numbers (initial data)
  - Cancellation when subtracting approximate results dangerous:

$$\begin{array}{rl} & .12345\text{xxx} \\ - & .12345\text{yyy} \\ \hline & .00000\text{zzz} \end{array}$$

- Cauchy: $C(i,j) = 1/(x_i + y_j)$
- Fact 1: $\det(C) = \Pi_{i<j}(x_j - x_i)(y_j - y_i)/\,\Pi_{i,j}(x_i + y_j)$ - No bad cancellation
- 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 (1)

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax = b$ Forw.	$Ax = b$ Backw.
Cauchy										
TP Cauchy										
Vandermonde										
TP Vandermonde										
Confluent Vandermonde										
TP Confluent Vandermonde										
Vandermonde 3 Term Orth. Poly.										
Same + other cond.										
Generalized Vandermonde										
TP Generalized Vandermonde										

NENP = Neville Elimination (bidiagonal factorization) with No Pivoting

$Ax = b$ Forw. = solving with small forward error: $|x - \hat{x}| \le O(\epsilon)|A^{-1}| \cdot |b|$

$Ax = b$ Backw. = solving with small backward error: $\max_i \frac{|A\hat{x}-b|_i}{(|A||\hat{x}|+|b|)_i} = O(\epsilon)$

Cost of Accuracy (2)

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax = b$ Forw.	$Ax = b$ Backw.
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 a j-th orthogonal poly. in 3-term recurrence									
Same + other cond.	$0 < x_i \nearrow$, positivity conditions on 3-term recurrence									
Generalized Vandermonde	$G_{ij} = x_i^{\lambda_j + j - 1}$, λ_j nonnegative increasing integer sequence									
TP Generalized Vandermonde	$0 < x_i \nearrow$									

TP = Totally Positive (all minors nonnegative)

Cost of Accuracy (3)
Known results of others

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax = b$ Forw.	$Ax = b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3		n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3		n^2	n^2	n^2
Vandermonde										
TP Vandermonde										
Confluent Vandermonde										
TP Confluent Vandermonde										
Vandermonde 3 Term Orth. Poly.										
Same + other cond.										
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 (4)

Known results of others + New Results

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax = b$ Forw.	$Ax = b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde										
TP Vandermonde										
Confluent Vandermonde										
TP Confluent Vandermonde										
Vandermonde 3 Term Orth. Poly.										
Same + other cond.										
Generalized Vandermonde										
TP Generalized Vandermonde										

Proof: Do GECP, apply new SVD algorithm

Cost of Accuracy (5)

Known results of others

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax=b$ Forw.	$Ax=b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde	n^2							n^2		
TP Vandermonde	n^2	n^3		n^2	n^2			n^2	n^2	n^2
Confluent Vandermonde										
TP Confluent Vandermonde										
Vandermonde 3 Term Orth. Poly.										
Same + other cond.										
Generalized Vandermonde										
TP Generalized Vandermonde										

Cost of Accuracy (6)

Known results of others + New Results

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax=b$ Forw.	$Ax=b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde	n^2						n^3	n^2		
TP Vandermonde	n^2	n^3		n^3			n^3	n^2	n^2	n^2
Confluent Vandermonde										
TP Confluent Vandermonde										
Vandermonde 3 Term Orth. Poly.										
Same + other cond.										
Generalized Vandermonde										
TP Generalized Vandermonde										

Proof: Vandermonde = Cauchy $\times$ DFT

Cost of Accuracy (7)

Known results of others + New Results

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax=b$ Forw.	$Ax=b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	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	n^2	n^2
Confluent Vandermonde										
TP Confluent Vandermonde										
Vandermonde 3 Term Orth. Poly.										
Same + other cond.										
Generalized Vandermonde										
TP Generalized Vandermonde										

Proof: Use new alg for Generalized Vandermonde ...

Cost of Accuracy (8)

Known results of others + New Results

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax = b$ Forw.	$Ax = b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2	No	
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2	n^2	n^2
Confluent Vandermonde										
TP Confluent Vandermonde										
Vandermonde 3 Term Orth. Poly.										
Same + other cond.										
Generalized Vandermonde										
TP Generalized Vandermonde										

Proof: Can't add $x + y + z$ in TM

Cost of Accuracy (9)
Known results of others

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax=b$ Forw.	$Ax=b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2	No	
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2	n^2	n^2
Confluent Vandermonde	n^2							n^2		
TP Confluent Vandermonde	n^2	n^3		n^3				n^2	n^2	n^2
Vandermonde 3 Term Orth. Poly.										
Same + other cond.										
Generalized Vandermonde										
TP Generalized Vandermonde										

Cost of Accuracy (10)

Known results of others + New Results

Type of Matrix	det(A)	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax = b$ Forw.	$Ax = b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2	No	
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2	n^2	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2	No	
TP Confluent Vandermonde	n^2	n^3		n^3				n^2	n^2	n^2
Vandermonde 3 Term Orth. Poly.										
Same + other cond.										
Generalized Vandermonde										
TP Generalized Vandermonde										

Proof: Can't add $x + y + z$ in TM

Cost of Accuracy (11)
Known results of others

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax = b$ Forw.	$Ax = b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2	No	
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2	n^2	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2	No	
TP Confluent Vandermonde	n^2	n^3		n^3				n^2	n^2	n^2
Vandermonde 3 Term Orth. Poly.	n^2									
Same + other cond.	n^2	n^3							n^2	
Generalized Vandermonde										
TP Generalized Vandermonde										

Cost of Accuracy (12)

Known results of others + New Results

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax = b$ Forw.	$Ax = b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2	No	
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2	n^2	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2	No	
TP Confluent Vandermonde	n^2	n^3		n^3				n^2	n^2	n^2
Vandermonde 3 Term Orth. Poly.	n^2						n^3			
Same + other cond.	n^2	n^3					n^3		n^2	
Generalized Vandermonde										
TP Generalized Vandermonde										

Proof: See Koev's talk

Cost of Accuracy (13)

Known results of others + New Results

Type of Matrix	det(A)	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax = b$ Forw.	$Ax = b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2	No	
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2	n^2	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2	No	
TP Confluent Vandermonde	n^2	n^3		n^3				n^2	n^2	n^2
Vandermonde 3 Term Orth. Poly.	n^2						n^3			
Same + other cond.	n^2	n^3					n^3		n^2	
Generalized Vandermonde	No	No	No	No	No	No	No	No	No	
TP Generalized Vandermonde										

Proof: Can't add $x + y + z$ in TM

Cost of Accuracy (14)

Known results of others + New Results

Type of Matrix	$\det(A)$	A^{-1}	Any minor	GE NP	GE PP	GE CP	SVD	NE NP	$Ax=b$ Forw.	$Ax=b$ Backw.
Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	
TP Cauchy	n^2	n^2	n^2	n^3	n^3	n^3	n^3	n^2	n^2	n^2
Vandermonde	n^2	No	No	No	No	No	n^3	n^2	No	
TP Vandermonde	n^2	n^3	exp	n^2	n^2	exp	n^3	n^2	n^2	n^2
Confluent Vandermonde	n^2	No	No	No	No	No		n^2	No	
TP Confluent Vandermonde	n^2	n^3		n^3				n^2	n^2	n^2
Vandermonde 3 Term Orth. Poly.	n^2						n^3			
Same + other cond.	n^2	n^3					n^3		n^2	
Generalized Vandermonde	No	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	Λn^2	Λn^2

$G_{ij} = x_i^{\lambda_j + j - 1}$, $0 \le \lambda_i \nearrow$

$\Lambda = (\lambda_1 + 1) \cdot (\lambda_2 + 1)^2 \cdots (\lambda_n + 1)^2$

Previous best algorithm: $n^{\lambda_1 + \cdots + \lambda_n}$

(For Proof, see Koev's PhD thesis)

Other examples in Traditional $1 + \delta$ 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 GE makes it accurate
- M-Matrices
  - Store as off-diagonals, nonnegative row sums
  - See Koev's PhD thesis
- 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 $1+\delta$ Model - What we can do

- Goal: evaluate homogeneous polynomial $f(x)$ accurately on $\mathcal{D}$
- Property A: $f = \Pi_m f_m$ where each factor f_m satisfies one of
  - f_m of the form x_i, $x_i - x_j$ or $x_i + x_j$
  - $|f_m|$ 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)
  - $\kappa_{rel} = 1/\min \frac{|x_i \pm x_j|}{|x_i|+|x_j|}$ = 1 / smallest relative gap among inputs
  - Tiny outputs often well conditioned
    - Smallest eigenvalues often desired
    - Relative perturbation theory justifies computing them!
- Intuition: Everything works if $f(x)$ has factors only of forms
  - x_i
  - $x_i \pm x_j$
  - positive stuff

Otherwise, $\forall$ algorithms $\exists$ roundoff errors that make relative error large

Bit Models of FP Arithmetic

- Inputs of form $f \cdot 2^e$, e and f integers
- size(X) = # bits used to represent X
- size($f \cdot 2^e$) = #bits(f) + #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 Exponent Model (SEM)
    * #bits(e) = O(log(#bits(f)))
    * Equivalent to integer arithmetic
    * Can CAE many problems
  - Large Exponent Model (LEM)
    * #bits(e) and #bits(f) independent
    * "Natural" model for algorithm design
    * Algorithms work for any input magnitudes

Differences between SEM and LEM - 1

- Recall definitions for size of $f \cdot 2^e$
  - Small Exponent Model (SEM): #bits(e) = $O(\log(\#\text{bits}(f)))$
  - Large Exponent Model (LEM): #bits(e), #bits(f) independent
- SEM and "integer arithmetic" equivalent
  - Represent $f \cdot 2^e$ as integer, not pair (f, e)
  - $\#\text{bits}(f \cdot 2^e) = \#\text{bits}(f) + e \approx \#\text{bits}(f) + 2^{\#\text{bits}(e)} = \text{poly}(\#\text{bits}(f))$
- LEM and "integer arithmetic" not equivalent
  - $2^{\#\text{bits}(e)}$ exponentially larger than #bits(e)
- # bits in FP expressions much bigger for LEM than SEM
  - SEM: $\text{size}(x \cdot y) \le \text{size}(x) + \text{size}(y)$
  - LEM: $\text{size}(x \cdot y) \le \text{size}(x) \cdot \text{size}(y)$
  - The product of two n-bit numbers:

$$\begin{aligned}
\text{size}(x \cdot y) &= \text{size}\Big(\sum_{i=1}^{n} 2^{r_i} \cdot \sum_{j=1}^{n} 2^{s_j}\Big) \\
&= \text{size}\Big(\sum_{i,j=1}^{n} 2^{r_i + s_j}\Big) \\
&= n^2 \text{ bits}, \text{not } 2n \text{ bits}
\end{aligned}$$

Differences between SEM and LEM - 2

- Recall definitions for size of $f \cdot 2^e$
  - Small Exponent Model (SEM): #bits(e) = O(log(#bits(f)))
  - Large Exponent Model (LEM): #bits(e), #bits(f) indep.
- 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)
    * $\kappa(\text{diag}(2^e, 1)) = 2^e \approx 2^{2^{\#\text{bits}(e)}}$
    * 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

Differences between SEM and LEM - 3

- Recall definitions for size of $f \cdot 2^e$
  - Small Exponent Model (SEM): #bits(e) = O(log(#bits(f)))
  - Large Exponent Model (LEM): #bits(e), #bits(f) indep.
- Determinant of any SEM matrix computable exactly in poly time
  - Put all $A_{ij}(x) = P_{ij}(x)/Q_{ij}(x)$ over common denominator
  - Compute each numerator, denominator exactly
  - Compute determinant accurately in poly time using Clarkson's Alg.
  - Can do accurate linear algebra in poly time
- Getting arbitrary bit of expression in LEM very hard
  - Getting arbitrary bit of $\prod_{i=1}^{n}(1 + x_i)$ is as hard as permanent(A)
  - A null vector of singular matrix of LEM floats can have exponentially many bits
    - Testing singularity may not even be in NP!
  - Matrices need structure to CAE

How to CAE $S = \sum_{i=1}^{n} s_i$

- Suppose
  - All s_i have m-bit fractions, normalized
  - One M-bit register S available ($M > m$)
  - Let $\bar{n} = 1 + 2^{M-m} + 2^{M-2m} + \cdots + 2^{M \bmod m}$
- Algorithm for $\sum_{i=1}^{n} s_i$:
  1. Sort so $\mathrm{EXP}(s_1) \geq \mathrm{EXP}(s_2) \geq \cdots \geq \mathrm{EXP}(s_n)$
  2. $S = 0$; for $i = 1$ to n do $S = S + s_i$ ... round to nearest
- Theorem (D., Hida): Exactly one error bound below applies, is attainable:
  1. If $n \leq \bar{n}$ then relative error in S at most 1 ulp
  2. If $n = \bar{n} + 1$ and $M \geq 2m$ then relative error in S at most 2 ulps
  3. If $n = \bar{n} + 1$ and $M < 2m$ then relative error in S at most 2^{2m-M} ulps
  4. If $n \geq \bar{n} + 2$ then relative error in S can be arbitrary
- Example: $S = \sum_{i=1}^{n} x_i \cdot y_i$, $m = 2 \cdot 24 = 48$, $M = 53 \rightarrow \bar{n} = 2^5 + 1 = 33$
- Example: $S = \sum_{i=1}^{n} x_i \cdot y_i$, $m = 2 \cdot 24 = 48$, $M = 64 \rightarrow \bar{n} = 2^{16} + 1 = 65537$
- Sorting can be mostly eliminated
- Priest, Knuth, Kahan, Bohlender, Dekker, Pichat, ...

Which FP Expressions can we CAE in the Large 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
  - 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.

Differences between Models

- **What can we CAE in LEM that we could not in TM?**
  - **Rational Expressions**
    - **LEM: anything polynomial in size (in factored form) can be computed accurately in polynomial time**
    - **(*Not* det A where each A_{ij} independent: size is $n!$)**
    - **TM: Constraints on zero/pole set: inside** $\mathcal{A} = \cup_i \{x_i = 0\} \;\cup\; \cup_{i \neq j}\{x_i = x_j\} \;\cup\; \cup_{i \neq j}\{x_i = -x_j\}$
  - **Matrix computations**
    - **LEM: 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 poly-sized entries**

Cost comparison of LEM to symbolic algebra

- Cost(Accurate evaluation in Large Exponent Model) $\geq$ Cost(symbolic expression $\equiv 0$?)
- Proof idea: Simulate symbolic algebra using large exponents
  - 2^a, 2^b like x and y, because we can extract a and b from $2^a \cdot 2^b = 2^{a+b}$
  - Cost(Accurate evaluation of p) $\geq$ Cost($p \equiv 0$?)
  - Suppose $p(x_0, ..., x_{m-1})$ a symbolic polynomial with max degree $D-1$, integer coeffs of max # bits $B-1$
  - Let $X_i = 2^{B \cdot D^i}$
  - Then $p(X_0, ..., X_{m-1}) = 0$ 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 LEM?
  - $\Pi_{i=1}^{n}(1+x_i) - \Pi_{j=1}^{n}(1+y_j)$
  - Determinant of general matrix
  - Determinant of tridiagonal matrix
  - Such an example could distinguish LEM from SEM
- What does symbolic computing complexity tell us about complexity in LEM?
- 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 (M-matrices)
  - What about other discretizations?
  - Conjecture: Accuracy depends only on geometry, not material properties
- Relationship to graph-based preconditioners of Vaidya et al
- Exploit sparsity
- 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: TM, LEM, SEM**
  - **New efficient algorithms for each**
  - **TM $\subsetneq$ LEM $\subsetneq?$ SEM**
- **Lots of open problems**
- **Reports available**
  - **Upcoming ICM paper at** `www.cs.berkeley.edu/~demmel/ICM_final.ps`
  - **Koev's UC Berkeley PhD thesis at** `www.math.berkeley.edu/~plamen/a.ps`
  - **Accurate FP Addition,** `www.cs.berkeley.edu/~demmel/AccurateSummation.ps`
  - **D + Koev paper on LEM in Structured Matrices in Math, CS and Eng II, AMS, 2001 (**`www.cs.berkeley.edu/~demmel/NASC.ps`**)**
  - **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/HH02.ps`

Postdocs Available!

- One to work on Parallel Eigensolvers (Holy Grail)
- One to work on Parallel Direct Solvers (SuperLU)
- See me or email: demmel@cs.berkeley.edu

There was a numerical analyst from Nantucket ...

There was a numerical analyst from Nantucket
Who sorted his exponents with a bucket.

There was a numerical analyst from Nantucket
Who sorted his exponents with a bucket.
When asked ”How do you cope
With huge exponent scope?”

There was a numerical analyst from Nantucket
Who sorted his exponents with a bucket.
When asked "How do you cope
With huge exponent scope?"
Said: "When I see such a problem, I duck it!"