Mixed Precision Numerical Linear Algebra for Statistics Computations

Nick Higham
Department of Mathematics
The University of Manchester

https://nhigham.com


SIAM Conference on Computational Science and Engineering
March 1, 2021
<table>
<thead>
<tr>
<th>Precision</th>
<th>Type</th>
<th>Signif (t)</th>
<th>Exp</th>
<th>Range</th>
<th>$u = 2^{-t}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>half</td>
<td>bfloat16</td>
<td>8</td>
<td>8</td>
<td>$10^{\pm38}$</td>
<td>$3.9 \times 10^{-3}$</td>
</tr>
<tr>
<td>half</td>
<td>fp16</td>
<td>11</td>
<td>5</td>
<td>$10^{\pm5}$</td>
<td>$4.9 \times 10^{-4}$</td>
</tr>
<tr>
<td>single</td>
<td>fp32</td>
<td>24</td>
<td>8</td>
<td>$10^{\pm38}$</td>
<td>$6.0 \times 10^{-8}$</td>
</tr>
<tr>
<td>double</td>
<td>fp64</td>
<td>53</td>
<td>11</td>
<td>$10^{\pm308}$</td>
<td>$1.1 \times 10^{-16}$</td>
</tr>
</tbody>
</table>

- **fp64, fp32, fp16** defined by IEEE Standard.
- **Bfloat16**: Google, Intel, Arm, NVIDIA.
## Accelerators

\[ D = C + AB \]

<table>
<thead>
<tr>
<th>Year of release</th>
<th>Device</th>
<th>Matrix dimensions</th>
<th>Input format</th>
<th>Output format</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td>Google TPU v2</td>
<td>128 × 128 × 128</td>
<td>bfloat16</td>
<td>fp32</td>
</tr>
<tr>
<td>2017</td>
<td>Google TPU v3</td>
<td>128 × 128 × 128</td>
<td>bfloat16</td>
<td>fp32</td>
</tr>
<tr>
<td>2017</td>
<td>NVIDIA V100</td>
<td>4 × 4 × 4</td>
<td>fp16</td>
<td>fp32</td>
</tr>
<tr>
<td>2018</td>
<td>NVIDIA T4</td>
<td>4 × 4 × 4</td>
<td>fp16</td>
<td>fp32</td>
</tr>
<tr>
<td>2019</td>
<td>ARMv8.6-A</td>
<td>2 × 4 × 2</td>
<td>bfloat16</td>
<td>fp32</td>
</tr>
<tr>
<td>2020</td>
<td>NVIDIA A100</td>
<td>8 × 8 × 4</td>
<td>bfloat16</td>
<td>fp32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 × 8 × 4</td>
<td>fp16</td>
<td>fp32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 × 2 × 2</td>
<td>fp64</td>
<td>fp64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 × 8 × 4</td>
<td>TensorFloat-32</td>
<td>fp64</td>
</tr>
</tbody>
</table>
Loss of definiteness in a covariance/correlation matrix is a common problem.

- Missing or inconsistent data.
- Aggregation.
- Indefinite kernels in machine learning, needing a definite similarity matrix.

Another reason:

rounding a definite matrix to lower precision.

\[ \text{fl}(A) = A + \Delta A, \quad \|\Delta A\|_2 \leq u\|A\|_2, \]

so \( \lambda_{\text{min}} \) is perturbed by \( u\lambda_{\text{max}} \).
General $A$ and $b$ given in precision $u = u_{64}$.

1. Compute LU fact’n (w/pivoting) in **mixed precs** $u_{32}$, $u_{16}$.
2. Solve $\tilde{L}\tilde{U}x_1 = b$ in **prec** $u_{32}$.
3. For $i = 1, 2, \ldots$
   - $r_i = b - Ax_i$  
     **prec** $u_r$
   - Solve $MAd_i = Mr_i$ by GMRES where
     $M = \tilde{U}^{-1}\tilde{L}^{-1}$  
     **prec** $u$
   - $x_{i+1} = x_i + d_i$  
     **prec** $u$

- **Carson & H (2017, 2018).**

- Implemented with $u_r = u$ in **MAGMA 2.5.0 (2019)**, **TCAIRS** in NVIDIA cuSOLVER library.
Performance on One NVIDIA GV100

Haidar et al. (2020). Factor 4 speedup over fp64.

Performance of solving $Ax=b$ to the FP64 accuracy

- FP16-TC->64 dhgesv
- FP32->64 dsgesv
- FP64 dgesv

Matrix size: 2k, 4k, 6k, 8k, 10k, 14k, 18k, 22k, 26k, 30k, 34k, 40k

Tflop/s vs. matrix size

Nick Higham

Mixed Precision Numerical Linear Algebra
Obvious approach

- Replace LU \((A = LU)\) by Cholesky \((A = R^T R)\).
- Replace GMRES by conjugate gradient (CG) method with symmetric (two-sided preconditioning).
Adapt to Symmetric Positive Definite $A$


**Obvious approach**

- Replace **LU** ($A = LU$) by **Cholesky** ($A = R^T R$).
- Replace **GMRES** by **conjugate gradient** (CG) method with symmetric (two-sided preconditioning).

**Flaws**

- Cholesky of $f_{16}(A)$ might fail.
- IR using CG might not converge.

*Solution to first flaw:* perturb $A$. 
Let $A = DHD$, $D = \text{diag}(a_{ii}^{1/2})$. Then $h_{ii} \equiv 1$.

- **Wilkinson (1968)**: Cholesky succeeds if $c_n \kappa_2(A) u < 1$.
- **Demmel (1989)**: Cholesky succeeds if $\lambda_{\min}(H) \gtrsim (n + 1) u$.

These results suggest

\[
\text{PertA} \quad A \leftarrow A + \Delta A_1, \quad \Delta A_1 = c_n \lambda_{\max}(A) u I.
\]

\[
\text{PertH} \quad A \leftarrow D(H + \Delta H_1)D, \quad \Delta H_1 = c'_n u I,
\]
\[
= A + \Delta A_2, \quad \Delta A_2 = c'_n u D^2.
\]

- Previous work concerned with roundoff uses **PertA**.
Consider double precision matrix

\[
A = \begin{bmatrix}
10^{40} & \times & \times \\
\times & 10^{20} & \times \\
\times & \times & 1
\end{bmatrix},
\]

and set \( c_n = c'_n = 1 \).

- **PertA**: \( \Delta A_1 \approx 10^{40} ul \) and \( a_{33} \) is lost.

- **PertH**: changes \( a_{33} \) to \( 1 + u \): a small relative perturbation.
Given spd $A \in \mathbb{R}^{n \times n}$ in precision $u$ compute approx Cholesky fact. Positive integer $c$ is a parameter.

1: $A^{(\ell)} = \text{fl}_{16}(A + c u_{16} \text{diag}(A))$
2: Attempt Cholesky fact. $A^{(\ell)} = R^T R$ in precision $u_{16}$.
3: if Cholesky failed then
4: $c \leftarrow 2c$, goto line 1
5: end if

- The Cholesky may be mixed precision.
- For fp16 need diagonal scaling to avoid overflow (H, Pranesh & Zounon, 2019).
If we replace GMRES with CG then

- Solve $MAd_i = Mr_i$ by GMRES where $M = \hat{R}^{-T}\hat{R}^{-1}$ becomes
  - Solve $\tilde{A}d_i = (\hat{R}^{-T}A\hat{R}^{-1})(\hat{R}d) = \hat{R}^{-T}r_i$ by CG.

The error analysis exploits backward stability of GMRES.

For PCG (or any Krylov solver based on 3-term recurrence) we have only (Greenbaum, 1997)

$$b'\text{err} \leq O(u) \min(\kappa_2(A)^{1/2}, \kappa_2(\hat{R}))$$
Abdelfattah, Tomov & Dongarra (2020) implement this alg for $Ax = b$ using fp32/fp16 for the Cholesky fact’n.

With $n \leq 42,000$, speedup $\leq 4.7$ over a double precision solver.
\[
\min_x \|Xb - y\|_2, \quad X \in \mathbb{R}^{m \times n}, \quad m \geq n = \text{rank}(X).
\]

statistics notation.
Extension to Least Squares Problem

\[
\min_x \|Ax - b\|_2, \quad A \in \mathbb{R}^{m \times n}, \quad m \geq n = \text{rank}(A).
\]

Solution satisfies normal equations \( A^T Ax = A^T b \).

Numerical analysts prefer QR factorization because

- Avoids loss of information in forming \( A^T A \).
- Avoids condition squaring effect.
- Normal equations method is not always backward stable.

But normal equations is faster.
Assume $A$ is \textbf{well conditioned}.

- Compute $C = A^T A$ at precision $u_f$.
- Compute the \textbf{Cholesky factorization}
  \[ C + c u_f \text{diag}(c_{ii}) = R^T R \]
  in precision $u_f$.
  (If Cholesky factorization failed $c \leftarrow 2c$ and repeat.)
- Solve
  \[ A^T A x = A^T b \]
  by \textbf{GMRES-IR}, computing $r_i = A^T (b - A x_i)$ at precision $u_r$ and round $r_i$ to precision $u$. 
IR convergence test is backward error $\leq nu$.
Max $\kappa_2(A) = 2 \times 10^4$. Precisions ($u_f, u, u_r$).

<table>
<thead>
<tr>
<th>Matrix</th>
<th>$(m, n)$</th>
<th>half, single, double</th>
<th>single, double, double</th>
</tr>
</thead>
<tbody>
<tr>
<td>divorce</td>
<td>(50,9)</td>
<td>4 (2)</td>
<td>2 (2)</td>
</tr>
<tr>
<td>Cities</td>
<td>(55,46)</td>
<td>11 (2)</td>
<td>2 (2)</td>
</tr>
<tr>
<td>ash219</td>
<td>(219,85)</td>
<td>1 (1)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>WorldCities</td>
<td>(315,100)</td>
<td>3 (1)</td>
<td>2 (2)</td>
</tr>
<tr>
<td>ash331</td>
<td>(331,104)</td>
<td>1 (1)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>ash608</td>
<td>(608,188)</td>
<td>1 (1)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>ash958</td>
<td>(958,292)</td>
<td>4 (2)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>illc1033</td>
<td>(1033,320)</td>
<td>13 (1)</td>
<td>24 (3)</td>
</tr>
<tr>
<td>well1033</td>
<td>(1033,320)</td>
<td>12 (1)</td>
<td>3 (2)</td>
</tr>
</tbody>
</table>
Conclusions

- $A \leftarrow A + cu_f \text{diag}(A)$ before Cholesky in low precision.

- **Cholesky-based GMRES-IR** works as well as LU form and gives slightly better speedups on GPUs.

- Can solve **LS problem** with GMRES-IR on normal equations for well conditioned $A$.

- Can replace GMRES with CG in practice.

- For more ill conditioned LS problems, see *Three-Precision GMRES-Based Iterative Refinement for Least Squares Problems* (Carson, H & Pranesh). Talk by **Pranesh, MS233, Thu 9.45 AM EST**.


References follow ...


Massimiliano Fasi, Nicholas J. Higham, Mantas Mikaitis, and Srikara Pranesh.  
**Numerical behavior of NVIDIA tensor cores.**  

Azzam Haidar, Harun Bayraktar, Stanimire Tomov, Jack Dongarra, and Nicholas J. Higham.  
**Mixed-precision iterative refinement using tensor cores on GPUs to accelerate solution of linear systems.**  

Nicholas J. Higham, Srikara Pranesh, and Mawussi Zounon.

Squeezing a matrix into half precision, with an application to solving linear systems.