Hierarchical approach for deriving a reproducible LU factorization

Roman Iakymchuk

joint work with

Sylvain Collange, David Defour, Erwin Laure, Enrique S. Quitana Ortí and Stef Graillat

1KTH Royal Institute of Technology, CSC, CST/PDC
riakymch@kth.se

RAIM 2017, Oct 24-26, 2017
Lyon, France
### BLAS-1 [1979]

- $y := y + \alpha x$
- $\alpha := \alpha + x^T y$

- $\alpha \in \mathbb{R}; x, y \in \mathbb{R}^n$

### BLAS-2 [1988]

- $A := A + xy^T$
- $y := A^{-1} x$

- $A \in \mathbb{R}^{n \times n}; x, y \in \mathbb{R}^n$

### BLAS-3 [1990]

- $C := C + AB$
- $C := A^{-1} B$

- $A, B, C \in \mathbb{R}^{n \times n}$
Linear Algebra Libraries

BLAS-1 [1979]:
\[ y := y + \alpha x \]
\[ \alpha := \alpha + x^T y \]
\[ \alpha \in \mathbb{R}; \ x, y \in \mathbb{R}^n \]
\[ y \in \mathbb{R}^n \]

BLAS-2 [1988]:
\[ A := A + xy^T \]
\[ y := A^{-1} x \]
\[ A \in \mathbb{R}^{n \times n}; \ x, y \in \mathbb{R}^n \]

BLAS-3 [1990]:
\[ C := C + AB \]
\[ C := A^{-1} B \]
\[ A, B, C \in \mathbb{R}^{n \times n} \]

Basic Linear Algebra Subprograms (BLAS)

Refer. BLAS
MKL, cuBLAS
OpenBLAS
ATLAS
LU Factorization

Ax = b \Rightarrow A = LU
LU Factorization

\[ A = LU \]

Diagram:

- \( q \) followed by \( N \) with TRSM
- \( M \) followed by \( LU \) and \( GEMM \)
- \( A \)
LU Factorization

\[ A = LU \]
Motivation
Unblocked LU Factorization

Variant 1
\[ a_{01} := L_{00}^{-1} a_{01} \]
\[ a_{10}^T := a_{10}^T U_{00}^{-1} \]
\[ \alpha_{11} := \alpha_{11} - a_{10}^T a_{01} \]

Variant 2
\[ a_{10}^T := a_{10}^T U_{00}^{-1} \]
\[ \alpha_{11} := \alpha_{11} - a_{10}^T a_{01} \]
\[ a_{12}^T := a_{12}^T - a_{10}^T A_{02} \]

Variant 3
\[ a_{01} := L_{00}^{-1} a_{01} \]
\[ \alpha_{11} := \alpha_{11} - a_{10}^T a_{01} \]
\[ a_{21} := \frac{a_{21} - A_{20} a_{01}}{\alpha_{11}} \]

Variant 4
\[ \alpha_{11} := \alpha_{11} - a_{10}^T a_{01} \]
\[ a_{21} := \frac{a_{21} - A_{20} a_{01}}{\alpha_{11}} \]
\[ a_{12}^T := a_{12}^T - a_{10}^T A_{02} \]

Variant 5
\[ a_{21} := a_{21} / \alpha_{11} \]
\[ A_{22} := A_{22} - a_{21} a_{12}^T \]

Roman Iakymchuk (KTH)
Outline

1. Accuracy and Reproducibility of FP Operations
2. Exact Multi-Level Parallel Reduction
3. ExBLAS and Reproducible LU
4. Performance Results
5. Conclusions and Future Work
Outline

1. Accuracy and Reproducibility of FP Operations
2. Exact Multi-Level Parallel Reduction
3. ExBLAS and Reproducible LU
4. Performance Results
5. Conclusions and Future Work
## Problems

- Floating-point arithmetic suffers from **rounding errors**
- Floating-point operations \((+,-,\times,/)\) are commutative but **non-associative**

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

Problems

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

\[ 2^{-53} \neq 0 \text{ in double precision} \]
Accuracy and Reproducibility

Problems

- Floating-point arithmetic suffers from rounding errors
- Floating-point operations ($+, \times$) 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

- Results computed by performance-optimized parallel floating-point libraries may be often inconsistent: each run returns a different result

Reproducibility – ability to obtain bit-wise identical results from run-to-run on the same input data on the same or different architectures
Sources of Non-Reproducibility

- **Changing Data Layouts:**
  - Data partitioning
  - Data alignment

- **Changing Hardware Resources**
  - Number of threads
  - Fused Multiply-Add support
  - Intermediate precision (64 bits, 80 bits, 128 bits, etc)
  - Data path (SSE, AVX, GPU warp, etc)
  - Number of processors
  - Network topology
1. Accuracy and Reproducibility of FP Operations
2. Exact Multi-Level Parallel Reduction
3. ExBLAS and Reproducible LU
4. Performance Results
5. Conclusions and Future Work
Fixed FP expansions (FPE) with Error-Free Transformations

Example: double-double or quad-double (Briggs, Bailey, Hida, Li) (work well on a set of relatively close numbers)

**Algorithm 1** FPE of size 2 (Dekker and Knuth)

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

1. \( r \leftarrow a + b \)
2. \( z \leftarrow r - a \)
3. \( s \leftarrow (a - (r - z)) + (b - z) \)
Fixed FP expansions (FPE) with Error-Free Transformations

Example: double-double or quad-double (Briggs, Bailey, Hida, Li)
(work well on a set of relatively close numbers)

**Algorithm 1** FPE of size 2 (Dekker and Knuth)

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

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

“Infinite” precision: reproducible independently from the inputs

Example: Kulisch accumulator (considered inefficient)
Parallel algorithm with 5-levels

Suitable for today’s parallel architectures

Based on FPE with EFT and Kulisch accumulator

Guarantees “inf” precision

→ bit-wise reproducibility
Level 1: Filtering

Input numbers

Level 1 (Filtering)

Thread 1
- EFT
- FP Expansion (register)
- Underflow?

Thread 2
- EFT
- FP Expansion (register)
- Underflow?

Thread n
- EFT
- FP Expansion (register)

Level 2 (Private SuperAccumulation)

Level 3 (Scalar SuperAccumulation)

Level 4 (Parallel Reduction)

Level 5 (Rounding)
Level 2 and 3: Scalar Superaccumulator

Input numbers

Thread 1
- EFT
- FP Expansion (register)
- Underflow?

Thread 2
- EFT
- FP Expansion (register)
- Underflow?

Level 1 (Filtering)
- Underflow?
- Private SuperAccumulator

Level 2 (Private SuperAccumulation)
- ...

Level 3 (Scalar SuperAccumulation)
- ...

Level 4 (Parallel Reduction)
- ...

Level 5 (Rounding)
- ...
Level 4 and 5: Reduction and Rounding

Level 1 (Filtering)
- Input numbers
  - Thread 1
    - EFT
    - FP Expansion (register)
    - Underflow?
  - Thread 2
    - EFT
    - FP Expansion (register)
    - Underflow?
  - Thread n
    - EFT
    - FP Expansion (register)

Level 2 (Private SuperAccumulation)

Level 3 (Scalar SuperAccumulation)

Level 4 (Parallel Reduction)

Level 5 (Rounding)
1. Accuracy and Reproducibility of FP Operations
2. Exact Multi-Level Parallel Reduction
3. ExBLAS and Reproducible LU
4. Performance Results
5. Conclusions and Future Work
ExBLAS Status

- **ExBLAS-1**: \(\text{ExSUM}^a, \text{ExSCAL}, \text{ExDOT}, \text{ExAXPY}, \ldots\)
- **ExBLAS-2**: \(\text{ExGER}, \text{ExGEMV}, \text{ExTRSV}, \text{ExSYR}, \ldots\)
- **ExBLAS-3**: \(\text{ExGEMM}, \text{ExTRSM}, \text{ExSYR2K}, \ldots\)

\(^a\text{Routines in blue are already in ExBLAS}\)

#### BLAS-1 routines

- Some are virtually built upon \(\text{ExSUM}\)
  - For instance, \(\text{ExDOT} = \text{TwoProd} + \text{ExSUM}\)
  - \(\text{TwoProd}(a,b)\):
    1: \(res \leftarrow a \cdot b\),
    2: \(err \leftarrow \text{FMA}(a, b, -res)\)
- The others are embarrassingly parallel
  - \(\text{ExAXPY} = \text{FMA}(\alpha, x[i], y[i])\)
ExSCAL

\[ x := \alpha \times x \rightarrow \text{correctly rounded and reproducible} \]
**ExSCAL**

- $x := \alpha \ast x \rightarrow$ correctly rounded and reproducible
- Within LU: $x := 1/\alpha \ast x \rightarrow$ not correctly rounded
ExBLAS Highlights (2/2)

ExSCAL

- $x := \alpha \times x \rightarrow$ correctly rounded and reproducible
- Within LU: $x := 1/\alpha \times x \rightarrow$ not correctly rounded
- ExInvSCAL: $x := x/\alpha \rightarrow$ correctly rounded and reproducible

ExGER

- General case: $A := \alpha \times x \times y^T + A$
**ExSCAL**

- \( x := \alpha \times x \rightarrow \text{correctly rounded and reproducible} \)
- Within LU: \( x := 1/\alpha \times x \rightarrow \text{not correctly rounded} \)
- ExInvSCAL: \( x := x/\alpha \rightarrow \text{correctly rounded and reproducible} \)

**ExGER**

- General case: \( A := \alpha \times x \times y^T + A \)
- Within LU: \( A := x \times y^T + A \). Using FMA \( \rightarrow \text{correctly rounded and reproducible} \)
An unblocked LU Factorization: Variant 5

LU Factorization

\[ \pi_1 := PivIndex \left( \frac{\alpha_{11}}{a_{21}} \right) \quad \text{(max)} \]

\[ \left( \frac{\alpha_{11}}{a_{21}} \right) := P(\pi_1) \left( \frac{\alpha_{11}}{a_{21}} \right) \quad \text{(swap)} \]

\[ a_{21} := a_{21}/\alpha_{11} \quad \text{(scal)} \]

\[ A_{22} := A_{22} - a_{21}a_{12}^T \quad \text{(ger)} \]

\[ 3 \times 3 \text{ partitioning of } A \]
Matrix-Vector Product

\[ y := \alpha Ax + \beta y \]

Based on ExDOT

\[ \text{TwoProd}(a, b) \]
1. \( r \leftarrow a \times b \)
2. \( s \leftarrow fma(a, b, -r) \)

\[ fma(a, b, c) = a \times b + c \]
for $i = 0 : b\text{l}sz : n$ do
  for $k = i : i + b\text{l}sz$ do
    for $j = 1 : k - 1$ do
      $[r, e] \leftarrow TwoProd(l_{kj}, -x_j)$
      ExpansionAccumulate($r$)
      ExpansionAccumulate($e$)
    end for
    ExpansionAccumulate($b_k$)
    $\hat{s} \leftarrow RNDN(acc(k))$
    $x_k \leftarrow \frac{\hat{s}}{l_{kk}}$
  end for
  for $k = i + b\text{l}sz : n$ do
    for $j = i : i + b\text{l}sz$ do
      $[r, e] \leftarrow TwoProd(l_{kj}, -x_j)$
      ExpansionAccumulate($r$)
      ExpansionAccumulate($e$)
    end for
  end for
end for
Unblocked left-looking algorithm of LU

**jik or jki variant**

\[
\begin{pmatrix}
\frac{a_{01}}{\alpha_{11}} \\
\frac{\alpha_{11}}{a_{21}}
\end{pmatrix} := P(p_0) \begin{pmatrix}
\frac{a_{01}}{\alpha_{11}} \\
\frac{\alpha_{11}}{a_{21}}
\end{pmatrix} \quad \text{(swap)}
\]

- \( a_{01} := L_{00}^{-1} a_{01} \quad \text{(trsv)} \)
- \( \alpha_{11} := \alpha_{11} - a_{10}^T a_{01} \quad \text{(dot)} \)
- \( a_{21} := a_{21} - A_{20} a_{01} \quad \text{(gemv)} \)

\[\pi_1 := \text{PivIndex} \left( \frac{\alpha_{11}}{a_{21}} \right) \quad \text{(max)}\]

\[
\begin{pmatrix}
\frac{\alpha_{11}}{a_{21}}
\end{pmatrix} := P(\pi_1) \begin{pmatrix}
\frac{\alpha_{11}}{a_{21}}
\end{pmatrix} \quad \text{(swap)}
\]

- \( a_{21} := a_{21} / \alpha_{11} \quad \text{(scal)} \)

3 × 3 partitioning of \( A \)

\[
\begin{array}{ccc}
i & A_{00} & a_{01} & A_{02} \\
1 & a_{10}^T & \alpha_{11} & a_{12}^T \\
p & A_{20} & a_{21} & A_{22}
\end{array}
\]
1. Accuracy and Reproducibility of FP Operations
2. Exact Multi-Level Parallel Reduction
3. ExBLAS and Reproducible LU
4. Performance Results
5. Conclusions and Future Work
Parallel Reduction
Data-Dependent Performance on NVIDIA Tesla K20c

$n = 67e06$

![Graph showing performance comparison of different parallel FP sums with varying dynamic ranges.](image)

- Parallel FP Sum
- Demmel fast
- Superacc
- FPE2 + Superacc
- FPE3 + Superacc
- FPE4 + Superacc
- FPE8 + Superacc
- FPE8EE + Superacc

Gacc/s vs. Dynamic range
GEMV: \( y := \alpha Ax + \beta y \)
Matrix-Vector Product

Accuracy

GEMV: $y := Ax$

- Preserve every bit of information
- Correctly-rounded

$\text{cond}(A, x) = \frac{||A|| \cdot ||x||}{||A \cdot x||}$
Triangular Solver
Performance Scaling on NVIDIA Tesla K420

\[
\text{TRSV: } Ax = b
\]

- Parallel DTRSV
- Superacc
- ExTRSV

- Blocked ExTRSV
- Internal ExGEMV
- Based on ExDOT

Normalized time vs. Matrix size [n]
Triangular Solver

Accuracy

TRSV: \( Ax = b \)

1: \( x_1 \leftarrow fl(b_1/l_{11}) \)
2: for \( i = 2 \rightarrow n \) do
3: \( s \leftarrow b_i \)
4: for \( j = 1 \rightarrow i - 1 \) do
5: \( s \leftarrow s - l_{ij}x_j \)
6: end for
7: \( x_i \leftarrow fl(RNDN(s)/l_{ii}) \)
8: end for

\[
\text{cond}(A, x) = \frac{\|A^{-1}\|A\|x\|_\infty}{\|x\|_\infty}
\]
LU Factorization (1/2)

Performance Scaling on NVIDIA Tesla K420

\[ A = LU \]

Matrix size \([m = n]\)

Parallel DLU
Superacc
ExLU

\(jik\) variant of LU

\[
\begin{align*}
  \text{swap()} & \\
  a_{01} & \leftarrow L_{00}^{-1} a_{01} & \text{trsv} \\
  \alpha_{11} & \leftarrow \alpha_{11} - a_{10}^T a_{01} & \text{dot} \\
  a_{21} & \leftarrow a_{21} - A_{20} a_{01} & \text{gemv} \\
  \text{max()} & \\
  \text{swap()} & \\
  a_{21} & \leftarrow a_{21}/\alpha_{11} & \text{scal}
\end{align*}
\]
$A = LU$

variant 5 of LU

\[
\begin{align*}
\max() \\
\text{swap()} \\
a_{21} & := a_{21}/\alpha_{11} \\
A_{22} & := A_{22} - a_{21}a_{12}^T
\end{align*}
\]

Roman Iakymchuk (KTH)

Reproducible unblocked LU

October 24-26, 2017
LU Factorization

Accuracy

\[ A = LU \]

- Slightly better accuracy than DLU
- But, always reproducible
1 Accuracy and Reproducibility of FP Operations
2 Exact Multi-Level Parallel Reduction
3 ExBLAS and Reproducible LU
4 Performance Results
5 Conclusions and Future Work
Conclusions and Future Work

Conclusions

- Leveraged a long accumulator and EFTs to design reproducing and correctly-rounded ExSUM and ExDOT
- Provided an extension to InvSCAL to perform division directly
- Developed a reproducible and accurate ExGEMV
- Derived two unblocked variants, e.g. $ji_k$, of the LU factorisation with partial pivoting
- Provide bit-wise reproducible results independently from
  - Data permutation, data assignment, partitioning/blocking
  - Thread scheduling
  - Reduction trees

Future directions
- Reproducible solution of linear systems
- Application of our implementations in real-world codes

Roman Iakymchuk (KTH)

Reproducible unblocked LU

October 24-26, 2017
Conclusions and Future Work

Conclusions

- Leveraged a long accumulator and EFTs to design **reproducible and correctly-rounded** ExSUM and ExDOT
- Provided an **extension to InvSCAL** to perform division directly
- Developed a reproducible and accurate ExGEMV
- Derived **two unblocked variants**, e.g. $jik$, of the LU factorisation with partial pivoting
- Provide **bit-wise reproducible** results independently from
  - Data permutation, data assignment, partitioning/blocking
  - Thread scheduling
  - Reduction trees

Future directions

- Reproducible solution of linear systems
- Application of our implementations in real-world codes
Thank you for your attention!

URL: https://exblas.lip6.fr

<table>
<thead>
<tr>
<th>ExBLAS Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ExBLAS-1:</strong></td>
</tr>
<tr>
<td>ExSUM(^a), ExSCAL, ExDOT, ExAXPY, ...</td>
</tr>
<tr>
<td><strong>ExBLAS-2:</strong></td>
</tr>
<tr>
<td>ExGER, ExGEMV, ExTRSV, ExSYR, ...</td>
</tr>
<tr>
<td><strong>ExBLAS-3:</strong></td>
</tr>
<tr>
<td>ExGEMM, ExTRSM, ExSYR2K, ...</td>
</tr>
</tbody>
</table>

\(^a\)Routines in blue are already in ExBLAS