Beyond mixed precision: a short guide to adaptive precision algorithms

Theo Mary
Sorbonne Université, CNRS, LIP6

Dagstuhl Conference on Reduced and Mixed Precision Computing
15-20 February 2026

What is a mixed precision algorithm?

- A mixed precision algorithm mixes several precision formats with the goal of improving performance (wrt high precision) and accuracy (wrt low precision)
- Traditionally limited to a small number of precisions fixed in advance
- Prototypical example: iterative refinement for $Ax = b$

Repeat	
$r = b - Ax$	in high precision
Solve $Ad \approx r$	in low precision
$x = x + d$	in high precision
Until convergence	

- Another common example: high precision accumulators in sums/dot products

Beyond mixed precision: adaptive precision

- In this talk, I advocate for an evolution of mixed precision algorithms towards a more expressive and powerful paradigm: adaptive precision

- Part I: general guidelines for designing adaptive precision algorithms
  - This will seem abstract but I want to extract the essence of adaptive precision without being distracted by algorithm-specific details
  - The ultimate goal is for you to consider applying the same methodology to *your* favorite algorithms

- Part II: examples of how to put these guidelines into action

Part I

General guidelines

Continuum of precisions

		Range $f_{\max}/f_{\min}$	Unit roundoff u
fp128	e15m113	$2^{32766} \approx 10^{9863}$	$2^{-114} \approx 1 \times 10^{-34}$
fp64	e11m52	$2^{2046} \approx 10^{616}$	$2^{-53} \approx 1 \times 10^{-16}$
fp32	e8m23	$2^{254} \approx 10^{76}$	$2^{-24} \approx 6 \times 10^{-8}$

Standard model:

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

Continuum of precisions via hardware

		Range $f_{\max}/f_{\min}$	Unit roundoff u
fp128	e15m113	$2^{32766} \approx 10^{9863}$	$2^{-114} \approx 1 \times 10^{-34}$
fp64	e11m52	$2^{2046} \approx 10^{616}$	$2^{-53} \approx 1 \times 10^{-16}$
fp32	e8m23	$2^{254} \approx 10^{76}$	$2^{-24} \approx 6 \times 10^{-8}$
tfloat32	e8m10	$2^{254} \approx 10^{76}$	$2^{-11} \approx 5 \times 10^{-4}$
fp16	e5m10	$2^{30} \approx 10^{9}$	$2^{-11} \approx 5 \times 10^{-4}$
bfloat16	e8m7	$2^{254} \approx 10^{76}$	$2^{-8} \approx 4 \times 10^{-3}$
fp8	e4m3	$2^{15} \approx 3 \times 10^{4}$	$2^{-4} \approx 6 \times 10^{-2}$
fp8	e5m2	$2^{30} \approx 10^{9}$	$2^{-3} \approx 1 \times 10^{-1}$
fp6	e2m3	$2^{3} \approx 8$	$2^{-4} \approx 6 \times 10^{-2}$
fp6	e3m2	$2^{7} \approx 128$	$2^{-3} \approx 0.125$
fp4	e2m1	$2^{3} \approx 8$	$2^{-2} \approx 0.25$

Standard model:

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

- Lower precisions on specialized hardware (GPUs)

Continuum of precisions via hardware and emulation

		Range $f_{\max}/f_{\min}$	Unit roundoff u
fp128	e15m113	$2^{32766} \approx 10^{9863}$	$2^{-114} \approx 1 \times 10^{-34}$
fp64	e11m52	$2^{2046} \approx 10^{616}$	$2^{-53} \approx 1 \times 10^{-16}$
int8×7	e?m55		$2^{-56} \approx 1 \times 10^{-17}$
int8×6	e?m47		$2^{-48} \approx 4 \times 10^{-15}$
int8×5	e?m39		$2^{-40} \approx 9 \times 10^{-13}$
int8×4	e?m31		$2^{-32} \approx 2 \times 10^{-10}$
fp32	e8m23	$2^{254} \approx 10^{76}$	$2^{-24} \approx 6 \times 10^{-8}$
bfloat16×3	e8m23	$2^{254} \approx 10^{76}$	$2^{-24} \approx 6 \times 10^{-8}$
fp16×2	e5m21	$2^{30} \approx 10^{9}$	$2^{-22} \approx 2 \times 10^{-7}$
bfloat16×2	e8m15	$2^{254} \approx 10^{76}$	$2^{-16} \approx 2 \times 10^{-5}$
tfloat32	e8m10	$2^{254} \approx 10^{76}$	$2^{-11} \approx 5 \times 10^{-4}$
fp16	e5m10	$2^{30} \approx 10^{9}$	$2^{-11} \approx 5 \times 10^{-4}$
bfloat16	e8m7	$2^{254} \approx 10^{76}$	$2^{-8} \approx 4 \times 10^{-3}$
fp8	e4m3	$2^{15} \approx 3 \times 10^{4}$	$2^{-4} \approx 6 \times 10^{-2}$
fp8	e5m2	$2^{30} \approx 10^{9}$	$2^{-3} \approx 1 \times 10^{-1}$
fp6	e2m3	$2^{3} \approx 8$	$2^{-4} \approx 6 \times 10^{-2}$
fp6	e3m2	$2^{7} \approx 128$	$2^{-3} \approx 0.125$
fp4	e2m1	$2^{3} \approx 8$	$2^{-2} \approx 0.25$

Standard model:

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

- Lower precisions on specialized hardware (GPUs)
- Emulated precisions via multiword arithmetic/Ozaki scheme, number of words/slices controls precision
- Designed for matrix–matrix multiplication ⇒ compute-bound operations

Continuum of precisions

		Range $f_{\max}/f_{\min}$	Unit roundoff u
fp128	e15m113	$2^{32766} \approx 10^{9863}$	$2^{-114} \approx 1 \times 10^{-34}$
fp64	e11m52	$2^{2046} \approx 10^{616}$	$2^{-53} \approx 1 \times 10^{-16}$
fp32	e8m23	$2^{254} \approx 10^{76}$	$2^{-24} \approx 6 \times 10^{-8}$

Standard model:

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

Continuum of precisions via software and memory accessors

		Range $f_{\max}/f_{\min}$	Unit roundoff u
fp128	e15m113	$2^{32766} \approx 10^{9863}$	$2^{-114} \approx 1 \times 10^{-34}$
fp64	e11m52	$2^{2046} \approx 10^{616}$	$2^{-53} \approx 1 \times 10^{-16}$
rp56	e11m44	$2^{2046} \approx 10^{616}$	$2^{-45} \approx 3 \times 10^{-14}$
rp48	e11m36	$2^{2046} \approx 10^{616}$	$2^{-37} \approx 7 \times 10^{-12}$
rp40	e11m28	$2^{2046} \approx 10^{616}$	$2^{-29} \approx 2 \times 10^{-9}$
fp32	e8m23	$2^{254} \approx 10^{76}$	$2^{-24} \approx 6 \times 10^{-8}$
rp24	e8m15	$2^{254} \approx 10^{76}$	$2^{-16} \approx 2 \times 10^{-5}$
rp16	e8m7	$2^{254} \approx 10^{76}$	$2^{-8} \approx 4 \times 10^{-3}$

- Custom precisions: storage-only, efficient on-the-fly decompression (memory accessors)
- Can also reduce exponent, or indeed use any other type of compressor (e.g., ZFP/SZ)

Standard model:

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

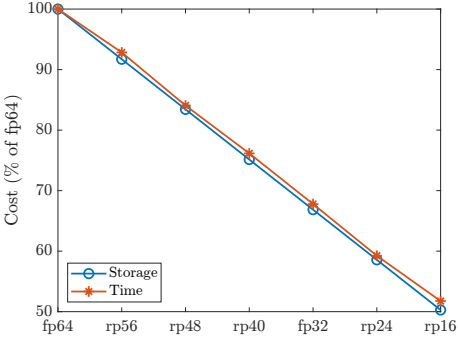
Continuum of precisions via software and memory accessors

		Range $f_{\max}/f_{\min}$	Unit roundoff u
fp128	e15m113	$2^{32766} \approx 10^{9863}$	$2^{-114} \approx 1 \times 10^{-34}$
fp64	e11m52	$2^{2046} \approx 10^{616}$	$2^{-53} \approx 1 \times 10^{-16}$
rp56	e11m44	$2^{2046} \approx 10^{616}$	$2^{-45} \approx 3 \times 10^{-14}$
rp48	e11m36	$2^{2046} \approx 10^{616}$	$2^{-37} \approx 7 \times 10^{-12}$
rp40	e11m28	$2^{2046} \approx 10^{616}$	$2^{-29} \approx 2 \times 10^{-9}$
fp32	e8m23	$2^{254} \approx 10^{76}$	$2^{-24} \approx 6 \times 10^{-8}$
rp24	e8m15	$2^{254} \approx 10^{76}$	$2^{-16} \approx 2 \times 10^{-5}$
rp16	e8m7	$2^{254} \approx 10^{76}$	$2^{-8} \approx 4 \times 10^{-3}$

- Custom precisions: storage-only, efficient on-the-fly decompression (memory accessors)
- Can also reduce exponent, or indeed use any other type of compressor (e.g., ZFP/SZ)
- Performance $\propto$ storage for memory-bound operations

Standard model:

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



SpMV performance with memory accessor

Long_Coup_dt0 matrix

Continuum of precisions

- To sum up, we have access to a continuum of precisions
  - via memory accessor-based custom precisions, for memory-bound operations
  - via hardware and/or emulation-based, for compute-bound operations

$\Rightarrow$ mixed precision algorithms limited to a small number of precisions leave potential performance on the table

- We should no longer think of precision as a static, discrete parameter:
  don't ask

  > Should this operation be in low or high precision?

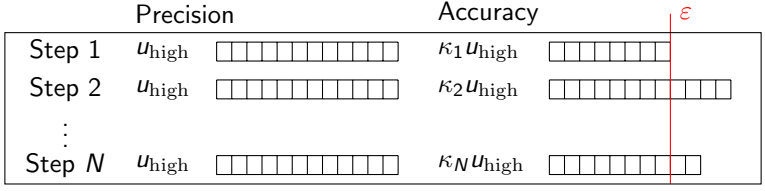
  but rather

  > What is the required precision for this operation?

Continuum of accuracies

- Precision $\neq$ accuracy
  - Precision is the error of elementary operations $\{+, -, \times, \div\}$
  - Accuracy is the error of a sequence of operations
- Consider an abstract computation partitioned into N steps, and let κ_i be the precision/accuracy amplification factor of step i
- Even with only two precisions u_{high} and u_{low}, any target accuracy $\varepsilon > 0$ can be achieved by switching to precision u_{low} any step i such that $\kappa_i u_{\text{low}} \leq \varepsilon u_{\text{high}}$

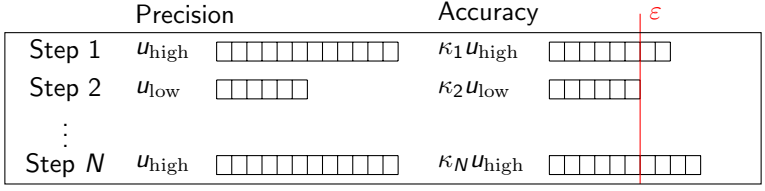
$\Rightarrow$ By mixing precisions, we can create a continuum of accuracies and finely tune the performance–accuracy tradeoff. Moreover, this allows for balancing rounding errors with errors from other sources (controlled by ε)



Continuum of accuracies

- Precision $\neq$ accuracy
  - Precision is the error of elementary operations $\{+, -, \times, \div\}$
  - Accuracy is the error of a sequence of operations
- Consider an abstract computation partitioned into N steps, and let κ_i be the precision/accuracy amplification factor of step i
- Even with only two precisions u_{high} and u_{low}, any target accuracy $\varepsilon > 0$ can be achieved by switching to precision u_{low} any step i such that $\kappa_i u_{\text{low}} \leq \varepsilon u_{\text{high}}$

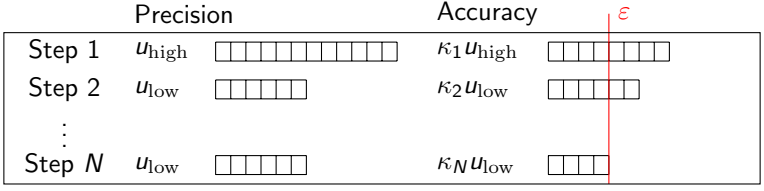
$\Rightarrow$ By mixing precisions, we can create a continuum of accuracies and finely tune the performance–accuracy tradeoff. Moreover, this allows for balancing rounding errors with errors from other sources (controlled by ε)



Continuum of accuracies

- Precision $\neq$ accuracy
  - Precision is the error of elementary operations $\{+, -, \times, \div\}$
  - Accuracy is the error of a sequence of operations
- Consider an abstract computation partitioned into N steps, and let κ_i be the precision/accuracy amplification factor of step i
- Even with only two precisions u_{high} and u_{low}, any target accuracy $\varepsilon > 0$ can be achieved by switching to precision u_{low} any step i such that $\kappa_i u_{\text{low}} \leq \varepsilon u_{\text{high}}$

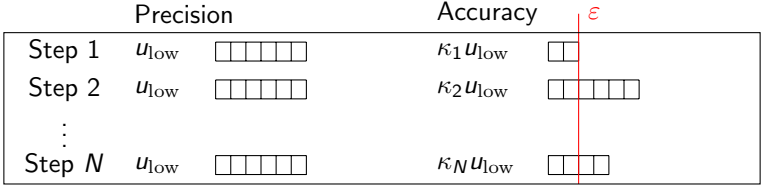
$\Rightarrow$ By mixing precisions, we can create a continuum of accuracies and finely tune the performance–accuracy tradeoff. Moreover, this allows for balancing rounding errors with errors from other sources (controlled by ε)



Continuum of accuracies

- Precision $\neq$ accuracy
  - Precision is the error of elementary operations $\{+, -, \times, \div\}$
  - Accuracy is the error of a sequence of operations
- Consider an abstract computation partitioned into N steps, and let κ_i be the precision/accuracy amplification factor of step i
- Even with only two precisions u_{high} and u_{low}, any target accuracy $\varepsilon > 0$ can be achieved by switching to precision u_{low} any step i such that $\kappa_i u_{\text{low}} \leq \varepsilon u_{\text{high}}$

$\Rightarrow$ By mixing precisions, we can create a continuum of accuracies and finely tune the performance–accuracy tradeoff. Moreover, this allows for balancing rounding errors with errors from other sources (controlled by ε)

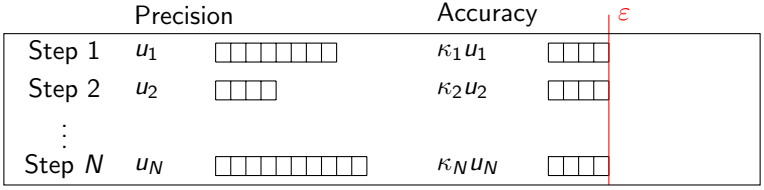


Continuum of accuracies

- Precision $\neq$ accuracy
  - Precision is the error of elementary operations $\{+, -, \times, \div\}$
  - Accuracy is the error of a sequence of operations
- Consider an abstract computation partitioned into N steps, and let κ_i be the precision/accuracy amplification factor of step i
- Even with only two precisions u_{high} and u_{low}, any target accuracy $\varepsilon > 0$ can be achieved by switching to precision u_{low} any step i such that $\kappa_i u_{\text{low}} \leq \varepsilon u_{\text{high}}$

$\Rightarrow$ By mixing precisions, we can create a continuum of accuracies and finely tune the performance–accuracy tradeoff. Moreover, this allows for balancing rounding errors with errors from other sources (controlled by ε)

- Combine with a continuum of precisions to equilibrate the accuracies: $\forall i \ \kappa_i u_i \approx \varepsilon$



The role of error analysis

- Given precisions $u_1 < \ldots < u_p$ and a target accuracy $\varepsilon > 0$, how do we decide:
  - how to define a partitioning of the computation into steps $1, \ldots, N$?
  - which precision $\{u_1, \ldots, u_p\}$ to assign to each step i ?

- Impossible to decide based on intuition or heuristics only!

- Use error analysis to reveal the numerical structure of the computation:
  - Find expressions for the κ_i factors
  - Group together operations with similar κ_i
  - Assign precision $\max\{k = 1\colon p,\ u_k \kappa_i \leq \varepsilon\}$ to each step i

$\Rightarrow$ Error analysis has a new role to play in the design of mixed precision algorithms
  - Traditionally: develop algorithm $\Rightarrow$ analyze it to prove stability/identify instabilities
  - Today: analyze computation $\Rightarrow$ develop algorithm that exploits mixed precision opportunities (in a provably accurate way)

Dynamic precision choice

- Note that κ_i usually depends on the data and is thus only known at runtime
  $\Rightarrow$ dynamic precision choice required

- When κ_i is cheap to compute, we not only get provable accuracy but also provable performance
  - Since it is provably accurate, the algorithm can't "fail" or kill performance
  - In the worst case, there are no mixed precision opportunities and the adaptive algorithm falls back to the high precision one

- What if κ_i is expensive to compute?
  - Replace with proxy estimate $\widetilde{\kappa}_i$
  - Use optimistic look-ahead: compute in low precision, check, recompute in high precision if needed
  - In the worst case, none of the above is possible/practical, but having done the error analysis still has taught us valuable insights (even if we can't exploit them)

Part II

Putting the guide into action

- Goal: compute $y = Ax$ with accuracy ε and precisions $u_1 \leq \varepsilon < u_2 < \ldots < u_p$.
- **Analysis:** define partitioning $A = \sum_{k=1}^{p} A^{(k)}$ with $A^{(k)}$ stored in precision u_k. The computed $\widehat{y}$ satisfies

$$\|\widehat{y} - y\| \leq c \sum_{k=1}^{p} u_k \|A^{(k)}\| \|x\|$$

- Goal: compute $y = Ax$ with accuracy ε and precisions $u_1 \leq \varepsilon < u_2 < \ldots < u_p$.
- **Analysis:** define partitioning $A = \sum_{k=1}^{p} A^{(k)}$ with $A^{(k)}$ stored in precision u_k. The computed $\widehat{y}$ satisfies

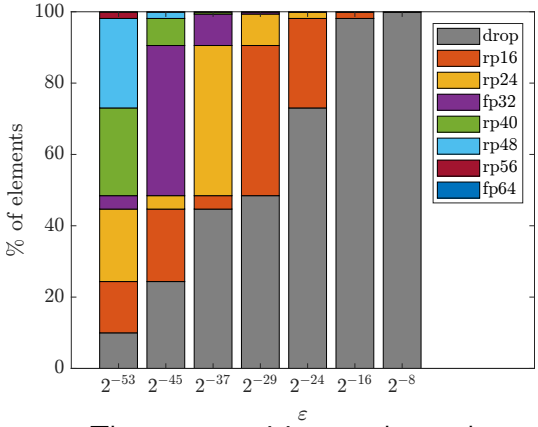
$$\|\widehat{y} - y\| \leq c \sum_{k=1}^{p} u_k \|A^{(k)}\| \|x\|$$

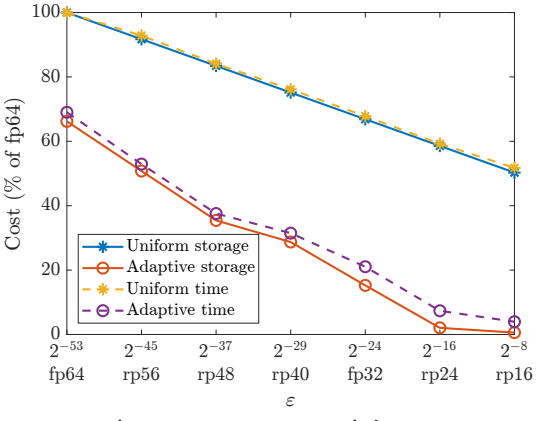
$\Rightarrow$ **Algorithm**: we build $A^{(k)}$ such that $\|A^{(k)}\| \approx \varepsilon \|A\| / u_k$:

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

and so we obtain $\|\widehat{y} - y\| \leq c' \varepsilon \|A\| \|x\|$

Long_Coup_dt0 matrix





- The more precisions we have, the more we can reduce storage $\Rightarrow$ exploit continuum of custom precisions with memory accessor
- Performance is entirely matrix-dependent, but consistent (no worse than uniform fp64)

AP GMRES

- Building the adaptive precision matrix is cheap: only need to scan A twice. Can then be reused for multiple SpMVs

$\Rightarrow$ Iterative solvers are a natural application. But how do we choose ε?

- First answer: set ε to the iterative stopping tolerance.

- Building the adaptive precision matrix is cheap: only need to scan A twice. Can then be reused for multiple SpMVs

$\Rightarrow$ Iterative solvers are a natural application. But how do we choose ε?

- First answer: set ε to the iterative stopping tolerance. Better answer:

Relaxed GMRES (Giraud, Gratton, Langou, 2009)

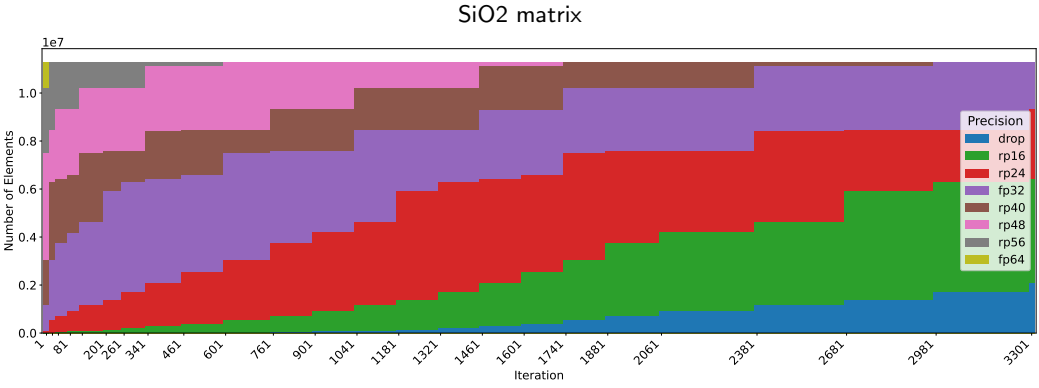
If at each iteration k the SpMV is performed with $\widetilde{A}_k = A + E_k$ such that

$$\frac{\|E_k\|}{\|A\|} \leq \frac{1}{2n\kappa(A)} \min\left(1, \frac{\|b\|}{\|\widetilde{r}_{k-1}\|}\frac{\varepsilon}{2}\right),$$

where $\widetilde{r}_{k-1}$ is the Arnoldi residual, then there exists an iteration ℓ such that the true residual satisfies $\|r_\ell\| \leq \varepsilon\|b\|$

- SpMV accuracy can be reduced to be inversely proportional to the residual norm $\Rightarrow$ lower and lower precision as iterations progress

AP GMRES (ongoing work w/ Agullo, Giraud, Jolivet, Tabouret)



$\Rightarrow$ 1.5$\times$ speedup wrt fp64 GMRES

AP domain decomposition preconditioning (ongoing work w/ Caruso, Jolivet, Nataf, Tournier)

- Given an SPD system, consider the additive Schwarz preconditioner

$$M^{-1} = \sum_{i=0}^{N} R_i^T A_i^{-1} R_i,$$

where $A_1, \ldots, A_N$ are N (overlapping) subdomains and A_0 is the coarse space

- **Analysis:** the adaptive precision preconditioner

$$\widehat{M}^{-1} = \sum_{i=0}^{N} R_i^T (A_i + E_i)^{-1} R_i, \quad \|E_i\| \leq u_i \|A_i\|$$

satisfies

$$\kappa(\widehat{M}^{-1} A) \leq \frac{1+\varepsilon}{1-\varepsilon} \kappa(M^{-1} A), \quad \varepsilon = \max_i u_i \kappa(A_i)$$

AP domain decomposition preconditioning (ongoing work w/ Caruso, Jolivet, Nataf, Tournier)

- Given an SPD system, consider the additive Schwarz preconditioner

$$M^{-1} = \sum_{i=0}^{N} R_i^T A_i^{-1} R_i,$$

where $A_1, \ldots, A_N$ are N (overlapping) subdomains and A_0 is the coarse space

- **Analysis:** the adaptive precision preconditioner

$$\widehat{M}^{-1} = \sum_{i=0}^{N} R_i^T (A_i + E_i)^{-1} R_i, \quad \|E_i\| \leq u_i \|A_i\|$$

satisfies

$$\kappa(\widehat{M}^{-1} A) \leq \frac{1+\varepsilon}{1-\varepsilon} \kappa(M^{-1} A), \quad \varepsilon = \max_i u_i \kappa(A_i)$$

$\Rightarrow$ **Algorithm:** set $u_i \propto 1/\kappa(A_i)$

- Factorize/invert A_i in high precision, estimate $\kappa(A_i)$, apply A_i^{-1} in precision u_i
- or factorize/invert optimistically in low precision, and recompute if needed

AP domain decomposition preconditioning (ongoing work w/ Caruso, Jolivet, Nataf, Tournier)

- Given an SPD system, consider the additive Schwarz preconditioner

$$M^{-1} = \sum_{i=0}^{N} R_i^T A_i^{-1} R_i,$$

where $A_1, \ldots, A_N$ are N (overlapping) subdomains and A_0 is the coarse space

- **Analysis:** the adaptive precision preconditioner

$$\widehat{M}^{-1} = \sum_{i=0}^{N} R_i^T (A_i + E_i)^{-1} R_i, \quad \|E_i\| \leq u_i \|A_i\|$$

satisfies

$$\kappa(\widehat{M}^{-1}A) \leq \frac{1+\varepsilon}{1-\varepsilon} \kappa(M^{-1}A), \quad \varepsilon = \max_i u_i \kappa(A_i)$$

⇒ **Algorithm:** set $u_i \propto 1/\kappa(A_i)$

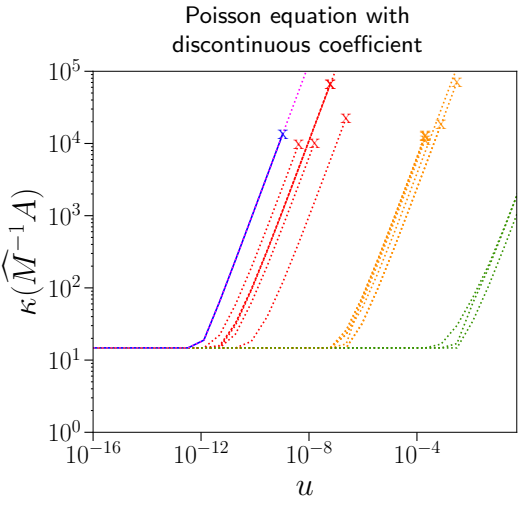
  - Factorize/invert A_i in high precision, estimate $\kappa(A_i)$, apply A_i^{-1} in precision u_i
  - or factorize/invert optimistically in low precision, and recompute if needed

- Our analysis also covers the GenEO method to build the coarse space. It involves solving local GEVPs whose precision should also be set $\propto 1/\kappa(A_i)$
- See also AP block Jacobi preconditioning (Anzt, Dongarra, Flegar, Higham, Quintana-Ortí, 2019)

AP domain decomposition preconditioning (ongoing work w/ Caruso, Jolivet, Nataf, Tournier)

Uniform precision: $u_i = u$

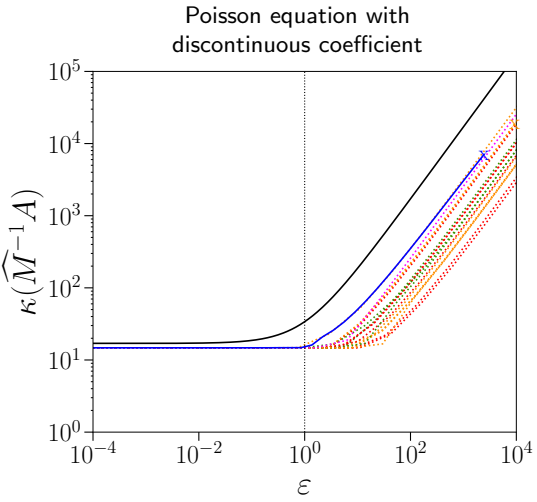
- All subdo.
- One subdo. $\kappa(A_i) \in [10^3, 10^4]$
- One subdo. $\kappa(A_i) \in [10^7, 10^9]$
- One subdo. $\kappa(A_i) \in [10^{11}, 10^{13}]$
- Coarse matrix $\kappa(A_0) = 5.0 \times 10^{12}$



AP domain decomposition preconditioning (ongoing work w/ Caruso, Jolivet, Nataf, Tournier)

Adaptive precision: $u_i = \varepsilon/\kappa(A_i)$

- All subdo.
- One subdo. $\kappa(A_i) \in [10^3, 10^4]$
- One subdo. $\kappa(A_i) \in [10^7, 10^9]$
- One subdo. $\kappa(A_i) \in [10^{11}, 10^{13}]$
- Coarse matrix $\kappa(A_0) = 5.0\times10^{12}$
- Theoretical bound



AP incomplete factorization (ongoing work w/ Lister and Scott)

- Let $A = LU$; ℓ_{ij} is computed as

$$\ell_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj} \right) / u_{jj}$$

- **Analysis:** let $\widehat{\ell}_{ij} = \ell_{ij}(1 + \delta_{ij})$; we have

$$\ell_{ij}(1 + \delta_{ij}) = \left(a_{ij} - \sum_{k=1}^{j-1} \widehat{\ell}_{ik} \widehat{u}_{kj} \right) / \widehat{u}_{jj}$$

$$\Leftrightarrow a_{ij} = \sum_{k=1}^{j} \widehat{\ell}_{ik} \widehat{u}_{kj} + \ell_{ij} \widehat{u}_{jj} \delta_{ij}$$

AP incomplete factorization (ongoing work w/ Lister and Scott)

- Let $A = LU$; ℓ_{ij} is computed as

$$\ell_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj} \right) / u_{jj}$$

- **Analysis:** let $\widehat{\ell}_{ij} = \ell_{ij}(1 + \delta_{ij})$; we have

$$\ell_{ij}(1 + \delta_{ij}) = \left(a_{ij} - \sum_{k=1}^{j-1} \widehat{\ell}_{ik} \widehat{u}_{kj} \right) / \widehat{u}_{jj}$$

$$\Leftrightarrow a_{ij} = \sum_{k=1}^{j} \widehat{\ell}_{ik} \widehat{u}_{kj} + \ell_{ij} \widehat{u}_{jj} \delta_{ij}$$

$\Rightarrow$ **Algorithm:**

$$\begin{cases} \text{if } |\ell_{ij} \widehat{u}_{jj}| \leq \varepsilon \|A\| & \text{drop } \ell_{ij} \ (\widehat{\ell}_{ij} = 0, \ \delta_{ij} = -1) \\ \text{else} & \text{store } \ell_{ij} \text{ in precision } \leq \frac{\varepsilon \|A\|}{|\ell_{ij} \widehat{u}_{jj}|} \end{cases}$$

Then $\|A - \widehat{L}\widehat{U}\| \leq \varepsilon \|A\|$

AP incomplete factorization (ongoing work w/ Lister and Scott)

- Let $A = LU$; ℓ_{ij} is computed as

$$\ell_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj}\right) / u_{jj}$$

- **Analysis:** let $\widehat{\ell}_{ij} = \ell_{ij}(1+\delta_{ij})$; we have

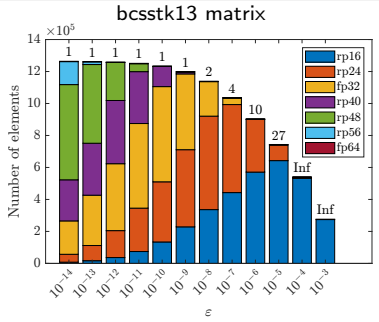
$$\ell_{ij}(1+\delta_{ij}) = \left(a_{ij} - \sum_{k=1}^{j-1} \widehat{\ell}_{ik}\widehat{u}_{kj}\right) / \widehat{u}_{jj}$$

$$\Leftrightarrow a_{ij} = \sum_{k=1}^{j} \widehat{\ell}_{ik}\widehat{u}_{kj} + \ell_{ij}\widehat{u}_{jj}\delta_{ij}$$

$\Rightarrow$ **Algorithm:**

$$\begin{cases} \text{if } |\ell_{ij}\widehat{u}_{jj}| \le \varepsilon\|A\| & \text{drop } \ell_{ij} \ (\widehat{\ell}_{ij} = 0,\ \delta_{ij} = -1) \\ \text{else} & \text{store } \ell_{ij} \text{ in precision } \le \frac{\varepsilon\|A\|}{|\ell_{ij}\widehat{u}_{jj}|} \end{cases}$$

Then $\|A - \widehat{L}\widehat{U}\| \le \varepsilon\|A\|$



AP incomplete factorization (ongoing work w/ Lister and Scott)

- Let $A = LU$; ℓ_{ij} is computed as

$$\ell_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj}\right) / u_{jj}$$

- **Analysis:** let $\widehat{\ell}_{ij} = \ell_{ij}(1+\delta_{ij})$; we have

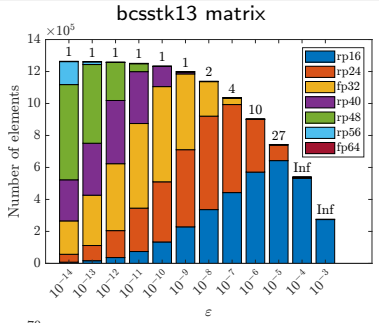
$$\ell_{ij}(1+\delta_{ij}) = \left(a_{ij} - \sum_{k=1}^{j-1} \widehat{\ell}_{ik}\widehat{u}_{kj}\right) / \widehat{u}_{jj}$$

$$\Leftrightarrow a_{ij} = \sum_{k=1}^{j} \widehat{\ell}_{ik}\widehat{u}_{kj} + \ell_{ij}\widehat{u}_{jj}\delta_{ij}$$

$\Rightarrow$ **Algorithm:**

$$\begin{cases} \text{if } |\ell_{ij}\widehat{u}_{jj}| \le \varepsilon\|A\| & \text{drop } \ell_{ij} \ (\widehat{\ell}_{ij} = 0,\ \delta_{ij} = -1) \\ \text{else} & \text{store } \ell_{ij} \text{ in precision } \le \frac{\varepsilon\|A\|}{|\ell_{ij}\widehat{u}_{jj}|} \end{cases}$$

Then $\|A - \widehat{L}\widehat{U}\| \le \varepsilon\|A\|$



AP incomplete factorization (ongoing work w/ Lister and Scott)

- Let $A = LU$; ℓ_{ij} is computed as

$$\ell_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj}\right) / u_{jj}$$

- **Analysis:** let $\widehat{\ell}_{ij} = \ell_{ij}(1+\delta_{ij})$; we have

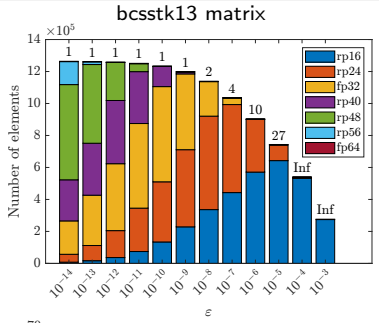
$$\ell_{ij}(1+\delta_{ij}) = \left(a_{ij} - \sum_{k=1}^{j-1} \widehat{\ell}_{ik}\widehat{u}_{kj}\right) / \widehat{u}_{jj}$$

$$\Leftrightarrow a_{ij} = \sum_{k=1}^{j} \widehat{\ell}_{ik}\widehat{u}_{kj} + \ell_{ij}\widehat{u}_{jj}\delta_{ij}$$

$\Rightarrow$ **Algorithm:**

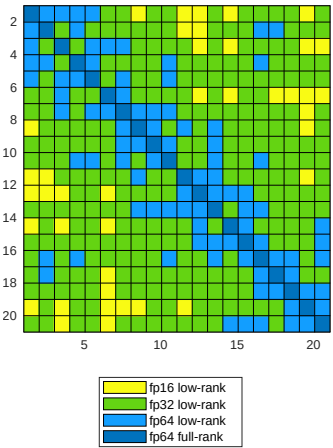
$$\begin{cases} \text{if } |\ell_{ij}\widehat{u}_{jj}| \le \varepsilon\|A\| & \text{drop } \ell_{ij} \ (\widehat{\ell}_{ij} = 0,\ \delta_{ij} = -1) \\ \text{else} & \text{store } \ell_{ij} \text{ in precision } \le \frac{\varepsilon\|A\|}{|\ell_{ij}\widehat{u}_{jj}|} \end{cases}$$

Then $\|A - \widehat{L}\widehat{U}\| \le \varepsilon\|A\|$



AP tile factorization (ongoing work w/ Ltaief)

Root frontal matrix
in 3D Poisson problem

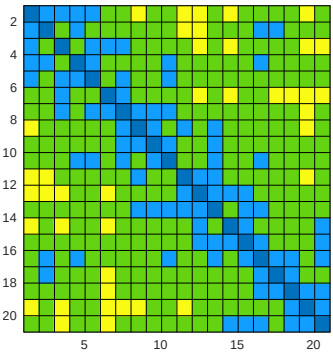


- Consider a $pb \times pb$ matrix A partitioned into $b \times b$ tiles A_{ij}
- **Analysis:** let $\widehat{A}$ be obtained by rounding each A_{ij} to precision u_{ij}. Then we have

$$\begin{aligned} \|A - \widehat{A}\|_F^2 &= \sum_{i,j} \|A_{ij} - \widehat{A}_{ij}\|_F^2 \\ &\leq \sum_{i,j} u_{ij}^2 \|A_{ij}\|_F^2 \end{aligned}$$

AP tile factorization (ongoing work w/ Ltaief)

Root frontal matrix
in 3D Poisson problem

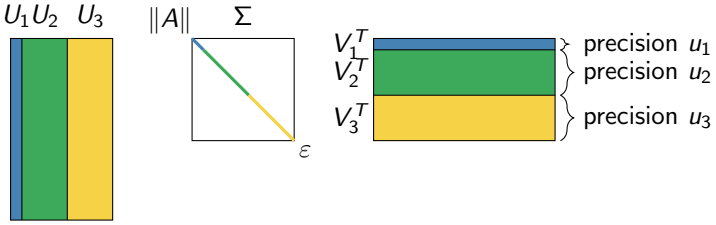


- Consider a $pb \times pb$ matrix A partitioned into $b \times b$ tiles A_{ij}
- **Analysis:** let $\widehat{A}$ be obtained by rounding each A_{ij} to precision u_{ij}. Then we have

$$\|A - \widehat{A}\|_F^2 = \sum_{i,j} \|A_{ij} - \widehat{A}_{ij}\|_F^2 \leq \sum_{i,j} u_{ij}^2 \|A_{ij}\|_F^2$$

$\Rightarrow$ **Algorithm:** store A_{ij} in precision $u_{ij} \propto \frac{\|A\|}{\|A_{ij}\|}\varepsilon$

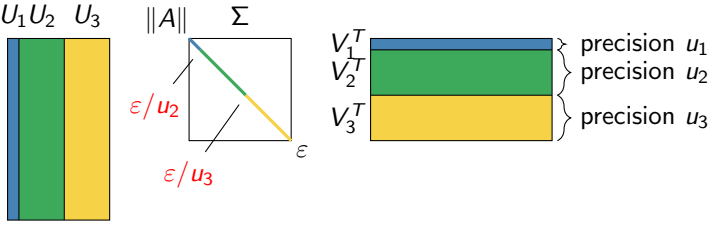
- Moreover, in the factorization of A, the update $A_{ij} \leftarrow A_{ij} - A_{ik}A_{kj}$ can be performed in precision $u_{ijk} \propto \frac{\|A\|}{\|A_{ik}\|\|A_{kj}\|}\varepsilon$



- **Analysis**: Given $A = U\Sigma V^T$, define the adaptive precision partitioning $\widehat{A} = \sum_{k=1}^{p} \widehat{U}_k \Sigma_k \widehat{V}_k^T$ where $\widehat{U}_k$ and $\widehat{V}_k$ are stored in precision u_k. Then

$$\|A - \widehat{A}\| \leq \sum_{k=1}^{p} c u_k \|\Sigma_k\|$$

AP low-rank approximation (Amestoy, Boiteau, Buttari, Gerest, Jézéquel, L'Excellent, M., 2022)



- **Analysis**: Given $A = U\Sigma V^T$, define the adaptive precision partitioning $\widehat{A} = \sum_{k=1}^{p} \widehat{U}_k \Sigma_k \widehat{V}_k^T$ where $\widehat{U}_k$ and $\widehat{V}_k$ are stored in precision u_k. Then

$$\|A - \widehat{A}\| \leq \sum_{k=1}^{p} c u_k \|\Sigma_k\|$$

$\Rightarrow$ **Algorithm**: sort Σ and partition it such that $\|\Sigma_k\| \leq \varepsilon \|A\|/u_k$, yielding $\|\widehat{A} - A\| \leq c'\varepsilon\|A\|$

- Beyond SVD, can be applied to any LRA with decaying rank-1 components (e.g., RRQR)

AP truncated Householder QR (Buttari, M., Pacteau, 2025)

- **Analysis:** consider a truncated QR factorization computed such that $k = \sum_{i=1}^{p} k_i$ Householder transformations are applied to a matrix A, where k_i is the number of transformations computed in precision u_i. The computed $\widehat{R}_i$ and $\widehat{Q}_i$ satisfy

$$\|A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i\| \leq \|A_{p+1}\| + \sum_{i=1}^{p} c u_i \|A_i\|.$$

where A_i is the trailing submatrix after $\sum_{j=1}^{i-1} k_j$ transformations.

AP truncated Householder QR (Buttari, M., Pacteau, 2025)

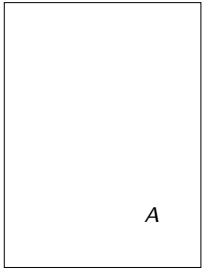
- **Analysis:** consider a truncated QR factorization computed such that $k = \sum_{i=1}^{p} k_i$ Householder transformations are applied to a matrix A, where k_i is the number of transformations computed in precision u_i. The computed $\widehat{R}_i$ and $\widehat{Q}_i$ satisfy

$$\|A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i\| \leq \|A_{p+1}\| + \sum_{i=1}^{p} c u_i \|A_i\|.$$

where A_i is the trailing submatrix after $\sum_{j=1}^{i-1} k_j$ transformations.

⇒ **Algorithm:** balance $\|A_{p+1}\| \approx u_i \|A_i\| \approx \varepsilon \|A\|$

1. start the factorization with $u_1 \leq \varepsilon$



AP truncated Householder QR (Buttari, M., Pacteau, 2025)

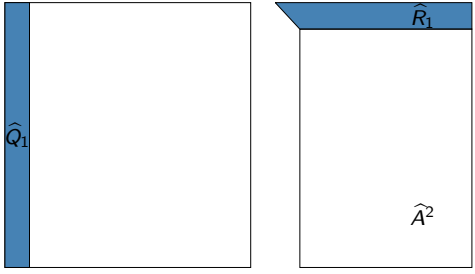
- **Analysis:** consider a truncated QR factorization computed such that $k = \sum_{i=1}^{p} k_i$ Householder transformations are applied to a matrix A, where k_i is the number of transformations computed in precision u_i. The computed $\widehat{R}_i$ and $\widehat{Q}_i$ satisfy

$$\|A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i\| \leq \|A_{p+1}\| + \sum_{i=1}^{p} c u_i \|A_i\|.$$

where A_i is the trailing submatrix after $\sum_{j=1}^{i-1} k_j$ transformations.

⇒ **Algorithm:** balance $\|A_{p+1}\| \approx u_i \|A_i\| \approx \varepsilon \|A\|$

1. start the factorization with $u_1 \leq \varepsilon$
2. if after k_1 transformations $\|\widehat{A}^2\| \leq \varepsilon / u_2 \|A\|$, switch to precision u_2



AP truncated Householder QR (Buttari, M., Pacteau, 2025)

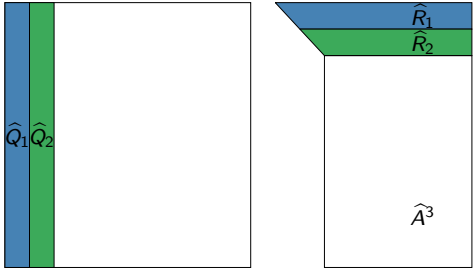
- **Analysis:** consider a truncated QR factorization computed such that $k = \sum_{i=1}^{p} k_i$ Householder transformations are applied to a matrix A, where k_i is the number of transformations computed in precision u_i. The computed $\widehat{R}_i$ and $\widehat{Q}_i$ satisfy

$$\|A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i\| \leq \|A_{p+1}\| + \sum_{i=1}^{p} c u_i \|A_i\|.$$

where A_i is the trailing submatrix after $\sum_{j=1}^{i-1} k_j$ transformations.

⇒ **Algorithm:** balance $\|A_{p+1}\| \approx u_i \|A_i\| \approx \varepsilon \|A\|$

1. start the factorization with $u_1 \leq \varepsilon$
2. if after k_1 transformations $\|A^2\| \leq \varepsilon / u_2 \|A\|$, switch to precision u_2
3. same for precisions $u_2, \ldots, u_p$



AP truncated Householder QR (Buttari, M., Pacteau, 2025)

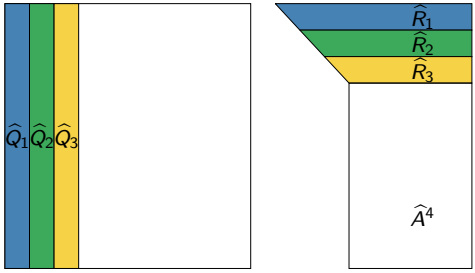
- **Analysis:** consider a truncated QR factorization computed such that $k = \sum_{i=1}^{p} k_i$ Householder transformations are applied to a matrix A, where k_i is the number of transformations computed in precision u_i. The computed $\widehat{R}_i$ and $\widehat{Q}_i$ satisfy

$$\|A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i\| \leq \|A_{p+1}\| + \sum_{i=1}^{p} c u_i \|A_i\|.$$

where A_i is the trailing submatrix after $\sum_{j=1}^{i-1} k_j$ transformations.

⇒ **Algorithm:** balance $\|A_{p+1}\| \approx u_i \|A_i\| \approx \varepsilon \|A\|$

1. start the factorization with $u_1 \leq \varepsilon$
2. if after k_1 transformations $\|A^2\| \leq \varepsilon / u_2 \|A\|$, switch to precision u_2
3. same for precisions $u_2, \ldots, u_p$
4. if after $k_1 + \cdots + k_p$ transformations $\|A^{p+1}\| \leq \varepsilon \|A\|$, stop



AP truncated Householder QR (Buttari, M., Pacteau, 2025)

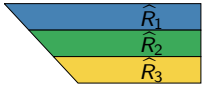
- **Analysis:** consider a truncated QR factorization computed such that $k = \sum_{i=1}^{p} k_i$ Householder transformations are applied to a matrix A, where k_i is the number of transformations computed in precision u_i. The computed $\widehat{R}_i$ and $\widehat{Q}_i$ satisfy

$$\|A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i\| \leq \|A_{p+1}\| + \sum_{i=1}^{p} c u_i \|A_i\|.$$

where A_i is the trailing submatrix after $\sum_{j=1}^{i-1} k_j$ transformations.

⇒ **Algorithm:** balance $\|A_{p+1}\| \approx u_i \|A_i\| \approx \varepsilon \|A\|$

1. start the factorization with $u_1 \leq \varepsilon$
2. if after k_1 transformations $\|A^2\| \leq \varepsilon / u_2 \|A\|$, switch to precision u_2
3. same for precisions $u_2, \ldots, u_p$
4. if after $k_1 + \cdots + k_p$ transformations $\|A^{p+1}\| \leq \varepsilon \|A\|$, stop



$$\|A - \widehat{Q}_1\widehat{R}_1 - \widehat{Q}_2\widehat{R}_2 - \widehat{Q}_3\widehat{R}_3\| \leq c'\varepsilon\|A\|$$

AP truncated Householder QR (Buttari, M., Pacteau, 2025)

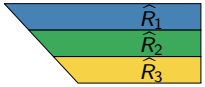
- **Analysis:** consider a truncated QR factorization computed such that $k = \sum_{i=1}^{p} k_i$ Householder transformations are applied to a matrix A, where k_i is the number of transformations computed in precision u_i. The computed $\widehat{R}_i$ and $\widehat{Q}_i$ satisfy

$$\|A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i\| \leq \|A_{p+1}\| + \sum_{i=1}^{p} c u_i \|A_i\|.$$

where A_i is the trailing submatrix after $\sum_{j=1}^{i-1} k_j$ transformations.

⇒ **Algorithm:** balance $\|A_{p+1}\| \approx u_i \|A_i\| \approx \varepsilon \|A\|$

1. start the factorization with $u_1 \leq \varepsilon$
2. if after k_1 transformations $\|A^2\| \leq \varepsilon / u_2 \|A\|$, switch to precision u_2
3. same for precisions $u_2, \ldots, u_p$
4. if after $k_1 + \cdots + k_p$ transformations $\|A^{p+1}\| \leq \varepsilon \|A\|$, stop



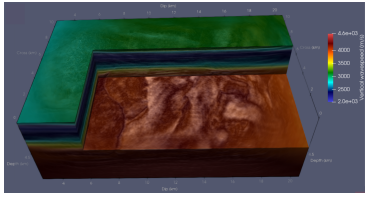
$$\|A - \widehat{Q}_1\widehat{R}_1 - \widehat{Q}_2\widehat{R}_2 - \widehat{Q}_3\widehat{R}_3\| \leq c'\varepsilon\|A\|$$

- Works for any pivoting strategy (e.g., Businger–Golub but also randomized)

AP block low-rank approximation

- We use AP BLR approximations in MUMPS for reducing the LU factor storage cost
- We use 7 custom RP precisions with truncated mantissa and a memory accessor to perform the LU solves (Amestoy, Jego, L'Excellent, M., Pichon, 2025)
- Illustrative application impact (Operto et al., 2023):

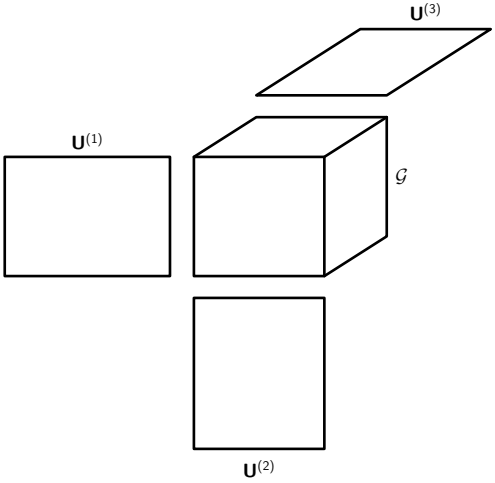
Gorgon Model, reservoir 23km x 11km x 6.5km,
grid size 15m, Helmholtz equation, 25-Hz
Complex matrix, 531 Million dofs



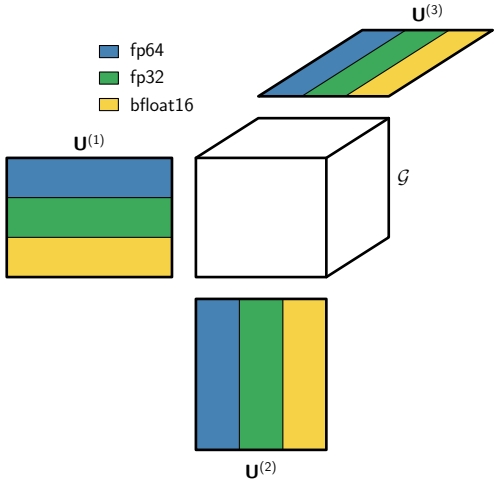
FR (Full-Rank); BLR with $\varepsilon = 10^{-5}$; 48 000 cores (500 MPI $\times$ 96 threads/MPI)
FR: fp32; AP BLR: 3 precisions (fp32, rp24, rp16) for storage

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}

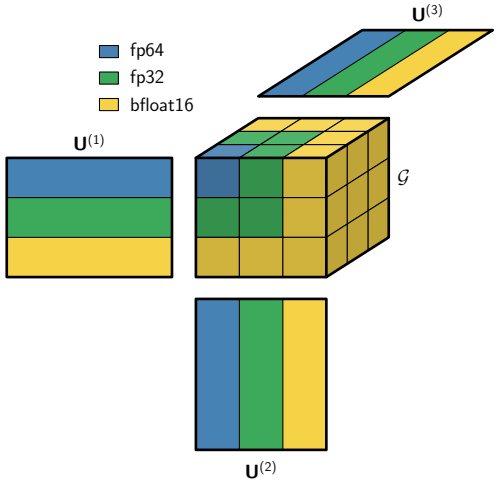
AP Tucker tensor approximation (ongoing work w/ Ballard and Verma)



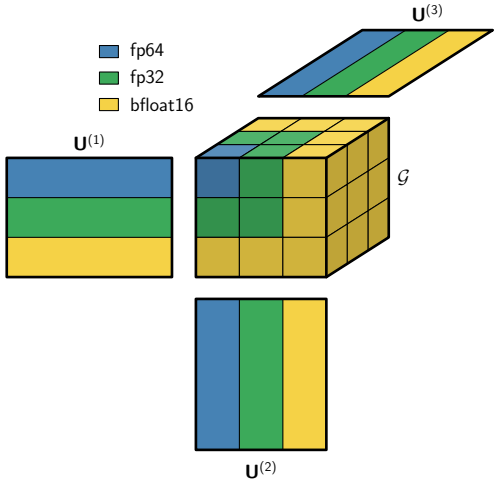
AP Tucker tensor approximation (ongoing work w/ Ballard and Verma)

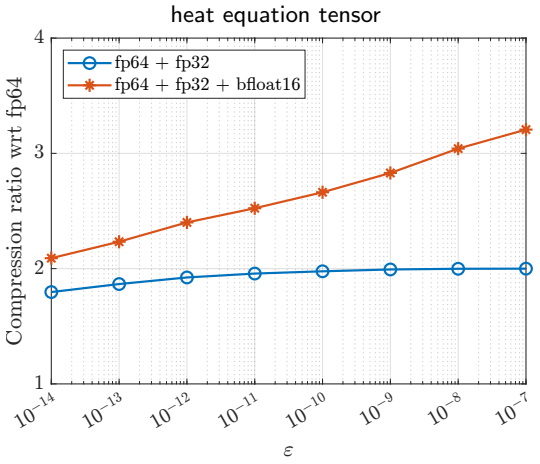


AP Tucker tensor approximation (ongoing work w/ Ballard and Verma)



AP Tucker tensor approximation (ongoing work w/ Ballard and Verma)





- Consider the forward pass on a neural network with L layers:

$$h_\ell = \phi_\ell(W_\ell h_{\ell-1}), \quad \ell = 1, \dots, L$$

where W_ℓ are the weight matrices, ϕ_ℓ are the activation functions, h_0 is the input, and h_L is the output

- **Analysis:** we consider an adaptive precision componentwise matrix–vector product error model:

$$\widehat{h}_\ell = \phi_\ell\big((W_\ell + \Delta W_\ell)\widehat{h}_{\ell-1}\big) \quad |\Delta W_\ell| \le u_\ell \circ |W_\ell|,$$

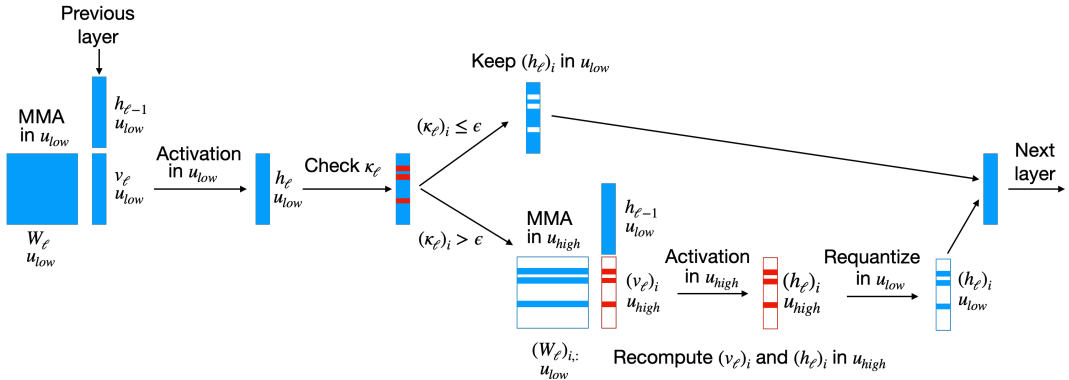
where $\Delta W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $u_\ell \in \mathbb{R}^{n_\ell}$, so that the precisions $(u_\ell)_i$ can vary both on each layer ℓ and each row i. Then the computed output satisfies

$$\widehat{h}_L = h_L \circ (\mathbf{1} + \Delta h_L), \quad \|\Delta h_L\|_\infty \le \sum_{\ell=1}^{L} \left[\Big(\prod_{k=\ell+1}^{L} \|\kappa_k\|_\infty \Big) \|\kappa_\ell \circ u_\ell\|_\infty \right]$$

where $\kappa_\ell \in \mathbb{R}^{n_\ell}$ is the vector of condition numbers of layer ℓ:

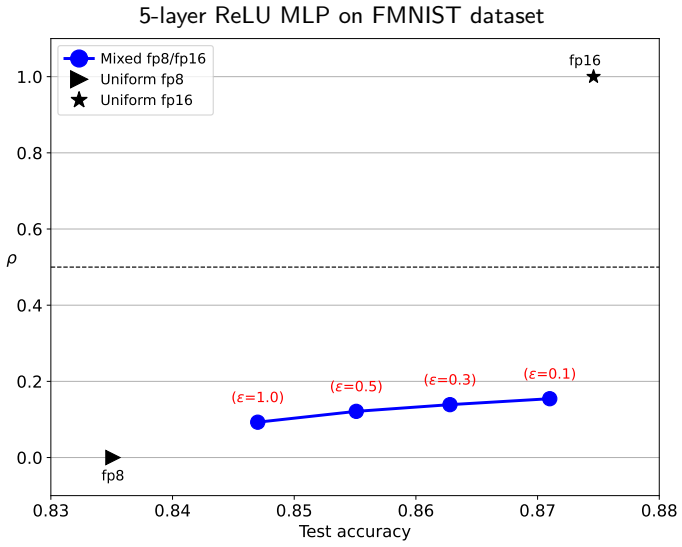
$$\kappa_\ell \approx \phi'_\ell(v_\ell) \oslash \phi_\ell(v_\ell), \quad v_\ell = W_\ell h_{\ell-1}$$

$\Rightarrow$ **Algorithm:** set $u_\ell \propto \mathbf{1} \oslash \kappa_\ell$, where $\kappa_\ell \approx \phi'_\ell(v_\ell) \oslash \phi_\ell(v_\ell)$, $v_\ell = W_\ell h_{\ell-1}$, can be obtained almost as a by-product of the layer output $h_\ell = \phi_\ell(v_\ell)$



- Efficient as long as the fraction ρ of inner products needing to be recomputed is small (e.g., $\rho < 0.5$ when u_{low} is twice faster than u_{high})

AP neural network inference (El Arar, Filip, M., Riccietti, 2025)



Conclusion

Adaptive precision algorithms:

- can exploit as many precisions as there are available
- can achieve a flexible performance–accuracy tradeoff, and thus balance rounding errors with other errors
- due to the combinatorial possibilities, are based on error analysis, which makes them provably accurate and allows for identifying (often all) mixed precision opportunities
- are by necessity dynamic, requiring to monitor key quantities at runtime
- are provably fast, whenever these key quantities are cheap to compute/estimate

and, perhaps most surprisingly,

- seem to be applicable just about anywhere: SpMV, GMRES, preconditioners, factorizations, LRA, tensors, DNNs, ...

Conclusion

Adaptive precision algorithms:

- can exploit as many precisions as there are available
- can achieve a flexible performance–accuracy tradeoff, and thus balance rounding errors with other errors
- due to the combinatorial possibilities, are based on error analysis, which makes them provably accurate and allows for identifying (often all) mixed precision opportunities
- are by necessity dynamic, requiring to monitor key quantities at runtime
- are provably fast, whenever these key quantities are cheap to compute/estimate

and, perhaps most surprisingly,

- seem to be applicable just about anywhere: SpMV, GMRES, preconditioners, factorizations, LRA, tensors, DNNs, ...

$\Rightarrow$ What about *your* favorite algorithms? What gold mine of mixed precision opportunities are we yet to uncover?