Reproducible and Accurate Parallel Triangular Solver
Chemseddine Chohra, Philippe Langlois, David Parello

To cite this version:

HAL Id: lirmm-02427986
https://hal-lirmm.ccsd.cnrs.fr/lirmm-02427986
Submitted on 4 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Reproducible and Accurate Parallel Triangular Solver

Chemseddine Chohra\(^1\),
Philippe Langlois\(^2\) and David Parello\(^2\)

\(^1\)University 8 Mai 1945 Guelma, LabSTIC, Algeria

\(^2\)DALI-LIRMM, U. Perpignan Via Domitia, U. Montpellier, CNRS, France

July, 17\(^{th}\) 2019

ICIAM 2019, Valencia, Spain.
# Rounding Errors and Reproducibility

## Parallel Triangular Solver

## Reproducible Triangular Solvers
- RTrsv
- BinnedTrsvIR

## Accuracy and Performance

## Conclusion and Future Works
IEEE-754 Floating-Point Numbers

- Approximate real numbers on computer.
- \[ f = \pm \text{mantissa} \cdot 2^{\text{exponent}}. \]
- IEEE-754 standard defines formats and rounding modes.
- *binary64* and RTN in this talk.

Floating-Point Operations

- For \( x, y \in \mathbb{F} \) and \( x + y \notin \mathbb{F} \), \( x + y \neq x \oplus y = \text{round}(x + y) \).
- The same applies for \( \ominus, \otimes \) and \( \oslash \).

Operation Order Matters: FP Addition is not Associative

- \( a \oplus (b \oplus c) \neq (a \oplus b) \oplus c. \)
- For *binary64*’s round-off unit \( u = 2^{-53} \):
  \[ 0 = -1 \oplus (1 \oplus u) \neq (-1 \oplus 1) \oplus u = u. \]
Does Numerical Reproducibility Matter?

Numerical Reproducibility and HPC

- Reproducibility: bitwise identical results for every $p$-parallel run, $p \geq 1$
- Reproducibility $\neq$ Accuracy
- How to debug? to test? to validate? to receive legal agreements?
  - Debug: rounding errors vs. bugs? reproduce errors?
  - Validate: reproduce the reference result? the same results from one run to another?

In Practice?


Telemac2D simulation: a white plot displays a non reproducible result (Nheili et al., 2016)

1 proc., $t = 0.2$ sec.  2 proc., 0.2 sec.  2 proc., 0.8 sec.  2 proc., 1.4 sec.
Rounding Errors and Reproducibility

How to Solve Numerical Reproducibility Problems?

Strategies for Reproducibility

- Static order of operations
  - Static scheduling.
  - Deterministic Reduction.
  - Intel MKL Conditional Numerical Reproducibility (CNR).
- Pre-rounding Techniques.
  - ReprodSum and FastReprodSum (Demmel et al., 2013).
  - Indexed (Binned) floating-point format (Demmel et al., 2016).
    - Used in ReproBLAS library.
- Higher precision (Villa et al., 2009, Iakymchuk et al., 2015).
- Correctly rounded (Chohra et al., 2016).

*http://bebop.cs.berkeley.edu/reproblas/*
Rounding Errors and Reproducibility

Our Aim

RARE-BLAS (2017-)

- Reproducible, Accurately Rounded and Efficient BLAS
- Parallel BLAS 1: correctly rounded \texttt{dot} and \texttt{asum}, reproducible and faithfully rounded \texttt{nrm2}
- Parallel BLAS 2: correctly rounded \texttt{gemv}
- Accuracy vs. efficiency
  - Chose and tune summation algorithms wrt. architecture and problem constraints.
  - SIMD (AVX2-512), openMP, MPI
  - Run-time overhead ratio: $\times 1 \rightarrow \times 10$

- https://gite.lirmm.fr/rare-blas-group/rare-blas

Today: Reproducible Parallel \texttt{trsv}

- Provide a reproducible, accurate and efficient triangular solver.
- Two different approaches are presented and compared.
- Performance evaluation on CPU and Intel Xeon Phi accelerator.
Table of Contents

1 Rounding Errors and Reproducibility

2 Parallel Triangular Solver

3 Reproducible Triangular Solvers
   - RTsv
   - BinnedTrsvIR

4 Accuracy and Performance

5 Conclusion and Future Works
Parallel Triangular Solver

From Classic Forward Substitution to Parallel trsv

Triangular solver

- Given a lower triangular \( n \times n \)-matrix \( T \) and \( n \)-vector \( b \).
- Find \( x \) such that \( Tx = b \).
- Forward substitution: \( x_i = \left( b_i - \sum_{j=1}^{i-1} t_{i,j} \times x_j \right) / t_{i,i} \).
- Dependency of \( x_i \) wrt. \( x_j \), \( j < i \).
Sequential computation

\[ x_1 = b_1/t_{1,1}. \]

\[ x_i = \left( b_i - \sum_{j=1}^{i-1} t_{i,j} \times x_j \right)/t_{i,i}. \]

Sources of non reproducibility

- Dot product accumulation
- SIMD lengths
- SIMD reduction schemes
Parallel Triangular Solver

Triangular Solver: Parallel Case

Parallel Process
- \textit{trsv}: sequential.
- \textit{gemv}: parallel.

Parallel computation
- \[ x_i = (b_i - \sum_{j=1}^{r} t_{i,j} \times x_j - \sum_{j=r+1}^{2r} t_{i,j} \times x_j - \sum_{j=2r+1}^{i-1} t_{i,j} \times x_j) / t_{i,i}. \]

Sources of non reproducibility
- Dot product: partial accumulations wrt. block size \( r \)
- Accumulation order wrt. \textit{gemv} scheduling
- \textit{gemv}: SIMD lengths, SIMD reduction schemes
Table of Contents

1 Rounding Errors and Reproducibility

2 Parallel Triangular Solver

3 Reproducible Triangular Solvers
   - RTrsv
   - BinnedTrsvIR

4 Accuracy and Performance

5 Conclusion and Future Works
Reproducible Triangular Solvers

Trade-off
Efficiency vs. Accuracy vs. Reproducibility

RTrsv
- Correctly rounded $b_i - \sum_{j=1}^{i-1} t_{i,j} \times x_j$
- EFT: TwoProd, HybridSum (Zhu-Hayes, 2009)

BinnedTrsvIR
- Reproducibility: BinnedTrsv
  - Binned accumulation à la Demmel-Nguyen’s ReproBLAS.
  - Efficiency: “only” target reproducibility
- Accuracy: Iterative refinement
Table of Contents

1 Rounding Errors and Reproducibility

2 Parallel Triangular Solver

3 Reproducible Triangular Solvers
   - RTrsv
   - BinnedTrsvIR

4 Accuracy and Performance

5 Conclusion and Future Works
RTrsv: Relies on HybridSum (Zhu-Hayes, 2009)

Parallel Process

- EFT blocks use HybridSum to transform several rows in parallel.
- trsv blocks build on previous transformation to ensure correctly rounded $b_i - \sum_{j=1}^{i-1} t_{i,j} \times x_j$ and then divide it by $t_{i,i}$. 
Error-Free Transformation for summation

\[ \sum_{j=1}^{n} v_j = \sum_{j=1}^{2048} c_j \]

\[ \text{Split}(v_j, H, L) \]

\[ C_{\exp}(L) += L \]

\[ C_{\exp}(H) += H \]

[Diagram of the splitting process and summation]
Error-Free Transformation for trsv
Error-Free Transformation for trsv

$\mathbf{T}[i, :] X[:]$ → $\text{TwoProd}(t_i, j, x_j)$ → $\text{result}$ → $\text{error}$ → $C$
Error-Free Transformation for \texttt{trsv}

\begin{align*}
\mathbf{b}_i - \sum_{j=1}^{m} t_{i,j} \times x_j - \sum_{j=m+1}^{i-1} t_{i,j} \times x_j &= \sum_{j=1}^{2048} C_j
\end{align*}
First Results (detail later)

Reproducible but a bit disappointing

- Reproducible solver
- but correctly rounded accumulation \(\not\Rightarrow\) solution accuracy improvement
- with run-time overhead.
Table of Contents

1 Rounding Errors and Reproducibility

2 Parallel Triangular Solver

3 Reproducible Triangular Solvers
   - RTrsv
   - BinnedTrsvIR

4 Accuracy and Performance

5 Conclusion and Future Works
BinnedTrsv: Relies on Indexed Floating-Point Format

Parallel Process
- The input matrix is recursively decomposed into:
  - Square GEMV blocks.
  - Triangular TRSV blocks.
- Sequential small TRSV blocks
- Parallel GEMV blocks.

Reproducibility
- FP multiplications and divisions
- All accumulations are performed into a $n$-vector of Indexed FP numbers: one for every $x_i$
Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

\[ p_i = \sum_k v_k \]

\[ v_1 \]
\[ v_2 \]
\[ v_3 \]
\[ \vdots \]
\[ v_n \]
Fixed exponent range decomposition

\[ p_i = \sum_{k=1}^{v_n} v_k \]

\[ v_1 \]
\[ v_2 \]
\[ v_3 \]
\[ \vdots \]
\[ v_n \]
Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

De facto parallel sharing

\[ p_i = \sum_{k} v_k \]

Thread 0

Thread 1

Thread 2
Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

Independent first significant shrunk

\[ p_i = \sum_{k} v_k \]

\( v_1 \)
\( v_2 \)
\( v_3 \)
\( \vdots \)
\( v_n \)

Thread 0
Thread 1
Thread 2
Operand splitting: $K = 1$

$$p_i = \sum_{k} v_k$$

Thread 0

Thread 1

Thread 2
Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

Operand splitting: \( K = 2 \)

\[ p_i = \sum_{k=1}^{K} v_k \]

\[ \begin{array}{c}
\text{Thread 0} \\
\text{Thread 1} \\
\text{Thread 2}
\end{array} \]
Reproducible Triangular Solvers

Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

"Exact" thread accumulations

\[ p_i = \sum_{k} v_k \]

- Thread 0
- Thread 1
- Thread 2
"Exact" reduction and final rounding

\[ p_i = \sum_{k_i} v_k \]

Thread 0

Thread 1

Thread 2

Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)
Reproducible Iterative Refinement

1. Solve the system with \textit{BinnedTrsv} and $K = 2$.
   - Reproducibility
   - Tradeoff efficiency vs. initial accuracy

2. Compute $r^{(i)} = b - T\hat{x}$ using higher precision.
   - $\times \rightarrow \text{TwoProd}$
   - Higher precision indexed FP numbers: $K = 3$
   - Parallel and reproducible

3. Solve the system $Ad^{(i)} = r^{(i)}$ with reproducible \textit{BinnedTrsv}

4. Update $\hat{x} = \hat{x} + d^{(i)}$.

5. Repeat from 2 until $\hat{x}$ is accurate enough.
Table of Contents

1 Rounding Errors and Reproducibility

2 Parallel Triangular Solver

3 Reproducible Triangular Solvers
   - RTrsv
   - BinnedTrsvIR

4 Accuracy and Performance

5 Conclusion and Future Works
## CPU Configuration
- Dual Intel Xeon E5-2650 v2 16 cores (8 per socket).
- Memory bandwidth 59.7 GB/s.

## Many-core Accelerator
- Intel Xeon Phi 7120 accelerator, 60 cores.
- Memory bandwidth 352 GB/s.

## Compiler and Options
- Intel compiler (17.0.1)
- Intel OpenMP 5.0
- `-O3 -fp-model double -fp-model strict -funroll-all-loops`
  - `-fp-model double` : rounds intermediate results to 53-bit precision
  - `-fp-model strict` : disable contractions
Accuracy and Performance Experiments

Experiments

- **System size**
  - Accuracy: $n = 1000$
  - Run-time: $n \in [10000, 15000]$

- **Conditioning**
  - $\text{Cond}(T, x) = \frac{\| \|T^{-1}\| T\|_\infty \|x\|_\infty}{\|x\|_\infty}$
  - Accuracy: $\text{Cond} \in [10^5, 10^{15}]$
  - Run-time: $\text{Cond} = 10^8$

- **Reference solution**: $\tilde{x} = \text{MPFR}(T^{-1}b)$

- **Relative error**
  - $\frac{\| \tilde{x} - \hat{x} \|_\infty}{\|\tilde{x}\|_\infty}$

- **Normalized Residual**
  - $\frac{\|b - T\hat{x}\|_\infty}{\|b\|_\infty}$

Challenging solutions

- **Accuracy**: Intel MKL Trsv (b64)
- **High accuracy**: XBLAS double-doubled Trsv
- **Performance**: Intel MKL Trsv
Accuracy: correctly rounded (RTrsv) vs. reproducible (BinnedTrsv) dot prods → slightly but not significantly better

More accuracy: similar and classic iterative refinement effect

Residual: no condition effect, nor solution accuracy significance, slight effect of the correctly rounded version (RTrsv)
### Reproducible Solvers: Performance Results

#### Run-time overhead ratio vs. MKL Trsv, Cond = $10^8$.

<table>
<thead>
<tr>
<th></th>
<th>Sequential</th>
<th>Parallel CPU</th>
<th>Parallel Accelerator</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTrsv</td>
<td>$\times 10$</td>
<td>$\times 5$</td>
<td>$\times 5 - 10$</td>
</tr>
<tr>
<td>BinnedTrsv</td>
<td>$\times 2$</td>
<td>$\times 1$</td>
<td>$\times 0.7 - 1.2$</td>
</tr>
<tr>
<td>BinnedTrsvIR</td>
<td>$\times 15 - 20$</td>
<td>$\times 8 - 10$</td>
<td>$\times 2$</td>
</tr>
</tbody>
</table>

Note: no accelerator benefit (trsv dependencies – as it)

---

- Parallel solutions scale well
- Reproducibility: from no over-cost to very reasonable cost with BinnedTrsv
- Accuracy cost: not free (iterative refinement) but interesting for accelerator
# Table of Contents

1. Rounding Errors and Reproducibility

2. Parallel Triangular Solver

3. Reproducible Triangular Solvers
   - RTrsv
   - BinnedTrsvIR

4. Accuracy and Performance

5. Conclusion and Future Works
Conclusion

- Numerical reproducibility for SIMD, multi-core, many-core architectures
- Accuracy: similar to XBLAS
- Run-time performance (vs. MKLTrsv):

<table>
<thead>
<tr>
<th></th>
<th>Sequential</th>
<th>Parallel CPU</th>
<th>Parallel Accelerator</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTrsv</td>
<td>×10</td>
<td>×5</td>
<td>×5 – 10</td>
</tr>
<tr>
<td>BinnedTrsvIR</td>
<td>×15 – 20</td>
<td>×8 – 10</td>
<td>×2</td>
</tr>
</tbody>
</table>

Future Works

- Higher level BLAS: compute bound algorithms ⇒ need for other strategies
- Auto tuning to easy optimization
- Build a reproducible LAPACK upon RARE-BLAS