Computing the Singular Value Decomposition to High Relative Accuracy

James Demmel
Department of Mathematics
Department of Electrical Engineering and Computer Science
University of California - Berkeley
demmel@cs.berkeley.edu

Plamen Koev
Department of Mathematics
University of California - Berkeley
plamen@math.berkeley.edu

Structured Matrices In Operator Theory, Numerical Analysis, Control, Signal and Image Processing
Boulder, Colorado

June 27-July 1, 1999

Supported by NSF and DOE
INTRODUCTION

- High Relative Accuracy means computing the correct SIGN and LEADING DIGITS

- Singular Value Decomposition (SVD):
  
  \[ A = U \Sigma V^T \]

  where \( U, V \) are orthogonal,

  \[
  \Sigma = \begin{bmatrix}
  \sigma_1 \\
  \sigma_2 \\
  \vdots \\
  \sigma_n
  \end{bmatrix}
  \text{ and } \sigma_1 \geq \sigma_2 \geq \ldots \sigma_n \geq 0
  \]

- GOAL: Compute all \( \sigma_i \) with high relative accuracy, even when \( \sigma_i \ll \sigma_1 \)

- It all comes down to being able to compute determinants to high relative accuracy.
Example: 100 by 100 Hilbert Matrix

\[ H(i, j) = \frac{1}{i + j - 1} \]

- Singular values range from 1 down to \(10^{-150}\)
- **Old algorithm, New Algorithm**, both in 16 digits

\[ D = \log(\text{cond}(A)) = \log(\sigma_1/\sigma_n) \] (here \(D = 150\))
- Cost of Old algorithm = \(O(n^3D^2)\)
– Run in double, not bignums as in Mathematica
– New hundreds of times faster than Old

• When does it work? Not for all matrices ...

• Why bother?
Why do we want tiny singular values accurately?

1. When they are determined accurately by the data
   - Hilbert: $H(i, j) = 1/(i + j - 1)$
   - Cauchy: $C(i, j) = 1/(x_i + y_j)$

2. In some applications, tiny singular values matter most
   - Quantum mechanics: want lowest energy levels only
   - Elasticity: want lowest frequencies of vibration only
   - Getting the sign of the determinant right
   - Always a good idea to get accurate results if we can do it at a similar cost
Overview of Results

• Being able to compute $|\det(A)| = \prod_{i=1}^{n} \sigma_i$ to high relative accuracy is a necessary condition for accurate SVD.

• Well known similar sufficient condition for accurate $A^{-1}$: Enough to compute $n^2 + 1$ minors (Cramer’s Rule)

• New: Similar sufficient condition for accurate SVD: Enough to compute $O(n^3)$ minors

• We have identified many matrix classes whose structure permits efficient ($O(n^3)$) accurate computation of these minors.
  - Sparse Matrices, depending on sparsity pattern
  - Cauchy, Vandermonde, Unit Displacement Rank Matrices
  - “Graded” matrices (diagonal $\cdot$ ”nice” $\cdot$ diagonal)
  - Appropriately discretized ODEs, PDEs
SVD Algorithm

\[ A = XDY^T \]

**Phase 1: compute Rank Revealing Decomp (RRD)**

\[ X, Y \text{ full column rank and "well-conditioned"} \]
\[ D \text{ diagonal} \]

**Phase 2: compute SVD of an RRD**

\[ A = U\Sigma V^T \]

- **Examples of RRDs:**
  - \( A = U\Sigma V^T \), SVD itself
  - \( A = QDR \), QR decomposition with pivoting
  - \( A = LDU \), Gaussian Elimination with “complete” pivoting (GECP)

- **Fact:** Each entry of \( L, D, U \) is a quotient of minors

- **Phase 1:** GECP via (implicitly) computing accurate minors of \( A \)
  - Depends on structure of \( A \)

- **Phase 2:** Works for any RRD in \( O(n^3) \)
  - Independent of structure of \( A \)
  - Uses 1 or 2 one-sided Jacobis, matmuls

- **Relative error in singular values bounded by**
  \( O(macheps \cdot \max(\text{cond}(X), \text{cond}(Y))) \)
How do we compute $A = L \cdot D \cdot U$ for a Hilbert (or Cauchy) Matrix?

- How can we lose accuracy in computing in floating point?
  - OK to multiply, divide, add positive numbers
  - OK to subtract exact numbers (initial data)
  - Cancellation when subtracting approximate results dangerous:
    \[
    \begin{array}{c}
    .12345xxx \\
    - .12345yyy \\
    \hline
    .00000zzz
    \end{array}
    \]

- Cauchy:
  \[
  C(i, j) = \frac{1}{x_i + y_j}
  \]

\[
\text{det}(C) = \frac{\prod_{i<j}(x_j - x_i)(y_j - y_i)}{\prod_{i,j}(x_i + y_j)}
\]

  - No bad cancellation ⇒ good to most digits

- Change inner loop of Gaussian Elimination from
  \[
  C(i, j) = C(i, j) - \frac{C(i, k)C(k, j)}{C(i, i)}
  \]
  to
  \[
  C(i, j) = C(i, j)\frac{(x_i - x_k)(y_j - y_k)}{(x_k + y_j)(x_i + y_k)}
  \]

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

- $V_{ij} = x_i^{j-1}$
- $\det(V) = \prod_{1 \leq i < j \leq n}(x_i - x_j)$
  - Computable to high relative accuracy
- For SVD use fact that $V \cdot DFT = Cauchy$
  - Extends to unit-displacement rank, but not higher
- Generalized Vandermonde Matrix $G_{ij}^{\mu} = x_i^{\mu_j}, x_i > 0$
  - Ex: $\mu_j = j - 1$ is usual Vandermonde
  - $G^{\mu}$ is a submatrix of a larger Vandermonde
  - Totally positive if $0 < x_1 < \cdots < x_n$
  - SVD determined to high relative accuracy; can we compute it?
- Let $\lambda_j = \mu_j - j + 1, \lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n), |\lambda| = \sum \lambda_i$
- $\det(G^{\mu}) = \det(V) \cdot s_\lambda(x_1, \ldots, x_n)$
  - Polynomial $s_\lambda$ called Schur Function
  - Widely studied in representation theory, combinatorics, ...
  - For usual Vandermonde, $|\lambda| = 0$ and $s_\lambda = 1$

- Theorem (Plamen Koev): Cost of computing $\det(G^{\mu})$ to high relative accuracy is $n^{|\lambda|} + n^2$
- Corollary: Cost of high relative accuracy for some minors of $V$ can be exponential in $n$, alas
When is High Accuracy Possible Efficiently?

• When are all minors computable accurately?
• Depends on model of arithmetic

Model 1: \( fl(a \odot b) = (a \odot b)(1 + \text{tiny}) \)

Model 2: IEEE floating point

• Model 1
  – Ok to multiply, divide, add positives, add or subtract initial data
  – Theorem 1. Can evaluate a polynomial accurately [in \( \text{poly}(n) \) time] if it factors into a product of [\( \text{poly}(n) \)] factors each of which is
    * initial data
    * sum or difference of initial data
    * sum of [\( \text{poly}(n) \)] positives
  – Covers many examples
    * Cauchy, real Vandermonde, graded, sparse, appropriately discretized ODEs and PDEs (so far)
    * Totally positive (eg generalized Vandermonde), but not in \( \text{poly}(n) \) time
  – Proposition (D,Kahan) Can’t add three numbers accurately
    * Model 1 much weaker than Model 2
Model 2 vs Model 1

- How much stronger is Model 2 than Model 1?

<table>
<thead>
<tr>
<th>Form of $\det(A)$</th>
<th>Model 1</th>
<th>Model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Pi_{i=1}^{d} a_i$</td>
<td>$d$</td>
<td>$d$</td>
</tr>
<tr>
<td>$\Pi_{i=1}^{d} (a_i - b_i)$</td>
<td>$d$</td>
<td>$d$</td>
</tr>
<tr>
<td>$\sum_{i=1}^{t} a_i$, $a_i &gt; 0$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
<tr>
<td>$\sum_{i=1}^{t} a_i$</td>
<td>imposs.</td>
<td>$t \log t$</td>
</tr>
<tr>
<td>poly($a_i$)</td>
<td>imposs.</td>
<td>$d^2 t \log t$</td>
</tr>
<tr>
<td>($t$ terms, degree $\leq d$)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Idea: represent numbers \textit{sparsely}, not \textit{densely}

- Theorem 2. Can evaluate a polynomial accurately in $\text{poly}(n)$ time if it factors into a produce of $\text{poly}(n)$ factors each of which is has $\text{poly}(n)$ terms of $\text{poly}(n)$ degree

- Examples
  - All examples from Model 1
  - Complex Vandermonde
  - Substitute polynomials for input data in previous example
    - Ex: Cauchy with $x_i, y_j$ replaced by $p(x_i), q(y_j)$
  - Inverse of a tridiagonal

- Open Problem: what is complexity of accurate determinant of a general floating point matrix?
Conclusions and Future Work

- We can compute the SVD much more accurately than via bidiagonalization for many matrix classes
- Some algorithms will appear in LAPACK
- We only solve accurately the problem as stored in the computer, i.e. the last link of the chain:

  Real World
  \[\rightarrow\] Continuous Problem
  \[\rightarrow\] Discrete Problem
  \[\rightarrow\] Rounded Discrete Problem
  \[\rightarrow\] Solution

- Many open problems remain to extend these algorithms to new problem classes, especially
  - finite element matrices, other PDE discretizations
  - totally positive matrices
  - higher displacement rank
  - sparse problems from these classes
• Slides for this talk
  http://math.berkeley.edu/~plamen/src99.ps

• Report on overall algorithm

• Report on Cauchy, Vandermonde

• Report on Generalized Vandermonde Matrices and Schur Functions
  http://math.berkeley.edu/~plamen/schur.ps