The Cost of Accurate Numerical Linear Algebra

or

Can we evaluate polynomials accurately?

James Demmel
Mathematics and Computer Science
UC Berkeley

Joint work with
Ioana Dumitriu, Olga Holtz

Plamen Koev, Yozo Hida, Ben Diament
W. Kahan, Ming Gu, Stan Eisenstat, Ivan Slapničar,
Krešimir Veselić, Zlatko Drmač

Supported by NSF and DOE
Outline

1. Motivation and Goals

2. What we can do in Traditional Model (TM) of arithmetic

3. What these example have in common:
   a condition for accurate evaluation in TM
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 \( e < 1 \)
  - \( |y_{\text{computed}} - y| \leq e \cdot |y| \)
  - \( e = 10^{-2} \) means 2 leading digits of \( y_{\text{computed}} \) 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

![Log10(eigenvalues)](image)

- Cost of Old algorithm in high enough precision = $O(n^3D^2)$ where $D = \# \text{ digits} = \log(\lambda_{\text{max}}/\lambda_{\text{min}}) = \log \text{cond}(A) = 150$ 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

- \( f_l(a \otimes b) = (a \otimes b)(1 + \delta) \) where \textit{roundoff error} \( |\delta| \leq \epsilon \ll 1 \)

- 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:
    \[
    \begin{array}{c}
    0.12345xxx \\
    - 0.12345yyy \\
    \hline
    0.00000zzz
    \end{array}
    \]

- Thm: In Traditional Model it is impossible to add \( x + y + z \) accurately
  - Proof sketch later

- Adding numbers represented as bits easier ...
Adding Numbers in Bit Model of Arithmetic

- \( x = m \cdot 2^e \) where \( m \) and \( e \) are integers, \( m \) at most \( b \) bits
- \( fl(x + y) \) is correctly rounded result
- Cancellation is obstable to accuracy and efficiency:
  - \((2^e + 1) - 2^e\) requires \( e \) bits of intermediate precision
  - Not polynomial time!
- “Sort and Sum” Algorithm for \( S = \sum_{i=1}^{n} x_i \)
  
  Sort so \(|e_1| \geq |e_2| \geq \cdots \geq |e_n|\) \(\ldots |x_1| \geq \cdots \geq |x_n|\) more than enough
  
  \( S = 0 \ldots B > b \) bits
  
  for \( i = 1 \) to \( n \)
  
  \( S = S + x_i \)

- Thm: Let \( N = 1 + 2^{B-b} + 2^{B-2b} + \cdots 2^{B \mod b} = 1 + \left\lfloor \frac{2^{B-b}}{1-2^{-b}} \right\rfloor \). Then
  - If \( n \leq N \), then \( S \) accurate to nearly \( b \) bits, despite any cancellation
  - If \( n \geq N + 2 \), then \( S \) may be completely wrong (wrong sign)
  - If \( n = N + 1 \), 2 cases, depending on whether \( s_2 \) denormal
- Ex: \( x_i \) double \((b = 53)\), \( S \) extended \((B = 64) \Rightarrow N = 2049\)
Structure of Prior Results

- 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):
   \[ fl(a \otimes b) = (a \otimes b)(1 + \delta) \text{ where } |\delta| \leq \epsilon \ll 1 \]
   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)

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

   \[ \text{TM} \subsetneq \text{LEM} \subsetneq \text{SEM} \]

- New algorithms can be exponentially faster than conventional algorithms that just use high enough precision
Structure of New Results

- All in Traditional Model (TM): $fl(a \otimes b) = (a \otimes b)(1 + \delta)$ where $|\delta| \leq \epsilon \ll 1$

- Necessary condition on polynomial $p(x)$ for existence of algorithm for accurate evaluation in TM model
  - Just depends on variety $V(p) = \{x : p(x) = 0\}$
  - Conjecture from ICM 2002 half right - not sufficient in real case!

- Goal: decision procedure to either exhibit accurate algorithm for $p$, or proof that one does not exist
  - When data complex: Simple necc. & suff. condition on $V(p)$
  - When data real: Will show main induction step
Outline

1. Motivation and Goals
2. What we can do in Traditional Model (TM) of arithmetic
3. What these example have in common:
   a condition for accurate evaluation in TM
Cost of Accuracy in TM (1)

<table>
<thead>
<tr>
<th>Matrix Type</th>
<th>det(A)</th>
<th>$A^{-1}$</th>
<th>Minor</th>
<th>GENP</th>
<th>GEPP</th>
<th>GECP</th>
<th>SVD</th>
<th>NENP</th>
<th>EVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cauchy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP Cauchy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vandermonde</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP Vandermonde</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Confluent Vandermonde</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP Confluent Vandermonde</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vandermonde</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Term Orth. Poly.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalized Vandermonde</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP Generalized Vandermonde</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Any TP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GENP/PP/CP = Gaussian Elimination with No/Partial/Complete Pivoting  
SVD = Singular Value Decomposition  
NENP = Neville Elimination (bidiagonal factorization) with No Pivoting  
EVD = Eigenvalue Decomposition
Cost of Accuracy in TM (2)

TP = Totally Positive (all minors nonnegative)

<table>
<thead>
<tr>
<th>Matrix Type</th>
<th>Expression</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cauchy</td>
<td>$C_{ij} = 1/(x_i + y_j)$</td>
<td>$x_i \nearrow, y_j \nearrow, x_1 + y_1 &gt; 0$</td>
</tr>
<tr>
<td>TP Cauchy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vandermonde</td>
<td>$V_{ij} = x_i^{j-1}, x_i$ distinct</td>
<td></td>
</tr>
<tr>
<td>TP Vandermonde</td>
<td>$0 &lt; x_i \nearrow$</td>
<td></td>
</tr>
<tr>
<td>Confluent Vandermonde</td>
<td>if some $x_i$ coincide, differentiate rows of $V$</td>
<td></td>
</tr>
<tr>
<td>TP Confluent Vandermonde</td>
<td>$0 &lt; x_i \nearrow$</td>
<td></td>
</tr>
<tr>
<td>Vandermonde 3 Term Orth. Poly.</td>
<td>$V_{ij} = P_j(x_i)$, $P_j$ orthogonal polynomial from 3-term recurrence</td>
<td></td>
</tr>
<tr>
<td>Generalized Vandermonde</td>
<td>$G_{ij} = x_i^{\lambda_i+j-1}, \lambda_j$ nonnegative increasing integer sequence</td>
<td></td>
</tr>
<tr>
<td>TP Generalized Vandermonde</td>
<td>$0 &lt; x_i \nearrow$</td>
<td></td>
</tr>
<tr>
<td>Any TP</td>
<td>Given by its Neville Factorization</td>
<td></td>
</tr>
</tbody>
</table>
## Cost of Accuracy in TM

**Known results + New Results**

<table>
<thead>
<tr>
<th>Matrix Type</th>
<th>det($A$)</th>
<th>$A^{-1}$</th>
<th>Minor</th>
<th>GENP</th>
<th>GEPP</th>
<th>GECP</th>
<th>SVD</th>
<th>NENP</th>
<th>EVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cauchy</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^3$</td>
<td>$n^3$</td>
<td>$n^2$</td>
<td></td>
</tr>
<tr>
<td>TP Cauchy</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^3$</td>
<td>$n^3$</td>
<td>$n^2$</td>
<td>$n^3$</td>
</tr>
<tr>
<td>Vandermonde</td>
<td>$n^2$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>$n^3$</td>
<td>$n^2$</td>
<td></td>
</tr>
<tr>
<td>TP Vandermonde</td>
<td>$n^2$</td>
<td>$n^3$</td>
<td>exp</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>exp</td>
<td>$n^3$</td>
<td>$n^2$</td>
<td>$n^3$</td>
</tr>
<tr>
<td>Confluent Vandermonde</td>
<td>$n^2$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
<td>$n^2$</td>
</tr>
<tr>
<td>TP Confluent Vandermonde</td>
<td>$n^2$</td>
<td>$n^3$</td>
<td>$n^3$</td>
<td></td>
<td></td>
<td></td>
<td>$n^3$</td>
<td>$n^2$</td>
<td>$n^3$</td>
</tr>
<tr>
<td>Vandermonde</td>
<td>$n^2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$n^3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Term Orth. Poly.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalized Vandermonde</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>TP Generalized Vandermonde</td>
<td>$\Lambda n^2$</td>
<td>$\Lambda n^3$</td>
<td>exp</td>
<td>$\Lambda n^2$</td>
<td>$\Lambda n^2$</td>
<td>exp</td>
<td>$\Lambda n^3$</td>
<td>$\Lambda n^2$</td>
<td>$\Lambda n^3$</td>
</tr>
<tr>
<td>Any TP</td>
<td>$n$</td>
<td>$n^3$</td>
<td>exp</td>
<td>$n^3$</td>
<td>exp</td>
<td>exp</td>
<td>$n^3$</td>
<td>0</td>
<td>$n^3$</td>
</tr>
</tbody>
</table>
Other examples in Traditional Model

- **Diagonal * Totally Unimodular * Diagonal**
  - Includes 2nd centered difference approximations to Sturm-Liouville equations and elliptic PDEs on uniform meshes
- **Sparse matrices with**
  - Acyclic sparsity patterns
  - Particular sparsity and sign patterns: “Total Sign Compound”
- **Weakly Diagonally Dominant M-Matrices**

- **What do these examples have in common?**
Outline

1. Motivation and Goals
2. What we can do in Traditional Model (TM) of arithmetic
3. What these example have in common:
   a condition for accurate evaluation in TM
What do all these examples have in common?

- **Notation**
  - \( p(x) \) is polynomial to be evaluated, \( x = (x_1, x_2, \ldots) \)
  - \( p_{\text{comp}}(x, \delta) \) is result of algorithm for \( p(x) \)
  - \( \delta = (\delta_1, \delta_2, \ldots) \) is vector of rounding errors

- **Goal:** Decide if \( \exists \) algorithm \( p_{\text{comp}}(x, \delta) \) to evaluate \( p(x) \) with small relative error on domain \( \mathcal{D} \):
  - \( \forall 0 < \eta < 1 \) ... for any \( \eta = \) desired relative error
  - \( \exists 0 < \epsilon < 1 \) ... there is an \( \epsilon = \) maximum rounding error
  - \( \forall x \in \mathcal{D} \) ... so that for all \( x \) in the domain
  - \( \forall |\delta_i| \leq \epsilon \) ... and for all rounding errors bounded by \( \epsilon \)
  - \( |p_{\text{comp}}(x, \delta) - p(x)| \leq \eta \cdot |p(x)| \) ... relative error is at most \( \eta \)

- **Given** \( p(x) \) and \( \mathcal{D} \), seek effective procedure ("compiler") to exhibit algorithm, or show one does not exist

- Not obviously Tarski-decideable: how do we express "\( \exists \) algorithm"?
Formalizing an Algorithm under Traditional Model

- **Numerical operations included**
  - Include $\pm$, $\times$, (exact) unary $-$
  - We omit $\div$ (restrictive?)

- **Comparison and Branching**
  - Assume branching on exact comparisons $a > b$, $c \leq d$, ...
  - Will sketch proof in nonbranching case

- **Determinism**
  - Is $3 + 7$ same no matter where computed?
  - Will assume nondeterministic for now

- **Available constants**
  - With $\sqrt{2}$, could compute $x^2 - 2 = (x - \sqrt{2}) \times (x + \sqrt{2})$ accurately, else not
  - Will sketch proof when no constants
  - Limits us to integer coefficients, zero constant term in $p(x)$
    * Replace $2 \times x$ by $x + x$, etc.
    * No loss of generality for homogeneous polynomials, integer coeffs
Recognizing Accuracy

- Ex: Compute \( p(x) = x_1 + x_2 + x_3 \)
  - Try \( p_{\text{comp}}(x, \delta) = ((x_1 + x_2)(1 + \delta_1) + x_3)(1 + \delta_2) \)
  - \( \text{rel\_err}(x, \delta) = \frac{p_{\text{comp}}(x, \delta) - p(x)}{p(x)} = \frac{x_1 + x_2}{x_1 + x_2 + x_3}(\delta_1 + \delta_2 + \delta_1 \cdot \delta_2) + \frac{x_3}{x_1 + x_2 + x_3}(\delta_2) \)
  - \( \forall \epsilon > 0, \text{rel\_err}(x, \delta) \) unbounded on an open subset of \((x, \delta)\) with \(|\delta_i| < \epsilon\)

- Generally: \( \text{rel\_err}(x, \delta) = \sum_\alpha \frac{p_\alpha(x)}{p(x)} \cdot \delta^\alpha \)
  - Each \( \frac{p_\alpha(x)}{p(x)} \) must be bounded near \( p(x) = 0 \)

- Ex: \( p(x) > 0 \) (positive definite) and homogeneous of degree \( d \)
  - If \( p_\alpha(x) \) also homogeneous of degree \( d \), then \( \frac{p_\alpha(x)}{p(x)} \) bounded
  - Holds if all intermediate results in \( p_{\text{comp}} \) are homogeneous
Examples

- $M_2(x, y, z) = z^6 + x^2 \cdot y^2 \cdot (x^2 + y^2 - 2 \cdot z^2)$
  - Positive definite and homogenous, easy to evaluate accurately

- $M_3(x, y, z) = z^6 + x^2 \cdot y^2 \cdot (x^2 + y^2 - 3 \cdot z^2)$
  - Motzkin polynomial, nonnegative, zero at $|x| = |y| = |z|$ if
  \[
  p = z^4 \cdot [4((x - z)^2 + (y - z)^2 + (x - z)(y - z))] + \\
  +z^3 \cdot [2(2(x - z)^3 + 5(y - z)(x - z)^2 + 5(y - z)^2(x - z) + \\
  2(y - z)^3)] + \\
  +z^2 \cdot [(x - z)^4 + 8(y - z)(x - z)^3 + 9(y - z)^2(x - z)^2 + \\
  8(y - z)^3(x - z) + (y - z)^4] + \\
  +z \cdot [2(y - z)(x - z)((x - z)^3 + 2(y - z)(x - z)^2 + \\
  2(y - z)^2(x - z) + (y - z)^3] + \\
  +(y - z)^2(x - z)^2((x - z)^2 + (y - z)^2)
  \]

  else \hspace{1cm} \ldots \hspace{1cm} 2^{|\text{vars}| - 1} \text{ more analogous cases}

- $M_4(x, y, z) = z^6 + x^2 \cdot y^2 \cdot (x^2 + y^2 - 4 \cdot z^2)$
  - Impossible to evaluate accurately
Allowable Sets

- Define basic *allowable sets*
  
  - $Z_i = \{x : x_i = 0\}$
  
  - $S_{ij} = \{x : x_i + x_j = 0\}$
  
  - $D_{ij} = \{x : x_i - x_j = 0\}$

- Def: A set is *allowable* if it can be written as an arbitrary union and intersection of basic allowable sets (plus null set, $\mathbb{R}^n$)

- We say $p(x)$ is allowable if its variety $V(p)$ is allowable
Necessary condition for existence of an Accurate Algorithm

- Theorem: A necessary condition for the existence of an accurate algorithm to evaluate $p(x)$ on $\mathbb{R}^n$ or $\mathbb{C}^n$ is that $V(p)$ be allowable.
  - Proof sketch later (if time)

- Examples
  - $p(x, y, z) = x + y + z$ not allowable (D., Koev)
  - $M_2(x, y, z) = z^6 + x^2 \cdot y^2 \cdot (x^2 + y^2 - 2 \cdot z^2)$ is allowable: $V(M_2) = \{0\}$
  - $M_3(x, y, z) = z^6 + x^2 \cdot y^2 \cdot (x^2 + y^2 - 3 \cdot z^2)$ is allowable: $V(M_3) = \{|x| = |y| = |z|\}$.
  - $M_4(x, y, z) = z^6 + x^2 \cdot y^2 \cdot (x^2 + y^2 - 4 \cdot z^2)$ is unallowable
  - $V(\det(\text{Toeplitz}))$ is unallowable $\Rightarrow$ no accurate linear algebra for Toeplitz matrices in TM: need arbitrary precision arithmetic
  - $V(\det(\text{your favorite structured matrix}))$ ...

Sufficient conditions for accurate evaluation

- Over $\mathbb{C}^n$, $V(p)$ being allowable is necessary and sufficient for accuracy
  - Proof Sketch: Can show $V(p)$ allowable $\Rightarrow p = c \cdot \prod p_i$ where each $p_i$ of form $x_j$ or $x_j \pm x_k$

- Over $\mathbb{R}^n$, $V(p)$ being allowable not a sufficient condition for accuracy:
  - $p = (u^4 + v^4) + (u^2 + v^2)(x + y + z)^2$ and $q = (u^4 + v^4) + (u^2 + v^2)(x^2 + y^2 + z^2)$ are both allowable: $V(p) = V(q) = \{u = v = 0\}$
  - But $q$ can be evaluated accurately and $p$ can’t be
  - Why: dominant term of $q$ near $V(q)$ is $q_{dom} = (u^2 + v^2)(x^2 + y^2 + z^2)$ which is allowable
  - But $p_{dom} = (u^2 + v^2)(x + y + z)^2$ is not allowable
  - Idea of inductive decision procedure: look at all ”dominant terms” near all components of $V(p)$
    * Ask if each dominant term can be evaluated accurately
    * Build accurate algorithm for $p$ by using accurate algorithms for each $p_{dom}$
    * Need Thm: $\exists$ accurate $p_{comp}$ iff $\forall p_{dom} \exists$ accurate $p_{dom,comp}$
What are ”dominant terms near V(p)”?

- Example:

\[ p = (x_1^8 + x_2^8) \cdot (x_3 + x_4 + x_5)^2 + (x_1^2x_2^4 + x_1^4x_2^2) \cdot ((x_3 - x_4)^4 + x_5^4) \]
\[ \equiv (x_1^8 + x_2^8) \cdot p_1 + (x_1^2x_2^4 + x_1^4x_2^2) \cdot p_2 \]

- \( V(p) = \{ x_1 = x_2 = 0 \} \cup \{ x_3 = x_4 = x_5 = 0 \} \) allowable

- Near \( \{ x_1 = x_2 = 0 \} \) dominant terms are
  1. \( x_2^8 \cdot p_1 \) when \( |x_1| \ll x_2^2 \)
  2. \( x_2^8 \cdot p_1 + x_1^2x_2^4 \cdot p_2 \) when \( |x_1| \approx x_2^2 \)
  3. \( x_1^2x_2^4 \cdot p_2 \) when \( x_2^2 \ll |x_1| < |x_2| \)
  4. \( (x_1^2x_2^4 + x_1^4x_2^2) \cdot p_2 \) when \( |x_1| \approx |x_2| \)
  5. \( x_1^4x_2^2 \cdot p_2 \) when \( x_1^2 \ll |x_2| < |x_1| \)
  6. \( x_1^4x_2^2 \cdot p_2 + x_1^8 \cdot p_1 \) when \( x_1^2 \approx |x_2| \)
  7. \( x_1^8 \cdot p_1 \) when \( |x_2| \ll x_1^2 \)

- In Cases 1 and 7, \( p_{dom} \) not allowable \( \Rightarrow \) no accurate algorithm

- These cases arise from examining set of exponents of \( x_1, x_2 \), namely (8, 0), (4, 2), (2, 4), (0, 8): Newton polytope
Proof Sketch that $V(p)$ allowable is necessary for accuracy (1/4):

- Def: $\text{Allow}(x)$ is the smallest allowable set containing $x$

  $\text{Allow}(x) = \mathbb{R}^n \cap (\cap_{i: x_i=0} \mathbb{Z}_i) \cap (\cap_{i,j: x_i+x_j=0} S_{ij}) \cap (\cap_{i,j: x_i-x_j=0} D_{ij})$

- Ex: $\text{Allow}((0, 1, -1, 2)) = Z_1 \cap S_{23}$

- If $V(p)$ not allowable, then

  $G(p) \equiv V(p) - \cup A$

  is nonempty, where the union is over all allowable sets $A$ contained in $V(p)$

- Def: $G(p)$ called the set of points in “general position” in $V(p)$
Proof Sketch (2/4)

- Assume no branching for simplicity
- Let \( p_{\text{comp}}(x, \delta) \) denote result of computation.
- Main Lemma: Choose any \( x \). One of following two cases must hold:
  1. \( p_{\text{comp}}(x, \delta) \) is nonzero at \( x \) for all \( \delta \) in a Zariski-open set
  2. \( p_{\text{comp}}(y, \delta) = 0 \) for all \( y \in \text{Allow}(x) \) and all \( \delta \)
- Suppose \( V(p) \) not allowable. Choose any \( x \in G(p) \subset V(p) \). Then either
  1. \( p_{\text{comp}}(x, \delta) \) is nonzero at \( x \) for all \( \delta \) in a Zariski-open set
     but \( p(x) = 0 \), so the relative error is \( \infty \)
  2. \( p_{\text{comp}}(y, \delta) = 0 \) for all \( y \in \text{Allow}(x) \) and all \( \delta \)
     but \( p(y) \neq 0 \) a.e., so the relative error is 1
- Can use continuity argument to show that relative error must be large on open set of \((x, \delta)\): i.e. large error on ”large” set
Proof Sketch (3/4)

- **Main Lemma:** Choose any $x$. One of following two cases must hold:
  1. $p_{comp}(x, \delta)$ is nonzero at $x$ for all $\delta$ in a Zariski-open set
  2. $p_{comp}(y, \delta) = 0$ for all $y \in \text{Allow}(x)$ and all $\delta$

- For simplicity, suppose no branching, no data reuse, nondeterminism
  - Implies that $p_{comp}(x, \delta)$ can be represented as a graph:
    * Source nodes representing data $x_i$, output edges connected to ...
    * Computational nodes, arranged in a tree, of following kinds:
      - 2-inputs, producing $fl(a \otimes b) = (a \otimes b)(1 + \delta_{\text{node}})$ ($\otimes \in \{+, -, \times\}$) with independent $|\delta_{\text{node}}| \leq \epsilon$ for each node
      - 1-input, producing $fl(x \otimes x) = (x \otimes x)(1 + \delta_{\text{node}})$ (note: $fl(x - x) = 0$ exactly)
      - 1-input, producing $-x$ exactly
    * Destination node, one input, no output
Proof Sketch (4/4)

• Main Lemma: Choose any $x$. One of following two cases must hold:
  1. $p_{\text{comp}}(x, \delta)$ is nonzero at $x$ for all $\delta$ in a Zariski-open set
  2. $p_{\text{comp}}(y, \delta) = 0$ for all $y \in \text{Allow}(x)$ and all $\delta$

• Def: Choose $x$. Call computational node “nontrivial” if it
  - Computes $fl(a \pm b)$, both $a$ and $b$ nonzero as polynomials in $\delta$
  - At least one of $a$ and $b$ not an input $x_i$

• Lemma: Output of all nontrivial nodes nonzero on Zariski-open set of $\delta$

• If ultimate output is from nontrivial node, done (Case 1)

• Otherwise, “trace back” zero output through tree as far as possible

• Can show (case analysis) that zero must result from one of
  - $x_i = 0$ (allowable)
  - $x_i \pm x_j = 0$ (allowable)
  - $x - x$ or $x + (-x)$ (in which case $\text{alg}(x, \delta) \equiv 0$)

• In any case, $p_{\text{comp}}(y, \delta)$ must be zero on Allow($x$) (Case 2)
Other results and Future Work

● Need to complete decision procedure

● Want to incorporate
  – Determinism (simulate deterministic machine by nondeterministic one)
  – Constants (add \( \{ x : x_i \pm \alpha = 0 \} \) to basic allowable sets for constant \( \alpha \))
  – Domain \( \mathcal{D} \) limited to (allowable?) semialgebraic sets
  – Division and rational functions
  – Other basic operation besides \( \pm, \times \)
    * How much more can we do with FMA \( (x + y \cdot z), x \cdot w - y \cdot z, \)
      \( \det(3 \times 3), \ldots \)
    * Use this to evaluate instruction sets, extended precision libraries

● Extend to interval arithmetic

● Perturbation theory
  – Conjecture: Accurate evaluation possible iff condition number can have certain simple singularities (depend on reciprocal distance to set of ill-posed problems)
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 \not\subset\subseteq LEM \not\subset\subseteq SEM

- New necessary condition for existence of accurate algorithm to evaluate \( p(x) \) in TM – working on effective decision procedure

- Lots of open problems

- For more information see
  - www.cs.berkeley.edu/~demmel
  - math.mit.edu/~plamen
Improving LAPACK and ScaLAPACK

• Proposal by J. Demmel and J. Dongarra

• Many opportunities for improvement
  – Putting more of LAPACK into ScaLAPACK
  – Better (faster and/or more accurate) algorithms
  – New functions
  – Improving ease of use
  – Performance tuning
  – Support and reliability

• Seeking suggestions via on-line survey:
  – icl.cs.utk.edu/lapack-survey.html
Better (faster and/or more accurate) algorithms

- Offer 2 "settings" for each driver:
  1. As fast as possible with "standard" accuracy
  2. As accurate as possible with "standard" speed
  3. What about memory usage?

- Consider SVD/EVD: Choice of algorithm depends on
  - values only / few vectors / many vectors,
  - left vectors / right / both

- Options for "fast as possible"
  - Successive Band Reduction (Lang et al)
  - Howell/Fulton bidiagonalization
  - One-sided bidiagonalization (Ralha et al, Barlow et al, ...)
  - Other?

- Options for "as accurate as possible"
  - Jacobi SVD (Drmač et al)
  - Symmetric EVD?