SIAM CSE23

2nd March 2023

Adaptive Precision Sparse Matrix–Vector Product and its Application to Krylov Solvers

Roméo Molina
LIP6, Sorbonne Université
IJCLab, CNRS

Joint work with
Stef Graillat, **Fabienne Jézéquel**, and **Theo Mary**

Today's floating-point landscape

		Number of bits Signif. (t)	Exp.	Range	$u = 2^{-t}$
fp128	quadruple	113	15	$10^{\pm 4932}$	1×10^{-34}
fp64	double	53	11	$10^{\pm 308}$	1×10^{-16}
fp32	single	24	8	$10^{\pm 38}$	6×10^{-8}
fp16	half	11	5	$10^{\pm 5}$	5×10^{-4}
bfloat16		8	8	$10^{\pm 38}$	4×10^{-3}
fp8 (e4m3)	quarter	4	4	$10^{\pm 2}$	6×10^{-2}
fp8 (e5m2)		3	5	$10^{\pm 5}$	1×10^{-1}

- Low precision increasingly supported by hardware
- **Great benefits:**
  - Reduced **storage**, data movement, and communications
  - Reduced **energy** consumption ($5\times$ with fp16, $9\times$ with bfloat16)
  - Increased **speed** ($16\times$ on A100 from fp32 to fp16/bfloat16)
- **Some limitations too:**
  - Low accuracy (large u)
  - Narrow range

Mixed precision algorithms

Mix several precisions in the same code with the goal of

- Getting the performance benefits of low precisions
- While preserving the accuracy and stability of high precision

Various terminologies, various approaches: Mixed precision, Multiprecision, Adaptive precision, Variable precision, Transprecision, Dynamic precision, . . .

Mixed precision algorithms

Mix several precisions in the same code with the goal of

- Getting the performance benefits of low precisions
- While preserving the accuracy and stability of high precision

Various terminologies, various approaches: Mixed precision, Multiprecision, Adaptive precision, Variable precision, Transprecision, Dynamic precision, . . .

How to select the right precision for the right variable/operation?

⇒ My PhD thesis area: **Precision tuning**, autotuning based on the source code.

- ○ **PROMISE [Graillat & al.'19] based on CADNA [Vignes'93]**

▲ Does not need any understanding of what the code does

▼ Does not have any understanding of what the code does

Adaptive precision algorithms

This work:
another point of view, **exploit as much as possible the knowledge we have about the code**

Given an algorithm and a prescribed accuracy ϵ, adaptively select the minimal precision for each computation

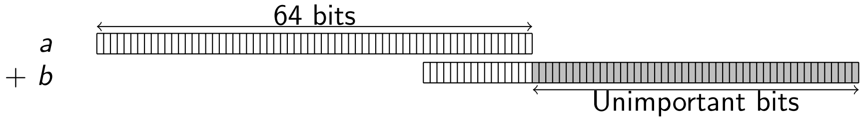
Adapting the precision to the data at hand

$\Rightarrow$ **Why does it make sense to make the precision vary?**

Adapting the precision to the data at hand

⇒ **Why does it make sense to make the precision vary?**

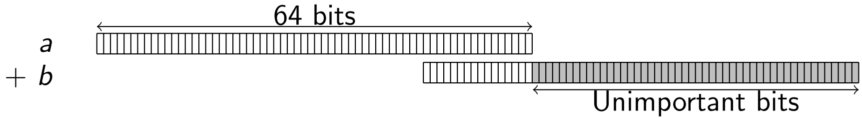
- Because not all computations are equally "important"!
  Example:



Adapting the precision to the data at hand

⇒ **Why does it make sense to make the precision vary?**

- Because not all computations are equally "important"!
  Example:



⇒ **Opportunity for mixed precision:** adapt the precisions to the data at hand by storing and computing "less important" (usually smaller) data in lower precision

Adaptive precision algorithms

Mixed precision algorithms in numerical linear algebra, section 14 [Higham & Mary (2022)]
⇒ adaptive precision algorithms, an emerging subclass

- Anzt, Dongarra, Flegar, Higham, and Quintana-Orti, *Adaptive precision in block-Jacobi preconditioning for iterative sparse linear system solvers* (2019).
- Doucet, Ltaief, Gratadour, and Keyes, *Mixed-precision tomographic reconstructor computations on hardware accelerator* (2019).
- Ahmad, Sundar, and Hall, *Data-driven mixed precision sparse matrix vector multiplication for GPUs* (2019).
- Ooi, Iwashita, Fukaya, Ida, and Yokota, *Effect of mixed precision computing on H-matrix vector multiplication in BEM analysis* (2020).
- Diffenderfer, Osei-Kuffuor, and Menon, *QDOT: Quantized dot product kernel for approximate high-performance computing* (2021).
- Abdulah, Cao, Pei, Bosilca, Dongarra, Genton, Keyes, Ltaief, and Sun, *Accelerating geostatistical modeling and prediction with mixed-precision computations* (2022).
- Amestoy, Boiteau, Buttari, Gerest, Jézéquel, L'Excellent, Mary *Mixed precision low-rank approximations and their application to block low-rank LU factorization* (2022)

Sparse matrix–vector product (SpMV) [Oettli–Prager'64], [Rigal–Gaches'67]

$y = Ax$, $A \in \mathbb{R}^{m\times n}$ performed in a uniform precision ϵ

```
for i = 1: m do
    y_i = 0
    for j ∈ nnz_i(A) do
        y_i = y_i + a_ij x_j
    end for
end for
```

Backward error: The computed result is the exact one for a perturbed matrix: $\widehat{y} = (A + \Delta A)x$

- Focus on $\varepsilon_{\mathrm{nw}} = \frac{\|\widehat{y}-y\|}{\|A\|\|x\|}$.
- Similar results for $\varepsilon_{\mathrm{cw}} = \max_i \left[\frac{|\widehat{y}_i - y_i|}{\sum_{j\in J_i} |a_{ij}x_j|}\right]$
- Analysis rely on standard result for scalar product $|\widehat{y}_i - y_i| \le n_i\epsilon \sum_{a_{ij}x_j\in nnz_i(A)} |a_{ij}x_j|$

Adaptive precision SpMV

Goal: compute the SpMV $y = Ax$ with accuracy ϵ using q precisions

$$u_1 \le \epsilon < u_2 < \ldots < u_q$$

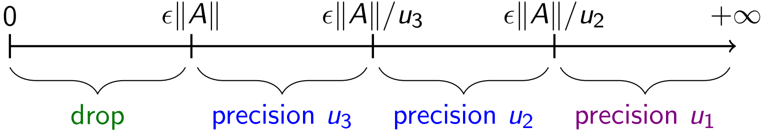
```
for i = 1: m do
    y_i = 0
    for k = 1: p do
        y_i^(k) = 0
        for j ∈ nnz_i(A) do
            if a_ij x_j ∈ B_ik then
                y_i^(k) = y_i^(k) + a_ij x_j at precision u_k
            end if
        end for
        y_i = y_i + y_i^(k)
    end for
end for
```

- Split elements a_{ij} on each row i into q buckets $B_{i1}, \ldots, B_{iq}$, where bucket B_{ik} uses precision u_k
- For each bucket: $|\widehat{y}_i^{(k)} - y_i^{(k)}| \le n_i^{(k)} u_k \sum_{a_{ij}x_j \in B_{ik}} |a_{ij}x_j|$

Adaptive precision SpMV: Normwise (NW) criteria

- How should we build the buckets?

$$\begin{cases} |a_{ij}| \leq \epsilon\|A\| & \Rightarrow \quad \text{drop} \\ |a_{ij}| \in [\epsilon\|A\|/u_{k+1}, \epsilon\|A\|/u_k) & \Rightarrow \quad \text{place in } B_{ik} \\ |a_{ij}| > \epsilon\|A\|/u_2 & \Rightarrow \quad \text{place in } B_{i1} \end{cases}$$



- **Theorem**: the computed $\widehat{y}$ satisfies $\|\widehat{y} - y\| \leq c\epsilon\|A\|\|x\|$ and so, $\varepsilon_{\mathrm{nw}} \leq \epsilon$.

SpMV experimental settings

- 33 matrices coming from SuiteSparse collection and industrial partners

SpMV experimental settings

- 33 matrices coming from SuiteSparse collection and industrial partners
- 3 different **accuracy targets**:
  - $\epsilon = 2^{-24}$ (equivalent to fp32)
  - $\epsilon = 2^{-37}$ (no equivalent)
  - $\epsilon = 2^{-53}$ (equivalent to fp64)

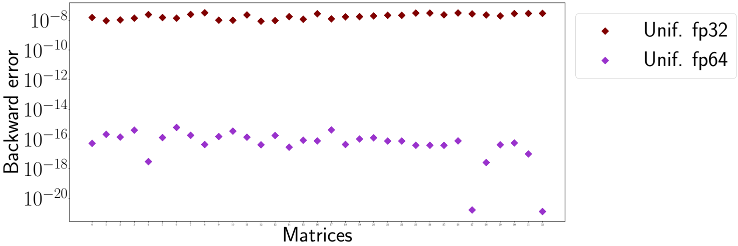
SpMV experimental settings

Various sets of precision formats:

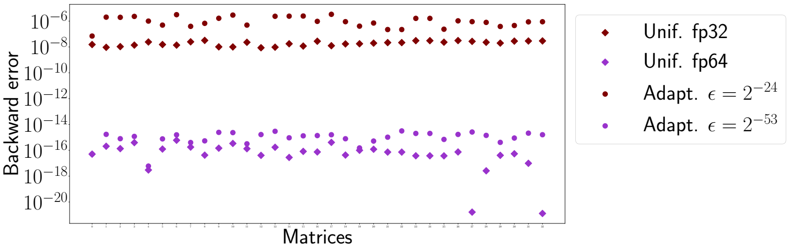
- 2 **precisions**: fp32, fp64
- 3 **precisions**: bfloat16, fp32, fp64
- 7 **precisions**: bfloat16, "fp24", fp32, "fp40", "fp48", "fp56", fp64

	Bits	
	Mantissa	Exponent
bfloat16	8	8
"fp24"	16	8
fp32	24	8
"fp40"	29	11
"fp48"	37	11
"fp56"	45	11
fp64	53	11

Maintaining normwise accuracy

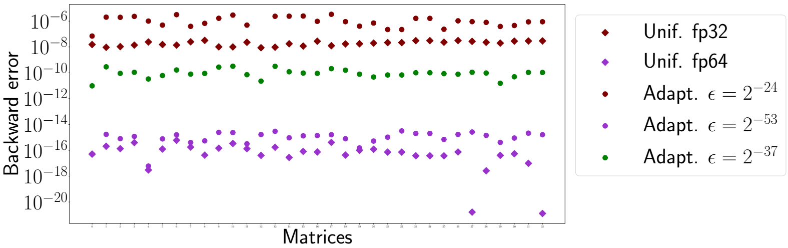


Maintaining normwise accuracy



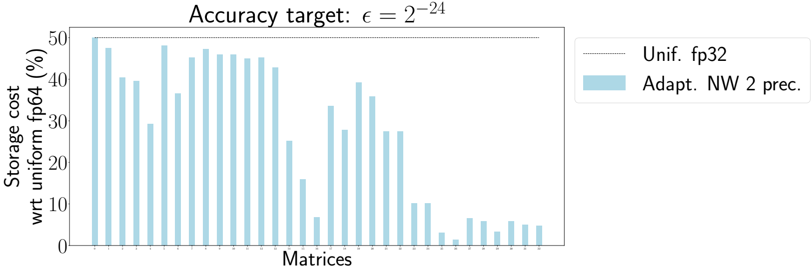
Adaptive methods preserve an accuracy close to the accuracy of uniform methods,

Maintaining normwise accuracy



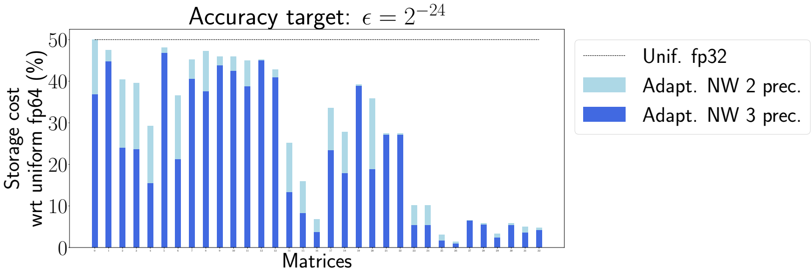
And we are able to target intermediate accuracy.

Theoretical storage gains targeting $\epsilon = 2^{-24}$ accuracy



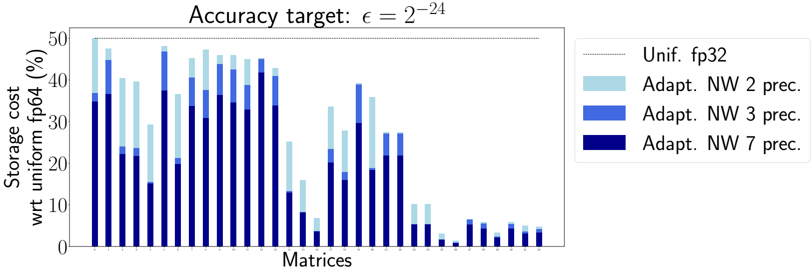
Small bars: most suitable matrices to the adaptive method

Theoretical storage gains targeting $\epsilon = 2^{-24}$ accuracy



Small bars: most suitable matrices to the adaptive method

Theoretical storage gains targeting $\epsilon = 2^{-24}$ accuracy

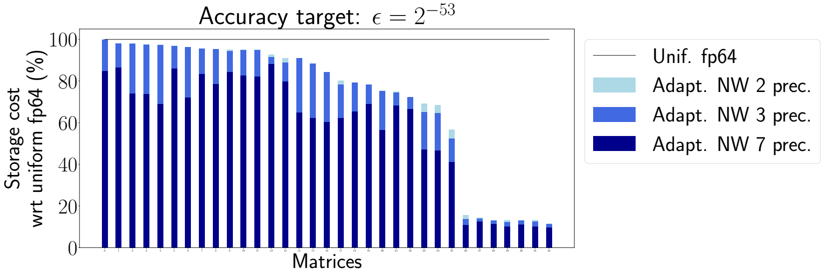


Small bars: most suitable matrices to the adaptive method

The more formats we have, the more the necessary data storage can be reduced up to 36×

Theoretical storage gains targeting $\epsilon = 2^{-53}$ accuracy

for the $\epsilon = 2^{-53}$ target...

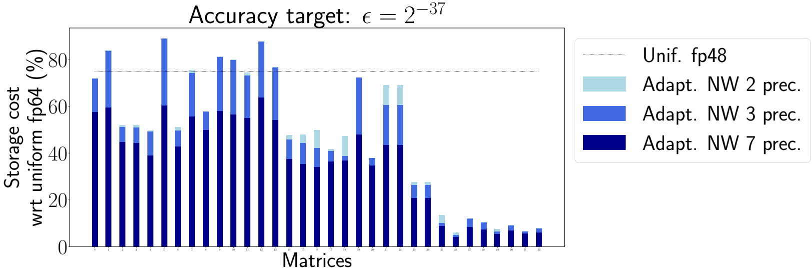


Small bars: most suitable matrices to the adaptive method

SpMV experiments: storage gains

Theoretical storage gains targeting $\epsilon = 2^{-37}$ accuracy

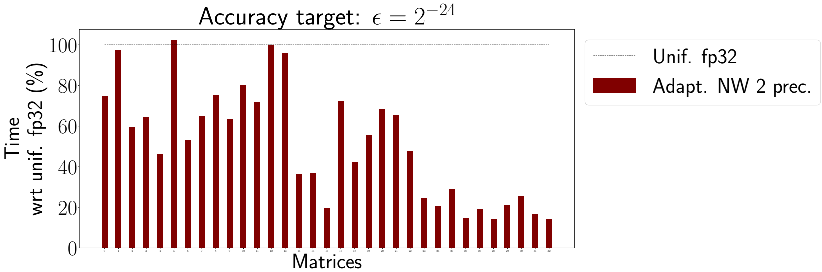
and for intermediate accuracy target.



Small bars: most suitable matrices to the adaptive method

Time experiments with two precisions: fp32 and fp64.

Actual time gains targeting $\epsilon = 2^{-24}$ accuracy (fp32)

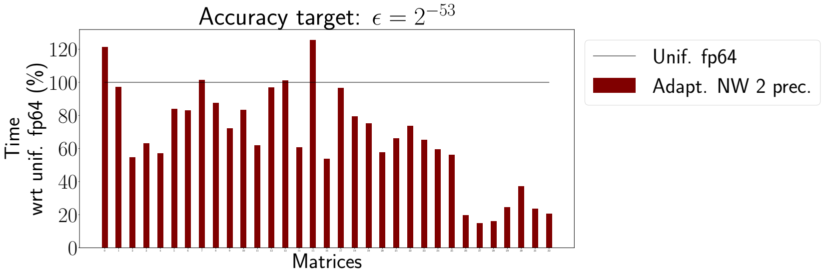


Up to 7× time reduction!

SpMV experiments: time gains

Time experiments with two precisions: fp32 and fp64.

Actual time gains targeting $\epsilon = 2^{-53}$ accuracy (fp64)

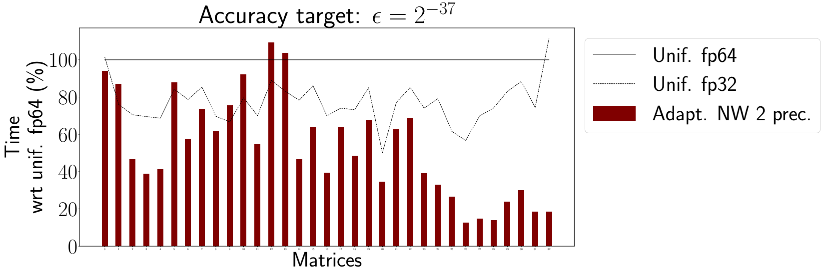


Up to 7× time reduction!

SpMV experiments: time gains

Time experiments with two precisions: fp32 and fp64.

Actual time gains targeting intermediate accuracy: $\epsilon = 2^{-37}$



Up to 7× time reduction!

GMRES-based iterative refinement

GMRES

$r = b - Ax_0$
$\beta = \|r\|_2$
$q_1 = r/\beta$
for $k = 1, 2, \ldots$ **do**
  $y = Aq_k$
  for $j = 1: k$ **do**
    $h_{jk} = q_j^T y$
    $y = y - h_{jk} q_j$
  end for
  $h_{k+1,k} = \|y\|_2$
  $q_{k+1} = y / h_{k+1,k}$
  Solve $\min_{c_k} \|Hc_k - \beta e_1\|_2$.
  $x_k = x_0 + Q_k c_k$
end for

- GMRES performance rely on matrix-vector product
- Interesting to implement adaptive SpMV in GMRES
- How does the adaptive method affect the convergence?

GMRES-based iterative refinement

GMRES

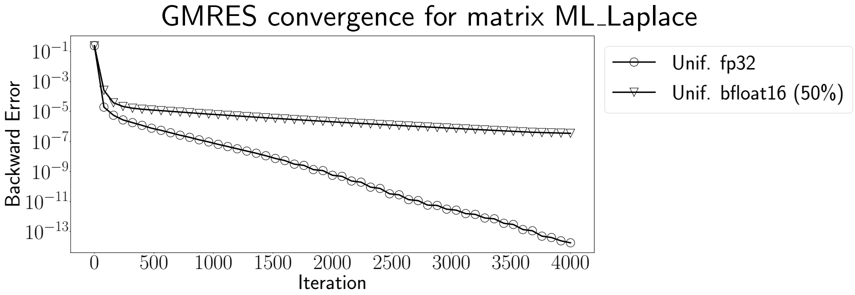
$r = b - Ax_0$
$\beta = \|r\|_2$
$q_1 = r/\beta$
for $k = 1, 2, \ldots$ **do**
  $y = Aq_k \to \epsilon_{\text{in}}$
  for $j = 1 : k$ **do**
    $h_{jk} = q_j^T y$
    $y = y - h_{jk} q_j$
  end for
  $h_{k+1,k} = \|y\|_2$
  $q_{k+1} = y / h_{k+1,k}$
  Solve $\min_{c_k} \|Hc_k - \beta e_1\|_2$.
  $x_k = x_0 + Q_k c_k$
end for

GMRES-IR

for $i = 1, 2, \ldots$ **do**
  $r_i = b - Ax_{i-1} \to \epsilon_{\text{out}}$
  Solve $Ad_i = r_i$ by GMRES
  $x_i = x_{i-1} + d_i$
end for

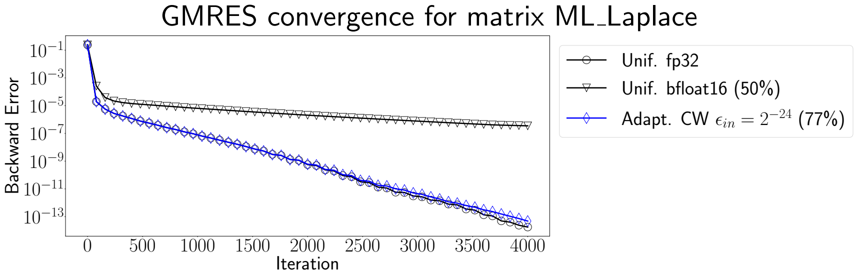
- **Larger speedups for lower accuracy targets**
- GMRES-IR particularly attractive
- Jacobi preconditioner
- $\epsilon_{out} = 2^{-53}$ (fp64)
- restart every 80 iterations

GMRES experiments: role of ϵ_{in}



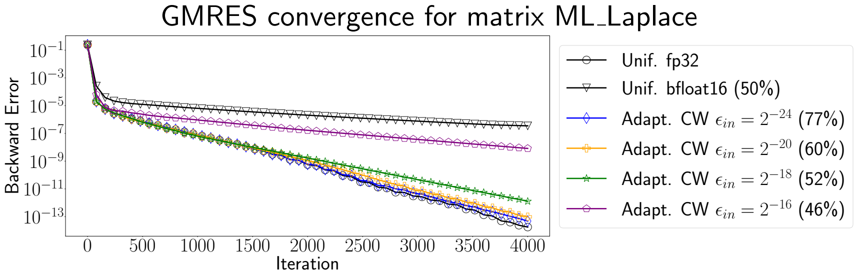
Uniform bfloat16 not enough to converge

GMRES experiments: role of ϵ_{in}



Adaptive SpMV with target $\epsilon_{in} = 2^{-24}$ converges as uniform fp32

GMRES experiments: role of ϵ_{in}



Lower accuracy targets maintain the convergence, one can tune ϵ_{in} for even larger gains!

Future work

To get the most out of adaptive precision SpMV

- experiment on hardware with **native bfloat16** support
- develop **optimized accessors** for custom-precision formats [Anzt et al., 21]
- use more suitable **sparse matrices formats** to reduce indices access cost

Future work

To get the most out of adaptive precision SpMV

- experiment on hardware with **native bfloat16** support
- develop **optimized accessors** for custom-precision formats [Anzt et al., 21]
- use more suitable **sparse matrices formats** to reduce indices access cost

Adaptive precision in the area of Krylov solvers

- Experiment the adaptive SpMV into other Krylov solvers
- Use more **advanced preconditioners**, and develop adaptive precision variants of them (e.g., ILU, SPAI)
- Introduce adaptive precision into the **Krylov basis** following the introduction of mixed-precision in the Krylov basis by [Aliaga & al'22]

Conclusion: take-home messages

- **Adaptive precision SpMV algorithm**
  - Buckets built according to the elements magnitude
  - Error analysis guarantees any accuracy target
  - Matrix-dependent gains up to
    - 97% data reduction
    - 88% time reduction
- **Application to Krylov solvers**
  - Reasonable accuracy targets preserve convergence
  - One can tune this target to find the best trade-off between cost per iteration and convergence speed

Preprint [*Adaptive Precision Sparse Matrix–Vector Product and its Application to Krylov Solvers* Graillat, Jézéquel, Mary, Molina'22]

Thank you! Any questions?