MUMPS workshop @ CINES

25 June 2025

Mixed precision algorithms in numerical linear algebra

Theo Mary

Sorbonne Université, CNRS, LIP6

Floating-point landscape

	Signif. bits	Exp. bits	Range ($f_{\max}/f_{\min}$)	Unit roundoff u
fp128	113	15	$2^{32766} \approx 10^{9863}$	$2^{-114} \approx 1 \times 10^{-34}$
fp64	52	11	$2^{2046} \approx 10^{616}$	$2^{-53} \approx 1 \times 10^{-16}$
fp32	23	8	$2^{254} \approx 10^{76}$	$2^{-24} \approx 6 \times 10^{-8}$
tfloat32	10	8	$2^{254} \approx 10^{76}$	$2^{-11} \approx 5 \times 10^{-4}$
fp16	10	5	$2^{30} \approx 10^{9}$	$2^{-11} \approx 5 \times 10^{-4}$
bfloat16	7	8	$2^{254} \approx 10^{76}$	$2^{-8} \approx 4 \times 10^{-3}$
fp8 (E4M3)	3	4	$2^{15} \approx 3 \times 10^{4}$	$2^{-4} \approx 6 \times 10^{-2}$
fp8 (E5M2)	2	5	$2^{30} \approx 10^{9}$	$2^{-3} \approx 1 \times 10^{-1}$
fp6 (E2M3)	3	2	$2^{3} \approx 8$	$2^{-4} \approx 6 \times 10^{-2}$
fp6 (E3M2)	2	3	$2^{7} \approx 128$	$2^{-3} \approx 0.125$
fp4 (E2M1)	1	2	$2^{3} \approx 8$	$2^{-2} \approx 0.25$

Lower precisions:

- ☺ Faster, consume less memory and energy
- ☹ Lower accuracy and narrower range
- $\Rightarrow$ Mixed precision algorithms

Standard model of FPA:

For any x such that $|x| \in [f_{\min}, f_{\max}]$,
$\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \le u$

Survey

Acta Numerica (2022), pp. 347–414
doi:10.1017/S0962492922000022

Mixed precision algorithms in numerical linear algebra

Nicholas J. Higham
Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK
E-mail: nick.higham@manchester.ac.uk

Theo Mary
Sorbonne Université, CNRS, LIP6, Paris, F-75005, France
E-mail: theo.mary@lip6.fr

https://bit.ly/mixed-survey

CONTENTS

Mixed precision strategies

1. Iterative refinement
   Run baseline algorithm in low precision, refine result to high accuracy
2. Multiword arithmetic
   Emulate high precision with low precision
3. Memory accessors
   Decouple the storage (low) precision and the compute (high) precision
4. Adaptive precision
   Adapt the precision of each instruction to the problem/input at hand
5. Discussion
   Comparison of strengths and weaknesses

Iterative refinement: main principle

Iterative refinement for $Ax = b$

1: Compute an initial approximation x
2: **repeat**
3: $\quad r = b - Ax$
4: $\quad$ Solve $Ac \approx r$
5: $\quad x = x + c$
6: **until** convergence

- If $Ac \approx r$ is solved with precision ε:
  - Attainable accuracy independent of ε
  - Convergence rate $\propto \kappa(A)\varepsilon$
- Can use direct, iterative, or any kind of approximate solvers

  E. Carson and N. J. Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. SISC 2018.

Iterative refinement: main principle

Iterative refinement for $Ax = b$

1: Compute an initial approximation x
2: **repeat**
3: $\quad r = b - Ax$
4: $\quad$ Solve $Ac \approx r$
5: $\quad x = x + c$
6: **until** convergence

- If $Ac \approx r$ is solved with precision ε:
  - Attainable accuracy independent of ε
  - Convergence rate $\propto \kappa(A)\varepsilon$
- Can use direct, iterative, or any kind of approximate solvers
  E. Carson and N. J. Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. SISC 2018.

General technique also applicable to

- Nonlinear solvers (Newton's method)
  F. Tisseur. Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. SIMAX 2001.
- Least-squares
  E. Carson, N. J. Higham, and S. Pranesh. Three-Precision GMRES-Based Iterative Refinement for Least Squares Problems. SISC 2020.
- Eigenvalue decomposition
  J. J. Dongarra, C. B. Moler, and J. H. Wilkinson. Improving the accuracy of computed eigenvalues and eigenvectors. SINUM 1983.
- Singular value decomposition
  J. J. Dongarra. Improving the accuracy of computed singular values. SISSC 1983.
- Low-rank approximations
  M. Baboulin, O. Kaya, T. M., and M. Robeyns. Mixed precision iterative refinement for low-rank matrix and tensor approximations. SISC 2025.

Iterative refinement: success stories

LU-based IR

1: $A \approx LU$ in low precision
2: $x = U^{-1}L^{-1}b$ in low precision
3: **repeat**
4: $\quad r = b - Ax$ in high precision
5: $\quad c = U^{-1}L^{-1}r$ in low precision
6: $\quad x = x + c$ in high precision
7: **until** convergence

P. R. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. M., and B. Vieublé. Combining sparse approximate factorizations with mixed-precision iterative refinement. TOMS 2023.

Results with fp32 BLR(ε)-MUMPS solver + fp64 refinement

Matrix	n	$\kappa(A)$	ε	Its	Time	Memory
ElectroPhys	10M	10^1	10^{-6}	3	5.2×	3.7×
tminlet	3M	10^7	10^{-6}	7	4.2×	2.9×
CarBody	25M	10^{13}	F	F	F	F

Iterative refinement: success stories

GMRES-based IR

1: $A \approx LU$ in low precision
2: $x = U^{-1}L^{-1}b$ in low precision
3: **repeat**
4: $\quad r = b - Ax$ in high precision
5: $\quad$ Solve $Ac \approx r$ with LU-preconditioned GMRES in mixed precision
6: $\quad x = x + c$ in high precision
7: **until** convergence

P. R. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. M., and B. Vieublé. Combining sparse approximate factorizations with mixed-precision iterative refinement. TOMS 2023.

P. R. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. M., and B. Vieublé. Five-precision GMRES-based iterative refinement. SIMAX 2024.

Results with fp32 BLR(ε)-MUMPS solver + fp64 refinement

Matrix	n	$\kappa(A)$	ε	Its	Time	Memory
ElectroPhys	10M	10^1	$10^{-6} \rightarrow$ same	$3 \rightarrow$ same	$5.2\times \rightarrow$ same	$3.7\times \rightarrow$ same
tminlet	3M	10^7	$10^{-6} \rightarrow 10^{-4}$	$7 \rightarrow 69$	$4.2\times \rightarrow 3.3\times$	$2.9\times \rightarrow 3.4\times$
CarBody	25M	10^{13}	F $\rightarrow 10^{-8}$	F $\rightarrow 23$	F $\rightarrow 0.7\times$	F $\rightarrow 1.9\times$

Multiword arithmetic: main principle

Goal: compute $C = AB$ to high accuracy

- Step 1: Compute the multiword decompositions

$$A \approx \sum_{i=1}^{w} \alpha_i A_i \qquad \text{and} \qquad B \approx \sum_{j=1}^{w} \beta_j B_j$$

where the words A_i and B_j are stored in low precision $\Rightarrow$ decomposition error

- Step 2: compute

$$C = \sum_{i+j \le k} \alpha_i \beta_j A_i B_j$$

for some k (e.g., $k = 2w$ or $k = w$) $\Rightarrow$ accumulation error

Multiword arithmetic: success stories

fp32 emulation

- NVIDIA Tensor Cores fp16 matrix multiplication with fp32 accumulation
- fp16-TC/fp32 speed ratio:

Volta	Ampere	Hopper	Blackwell
8×	16×	15×	56×

- Compute $A \approx A_1 + A_2$ and $B \approx B_1 + B_2$
  $\Rightarrow$ decomposition error $= 2^{-22}$
- Compute $C \approx A_1B_1 + A_1B_2 + A_2B_1$ with fp32 accumulation
  $\Rightarrow$ accumulation error $\propto 2^{-24}$
- Three products $\Rightarrow$ large speedups
- bfloat16×6 or bfloat16×9 also possible. See
  M. Fasi, N. J. Higham, F. Lopez, T. M. and M. Mikaitis. Matrix Multiplication in Multiword Arithmetic: Error Analysis and Application to GPU Tensor Cores. SISC 2023.
  and references therein

Multiword arithmetic: success stories

fp64 emulation

- Ozaki scheme: decompose A and B such that $A_i B_j$ can be computed exactly
  K. Ozaki, T. Ogita, S. Oishi, and S. M. Rump. Error-free transformations of matrix multiplication by using fast routines of matrix multiplication and its applications. Numer. Algorithms 2012.
- fp16 and int8 words both possible, int8 usually more efficient
  H. Ootomo, K. Ozaki, and R. Yokota. DGEMM on integer matrix multiplication unit. IJHPCA 2024.
- Number of products quite large ($\sim$35), but int8/fp64 speed ratio is 112$\times$
- Ozaki scheme II uses multimodular arithmetic to reduce number of products ($\sim$16)
  K. Ozaki, Y. Uchino, and T. Imamura. Ozaki Scheme II: A GEMM-oriented emulation of floating-point matrix multiplication using an integer modular technique. Preprint 2025.
- Number of products must depend on dynamical range of values; otherwise componentwise error can be large for badly scaled matrices.
  A. Abdelfattah, J. Dongarra, M. Fasi, M. Mikaitis, and F. Tisseur. Analysis of floating-point matrix multiplication computed via integer arithmetic. Preprint 2025.

Memory accessors: main principle

- Decouple storage and compute precisions: data is stored (compressed) in low precision and accessed (decompressed) back to high precision to be computed on/with
  H. Anzt, G. Flegar, T. Grützmacher, and E. S. Quintana-Ortí. Toward a modular precision ecosystem for high-performance computing. IJHPCA 2019.
- Lowering storage precision reduces memory consumption and volume of data transfers ⇒ faster memory-bound computations
  T. Grützmacher, H. Anzt, and E. S. Quintana-Ortí. Using Ginkgo's memory accessor for improving the accuracy of memory-bound low precision BLAS. Software: Practice and Experience 2023.
- Compression may use custom formats without hardware support, e.g., floating-point numbers with truncated mantissa
  D. Mukunoki, M. Kawai, and T. Imamura. Sparse Matrix-Vector Multiplication with Reduced-Precision Memory Accessor MCSoC 2023.

Memory accessors: success stories

- Tensor Cores implement mixed precision matrix multiply–accumulate $C \leftarrow C + AB$, where A and B are stored in, and C is accumulated in, fp32 or fp16
- Accumulating the update operations $A_{ij} \leftarrow A_{ij} - L_{ik}U_{kj}$ of LU factorization in fp32 reduces the error bound from nu_{16} to $2u_{16} + nu_{32}$

  P. Blanchard, N. J. Higham, F. Lopez, T. M., and S. Pranesh. Mixed Precision Block Fused Multiply-Add: Error Analysis and Application to GPU Tensor Cores. SISC 2020.
- Significant accuracy boost, but performance limited by storage/data transfers
  $\Rightarrow$ store matrix in fp16, and preserve accuracy by accumulating in fp32 buffers

$$B_{ij} = \sum_k L_{ik}U_{kj}, \quad A_{ij} \leftarrow A_{ij} - B_{ij}$$

F. Lopez and T. M.. Mixed Precision LU Factorization on GPU Tensor Cores: Reducing Data Movement and Memory Footprint. IJHPCA 2023.

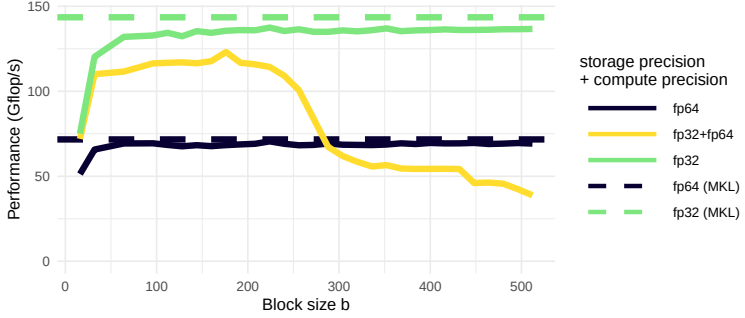
	fp16	fp16/fp32 Tensor Cores	
		fp32 storage	fp16 storage
Accuracy	1×10^{-3}	1×10^{-5}	3×10^{-5}
Speed (TFLOPS)	—	50	140

Memory accessors: success stories

At what data granularity should we use memory accessors?

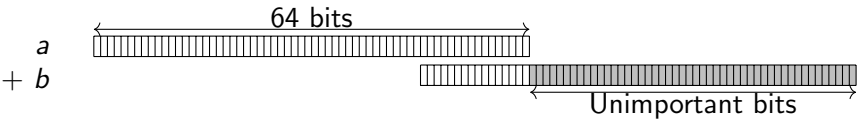
- Too small (e.g., variable-wise) ⇒ need to rewrite all the code ☹
- Too large (e.g., matrix-wise) ⇒ accessed data does not fit into fast memory, inefficient ☹
- Just right (e.g., block-wise) ⇒ blocks fit into fast memory and computations can use BLAS ! ☺

P. R. Amestoy, A. Jego, J.-Y. L'Excellent, T. M., and G. Pichon. BLAS-based Block Memory Accessor with Applications to Mixed Precision Sparse Direct Solvers. Preprint 2025.



Adaptive precision: main principle

- Not all variables/operations need the same precision!
  Example:



$\Rightarrow$ Here, b can be stored and computed in low precision

- Adaptive precision algorithms exploit this observation by dynamically selecting the minimal precision for each variable/operation, depending on the data and on the prescribed accuracy ε

Adaptive precision: success stories

- Goal: compute $y = Ax$, where A is a sparse matrix, with a prescribed accuracy ε
- Given p available precisions $u_1 < \varepsilon < u_2 < \ldots < u_p$, define partition

$$A = \sum_{k=1}^{p} A^{(k)}, \qquad a_{ij}^{(k)} = \begin{cases} \mathrm{fl}_k(a_{ij}) & \text{if } |a_{ij}| \in (\varepsilon\|A\|/u_k, \varepsilon\|A\|/u_{k+1}] \\ 0 & \text{otherwise} \end{cases}$$

  $\Rightarrow$ the precision of each element is chosen inversely proportional to its magnitude

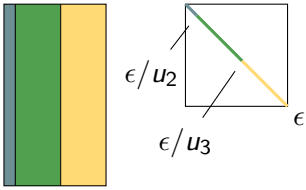
  S. Graillat, F. Jézéquel, T. M., and R. Molina. Adaptive precision sparse matrix-vector product and its application to Krylov solvers. SISC 2024.

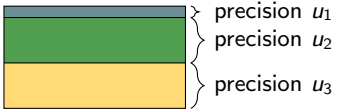
- Example on Long_Coup_dt6 matrix with $\varepsilon = 2^{-53}$ and 7 precisions:

fp64	fp56	fp48	fp40	fp32	fp24	fp16	drop
0.05%	2%	25%	25%	4%	20%	14%	10%

Performance impact: 1.5× storage reduction $\Rightarrow$ 1.4× time reduction

Adaptive precision: success stories



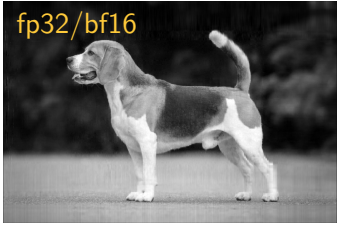


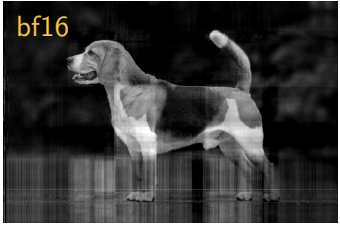
$\Rightarrow$ the precision of each singular vector is chosen inversely proportional to its singular value

P. R. Amestoy, O. Boiteau, A. Buttari, M. Gerest, F. Jézéquel, J.-Y. L'Excellent, and T. M.. Mixed Precision Low Rank Approximations and their Application to Block Low Rank LU Factorization. IMAJNA 2022.

A. Buttari, T. M., and A. Pacteau. Truncated QR factorization with pivoting in mixed precision. SISC 2025.

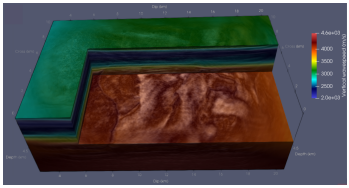
- Image of size 1057×1600 and of rank 191 (with $\varepsilon = 0.04$)
- In fp32/bf16, only 13 columns in fp32





Adaptive precision: success stories

- Adastra MUMPS4FWI project led by WIND team
- Application: Gorgon Model, reservoir 23km x 11km x 6.5km, grid size 15m, Helmholtz equation, 25-Hz
- Complex matrix, 531 Million dofs, storage(A)=220 GBytes;
- FR cost: flops for one LU factorization= 2.6×10^{18};
  Estimated storage for LU factors= 73 TBytes



(25-Hz Gorgon FWI velocity model)

FR (Full-Rank); BLR with $\varepsilon = 10^{-5}$; 48 000 cores (500 MPI $\times$ 96 threads/MPI)
FR: fp32; Adaptive precision BLR: 3 precisions (32bits, 24bits, 16bits) for storage

LU size (TBytes)			Flops		Time BLR + Mixed (sec)			Scaled Resid.
FR	BLR	+adapt.	FR	BLR+adapt.	Analysis	Facto	Solve	BLR+adapt.
73	34	26	2.6×10^{18}	0.5×10^{18}	446	5500	27	7×10^{-4}

Discussion

- **Accuracy:** how far is the computed result from the exact one?
- **Performance:** how much does it cost to reach this result?
- **Robustness:** for what range of inputs is the method guaranteed to work?

	Iterative Refinement	Multiword Arithmetic	Memory Accessor	Adaptive Precision
Accuracy	Rigorously controlled	Rigorously controlled	Rigorously controlled	Rigorously controlled
Performance	High if it converges quickly	High on GPUs and if compute-bound	High if memory-bound	Data-dependent
Robustness	$\kappa(A)$ not too large	Black box	Application-specific	Black box

Discussion

- **Accuracy:** how far is the computed result from the exact one?
- **Performance:** how much does it cost to reach this result?
- **Robustness:** for what range of inputs is the method guaranteed to work?

	Iterative Refinement	Multiword Arithmetic	Memory Accessor	Adaptive Precision
Accuracy	Rigorously controlled	Rigorously controlled	Rigorously controlled	Rigorously controlled
Performance	High if it converges quickly	High on GPUs and if compute-bound	High if memory-bound	Data-dependent
Robustness	$\kappa(A)$ not too large	Black box	Application-specific	Black box

Thanks! Questions?