Reliable and sustainable computations: Can we effectively achieve both?

Roman Iakymchuk\textsuperscript{1,2}

joint work with
José Aliaga (UJI), Stef Graillat (Sorbonne),
and Pablo Oliveira (Paris-Saclay)

\textsuperscript{1}Umeå University, Sweden
\textsuperscript{2}Sorbonne University, France
riakymch@cs.umu.se

4th Workshop on Scientific Computing in Sweden (SwedComp22)
Chalmers, Göteborg
October 26-27, 2022
How do we verify numerical reliability of our parallel programs?
Numerical reliability

How do we verify numerical reliability of our parallel programs?

Comparison against

- Sequential
- MATLAB
- MPFR (single threaded)
How do we verify numerical reliability of our parallel programs?

Comparison against
- Sequential
- MATLAB
- MPFR (single threaded)

Initiatives
- (bitwise) reproducibility (Demmel, RI, Ogita, Ozaki, Langois)
- ACM Artifact Review and Badging
  - Repeatability (Same team, same experimental setup)
  - Reproducibility (Different team, same experimental setup)
  - Replicability (Different team, different experimental setup)
- Correctness workshop at SC
Motivation
Accuracy and Reproducibility of Preconditioned Conjugate Gradient

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Residual</th>
<th>MPFR</th>
<th>Original 1 core</th>
<th>Original 48 cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x1.19f179eb7f032p+49</td>
<td>0x1.19f179eb7f033p+49</td>
<td>0x1.19f179eb7f033p+49</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0x1.f86089ece9f75p+38</td>
<td>0x1.f86089ece5bd4p+38</td>
<td>0x1.f86089eceaf76p+38</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0x1.fc59a29d329ffp+28</td>
<td>0x1.fc59a29d3599ap+28</td>
<td>0x1.fc59a29d32d1bp+28</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0x1.74f5ccc211471p+22</td>
<td>0x1.74f5ccc1d03cbp+22</td>
<td>0x1.74f5ccc201246p+22</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>40</td>
<td>0x1.7031058eb2e3ep-19</td>
<td>0x1.7031058dd6bcfp-19</td>
<td>0x1.7031058eaf4c2p-19</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>0x1.4828f76bd68afp-23</td>
<td>0x1.4828f76d1aa3p-23</td>
<td>0x1.4828f76bd71ap-23</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>0x1.8646260a70678p-26</td>
<td>0x1.8646260a2dae8p-26</td>
<td>0x1.8646260a6da06p-26</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>0x1.13fa97e2419c7p-33</td>
<td>0x1.13fa97e1e76bfp-33</td>
<td>0x1.13fa97e240f7cp-33</td>
<td></td>
</tr>
</tbody>
</table>

Residuals for a matrix with \( \text{cond}(A) = 10^{12} \). The matrix is from the finite-difference method of a 3D Poisson’s equation with 27 stencil points, \( n=4,019,679 \). \(^a\)

<table>
<thead>
<tr>
<th>Matrix</th>
<th>( \text{cond}(A) )</th>
<th>MPI@MN4</th>
<th>MPI+OMP@MN4</th>
<th>MPI</th>
<th>MPI+OMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>gyro_k</td>
<td>( 1.10e + 09 )</td>
<td>16,557</td>
<td>16,064</td>
<td>16,518</td>
<td>16,623</td>
</tr>
</tbody>
</table>

Iterations till convergence with \( \text{tol} = 10^{-8} \) for the gyro_k matrix from SuiteSparse

\(^a\)RI et al. Reproducibility Strategies for Parallel Preconditioned Conjugate Gradient. JCAM, 371, 2020, 112697
Computer arithmetic approximates real numbers with their finite representations.
Computer arithmetic approximates real numbers with their finite representations.

**Issues**

- Floating-point arithmetic suffers from rounding errors.
- Floating-point operations (+, ×) are commutative but non-associative.

\[
(-1 + 1) + 2^{-53} \neq -1 + (1 + 2^{-53}) \quad \text{in double precision}
\]

- Consequence: results of floating-point computations depend on the order of computation.
Computer arithmetic approximates real numbers with their finite representations

**Issues**

- Floating-point arithmetic suffers from rounding errors
- Floating-point operations (+, ×) are commutative but non-associative
  \[
  (-1 + 1) + 2^{-53} \neq -1 + (1 + 2^{-53}) \quad \text{in double precision}
  \]
- Consequence: results of floating-point computations depend on the order of computation

- **Reproducibility** – ability to obtain identical and accurate results from run-to-run on the same input data on the same or different architectures
"Infinite" precision: reproducible independently from the inputs

→ Example: Kulisch accumulator (=16 FLOPs)
“Infinite” precision: reproducible independently from the inputs

→ Example: **Kulisch accumulator** (=16 FLOPs)

- **Fixed FP Expansions** (FPE) with **Error-Free Transformations** (EFT)
- Augmented operations, part of IEEE 754-2019 (work well on a set of relatively close numbers)

**Algorithm 1 (Dekker and Knuth)**

Function $[r, s] = 2\text{sum}(a, b)$

1: $r \leftarrow a + b$
2: $z \leftarrow r - a$
3: $s \leftarrow (a - (r - z)) + (b - z)$

**Algorithm 2 ($|a| \geq |b|$)**

Function $[r, s] = \text{fast2sum}(a, b)$

1: $r \leftarrow a + b$
2: $z \leftarrow r - a$
3: $s \leftarrow b - z$
- VRP enables efficient computation ... of iterative linear algebra kernels
- Augmented accuracy inside
- A dedicated register to store up to 32 scalars with up to 512 bits of mantissa
- VRP programming model is meant for smooth integration with legacy scientific libraries such as BLAS, MAGMA and linear solver libraries

Source: https://www.european-processor-initiative.eu/accelerator/
ExBLAS: Parallel Reduction

Highlights of the Algorithm

- Based on FPE with EFT and Kulisch accumulator
- Suitable for CPUs, GPUs, Xeon Phi
- Guarantees "inf" precision → bit-wise reproducibility

---

S. Collange, RI et al. *Numerical Reproducibility for the Parallel Reduction on Multi- and Many-Core Architectures*. ParCo, 49, 2015, 83-97
### Preconditioned BiCGStab

\[ Ax = b \]

**while** \((\tau > \tau_{\text{max}})\)

<table>
<thead>
<tr>
<th>Step</th>
<th>Operation</th>
<th>Kernel</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>( v := Ap )</td>
<td>SPMV</td>
<td>Allgatherv</td>
</tr>
<tr>
<td>S2</td>
<td>( \alpha := \langle r_0, r \rangle / \langle r, v \rangle )</td>
<td>DOT product</td>
<td>Allreduce</td>
</tr>
<tr>
<td>S3</td>
<td>( s := s - \alpha v )</td>
<td>AXPY</td>
<td>-</td>
</tr>
<tr>
<td>S4</td>
<td>( \tau := |s| )</td>
<td>DOT product</td>
<td>Allreduce</td>
</tr>
<tr>
<td>S5</td>
<td>( x := x + \alpha p )</td>
<td>AXPY</td>
<td>-</td>
</tr>
<tr>
<td>S6</td>
<td>( s := M^{-1}s )</td>
<td>Apply preconditioner</td>
<td>-</td>
</tr>
<tr>
<td>S7</td>
<td>(...)</td>
<td>SPMV</td>
<td>Allgatherv</td>
</tr>
<tr>
<td>S8</td>
<td>( \omega := \langle v, s \rangle / \langle v, v \rangle )</td>
<td>2 DOT product</td>
<td>Allreduce</td>
</tr>
<tr>
<td>S9</td>
<td>( x := x + \omega s )</td>
<td>AXPY</td>
<td>-</td>
</tr>
<tr>
<td>S10</td>
<td>( r := s - \omega v )</td>
<td>AXPY-like</td>
<td>-</td>
</tr>
<tr>
<td>S11</td>
<td>( \beta := \alpha / \omega &lt; r_0, r &gt; )</td>
<td>DOT product</td>
<td>Allreduce</td>
</tr>
<tr>
<td>S12</td>
<td>( p := r + \beta (p - \omega v) )</td>
<td>2 AXPY-like</td>
<td>-</td>
</tr>
<tr>
<td>S13</td>
<td>( p := M^{-1}p )</td>
<td>Apply preconditioner</td>
<td>-</td>
</tr>
<tr>
<td>S14</td>
<td>( \tau := |r| )</td>
<td>DOT product</td>
<td>Allreduce</td>
</tr>
</tbody>
</table>

**end while**
Re-assuring numerical reliability

Sources of non-reproducibility

- **parallel reduction**: dot product with MPI_Allreduce
- **compiler auto-replacement** of $ax + b$ in favor of $fma$ (axpy)

$\rightarrow a \times b + c \times d \times e$ with or without $fma$ (spmv)
Re-assuring numerical reliability

Sources of non-reproducibility

- **parallel reduction**: dot product with MPI_Allreduce
- **compiler auto-replacement** of $ax + b$ in favor of $fma$ (axpy)
  $\rightarrow a \ast b + c \ast d \ast e$ with or without $fma$ (spmv)

Solutions

- **Combined arithmetic solutions, programmability effort, and sequential executions**
- **careful initialization**
  $\rightarrow b = Ad \; d = \frac{1}{\sqrt{N}} (1, \ldots, 1)^T \rightarrow b = Ad$ and $b = \frac{1}{\sqrt{N}} b$
- **axpy and spmv** rely on intrinsics for $fma$
- **accurate and reproducible dot**
  $\rightarrow$ ExBLAS-based approach
  $\rightarrow$ FPE of size eight and early-exit
Reproducibility: required precision

\[ \text{nnz} = 26, 198 \]
\[ \text{cond}(A) = 9.0e + 15 \]

\[ \text{nnz} = 1, 021, 159 \]
\[ \text{cond}(A) = 1.1e + 09 \]

Reproducible PBiCGStab: convergence

Residual history for rdb3200l (18.8K nnz) $tol = 10^{-6}$

SkyLake nodes: 2x14-core Intel Xeon Gold 6132 CPU @2.6 GHz
Reproducible PBiCGStab: performance

**Strong scaling** for s3dkq4m2 (4.4M nnz) $tol = 10^{-6}$

SkyLake nodes: 2x14-core Intel Xeon Gold 6132 CPU @2.6 GHz
Reproducible PBiCGStab: performance

3D Poisson’s equation with 27 stencil points, **16M nnz**, \( tol = 10^{-8} \)
perturbed matrix \((1 - 10^{-4})\) aka the e-type model

**Strong scaling** on 32 SkyLake nodes w 2x14 cores each
Numerical reliability → Strategy → Revision of algorithms
Numerical reliability $\rightarrow$ Strategy $\rightarrow$ Revision of algorithms

**Sustainability success metrics**

- time-to-solution
- energy-to-solution
- robustness
Sustainability

**Numerical reliability → Strategy → Revision of algorithms**

**Sustainability success metrics**
- time-to-solution
- energy-to-solution
- robustness

**Workplan**
1. Arithmetic tool applied to code → optimized binary
2. iff \( gain \geq 5\% \), apply algorithmic solutions
3. conduct probabilistic (aka optimistic) error analysis
VerifiCarlo: first step

- a tool for debugging and assessing FP precision and reproducibility
- VPREC is a backend to emulate variable FP representations

---

The Newton-Raphson method

---

\[\text{Pablo Oliveira et al. Automatic exploration of reduced floating-point representations in iterative methods. Euro-Par 2019}\]
Conclusion

- numerically reliable algorithm → combined algorithmic and programming strategies
- All inner-product-based operations are correctly-rounded
- Reproducible PBiCGStab showed 2.45x and 2x overhead
- Drew strategies for sustainable computations

Future Work

- Studying a class of proxies with VPREC
- Probabilistic error analysis
- Testing on hardware with stochastic rounding
Thank you for your attention!

→ This research was partially supported by the EU H2020 MSCA-IF Robust project (No. 842528)

→ Participation at SwedComp was kindly provided via eSSENCE
References

David Goldberg
What every computer scientist should know about floating-point arithmetic

David H. Bailey
Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers

Georg Hager
Fooling the masses with performance results on parallel computers