InterFLOP Meeting

6 October 2021

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

Theo Mary

Sorbonne Université, CNRS, LIP6

Joint work with

Stef Graillat, **Fabienne Jézéquel**, and **Roméo Molina**

Today's floating-point landscape

		Bits Signif. (t)	Exp.	Range	$u = 2^{-t}$
bfloat16	B	8	8	$10^{\pm 38}$	4×10^{-3}
fp16	H	11	5	$10^{\pm 5}$	5×10^{-4}
fp32	S	24	8	$10^{\pm 38}$	6×10^{-8}
fp64	D	53	11	$10^{\pm 308}$	1×10^{-16}
fp128	Q	113	15	$10^{\pm 4932}$	1×10^{-34}

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

Terminology varies: Mixed precision, Multiprecision, Adaptive precision, Variable precision, Transprecision, Dynamic precision, ...

Mixed precision algorithms

Mix several precisions in the same code with the goal of

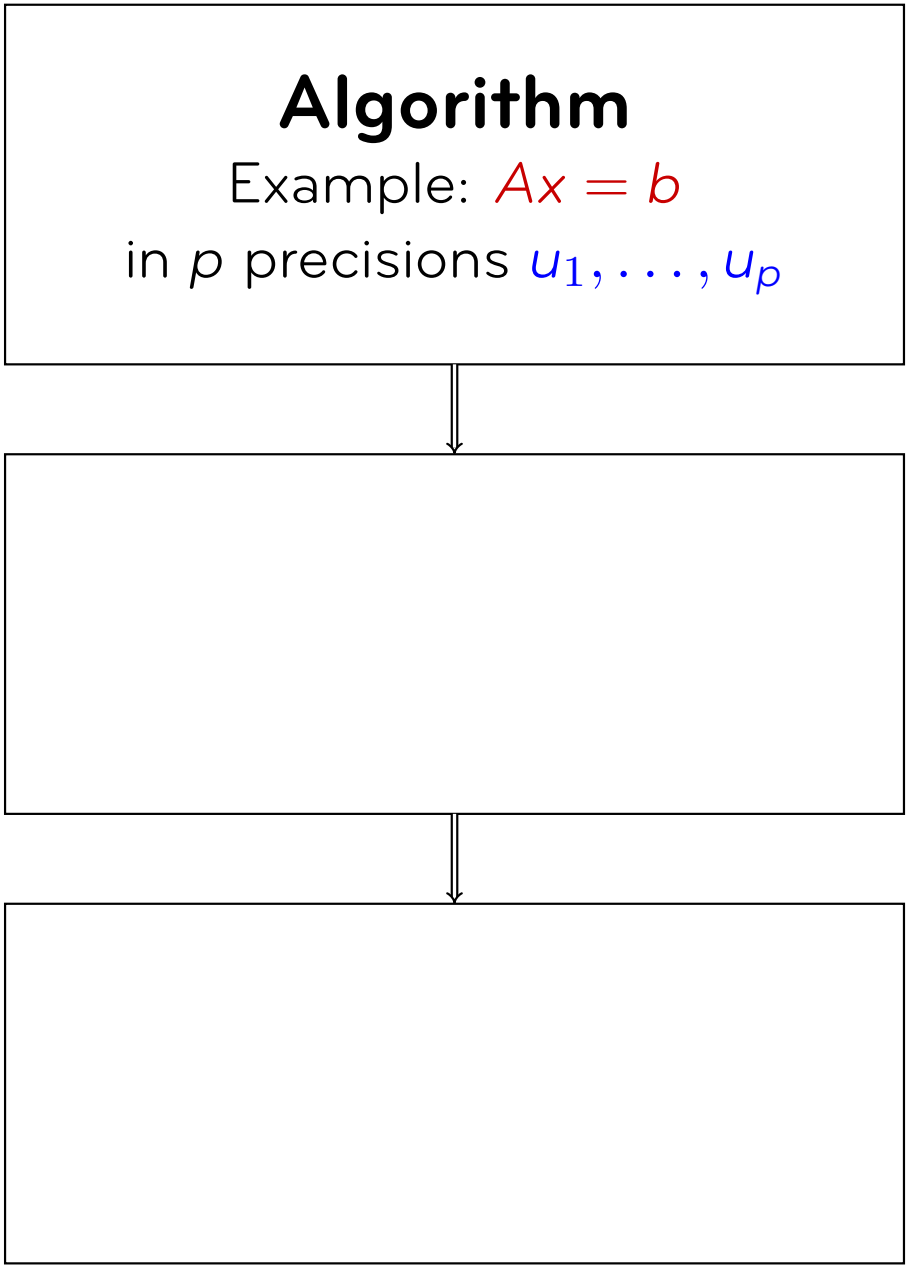
- Getting the performance benefits of low precisions
- While preserving the accuracy and stability of the high precision

Terminology varies: Mixed precision, Multiprecision, Adaptive precision, Variable precision, Transprecision, Dynamic precision, ...

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

- **Precision tuning:** autotuning based on the source code
  - ▲ Does not need any understanding of what the code does
  - ▼ Does not have any understanding of what the code does
- **This work:** **exploit as much as possible the knowledge we have about the code**

The AAA approach (1/3)

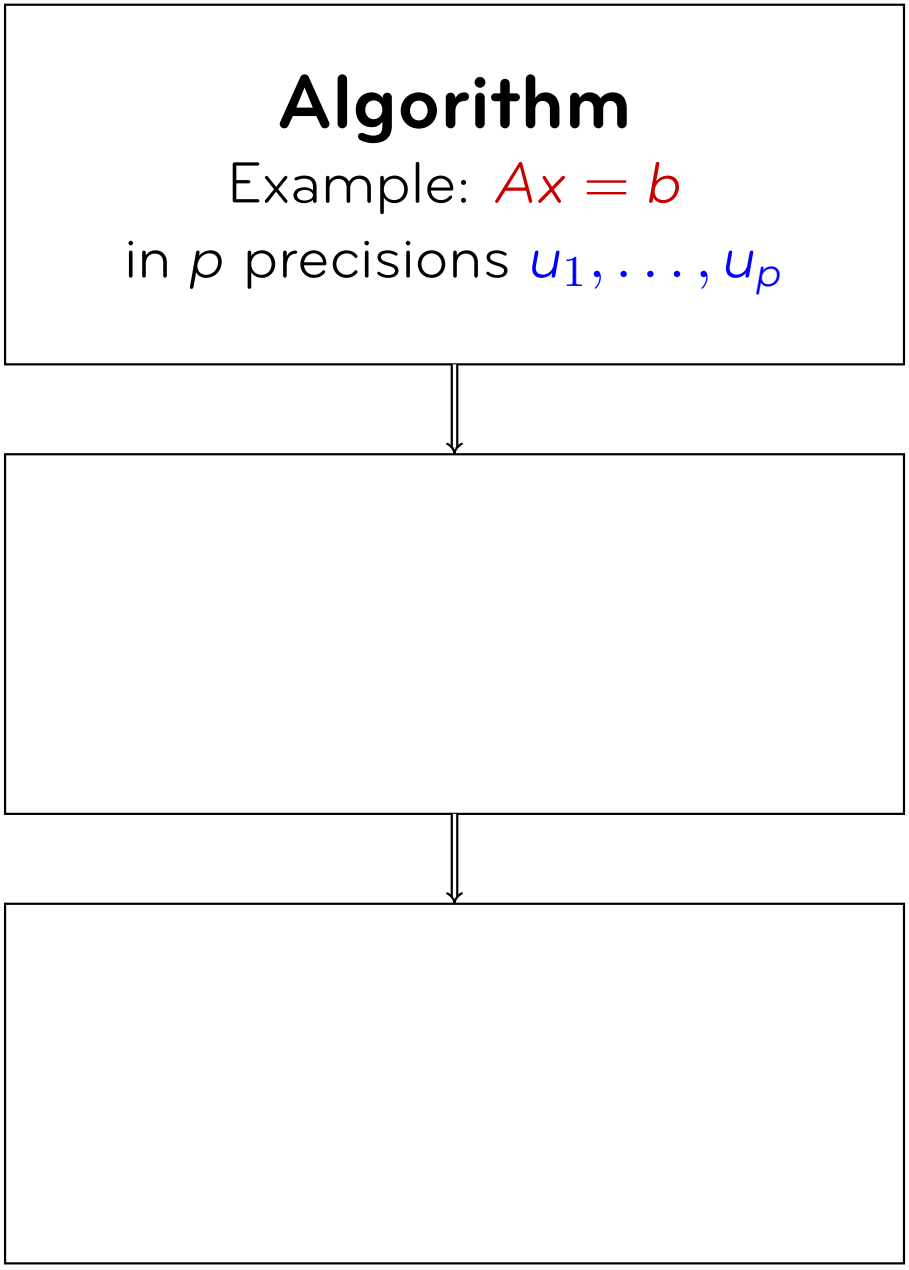


Iterative refinement

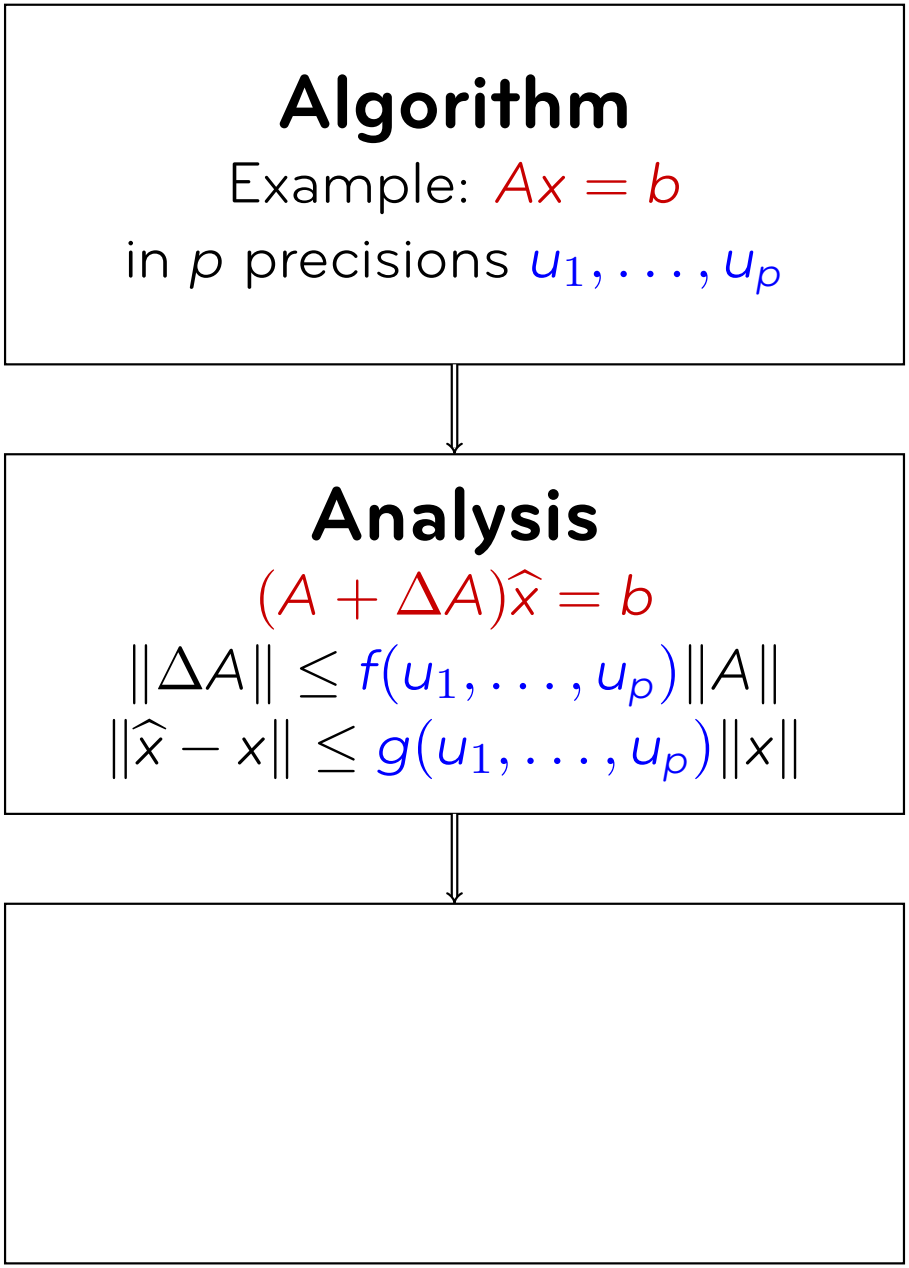
Factorize $A = LU$ **at low precision**
Solve $Ax_1 = b$ via $x_1 = U^{-1}(L^{-1}b)$
repeat **at high precision**
    $r_i = b - Ax_i$
    Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$
    $x_{i+1} = x_i + d_i$
until converged

- $O(n^3)$ flops at low precision, $O(n^2)$ flops per iteration at high precision
- What guarantees of convergence? What does "low" and "high" mean?

The AAA approach (2/3)



The AAA approach (2/3)



Error analysis of iterative refinement

Factorize $A = LU$ **at precision $\mathbf{u_f}$**
Solve $Ax_1 = b$ via $x_1 = U^{-1}(L^{-1}b)$ **at precision $\mathbf{u_f}$**
repeat
  $r_i = b - Ax_i$ **at precision $\mathbf{u_r}$**
  Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ **at precision $\mathbf{u_f}$**

  $x_{i+1} = x_i + d_i$ **at precision $\mathbf{u}$**
until converged

- Theorem from Carson and Higham (2018) : provided that $\kappa(A)\mathbf{u_f} < 1$, we reach $\|\widehat{x} - x\| \leq (\mathbf{u} + \mathbf{u_r}\kappa(A))\|x\|$

  $\Rightarrow$ **Up to $\kappa(A) = O(10^3)$ for fp16 factorization**

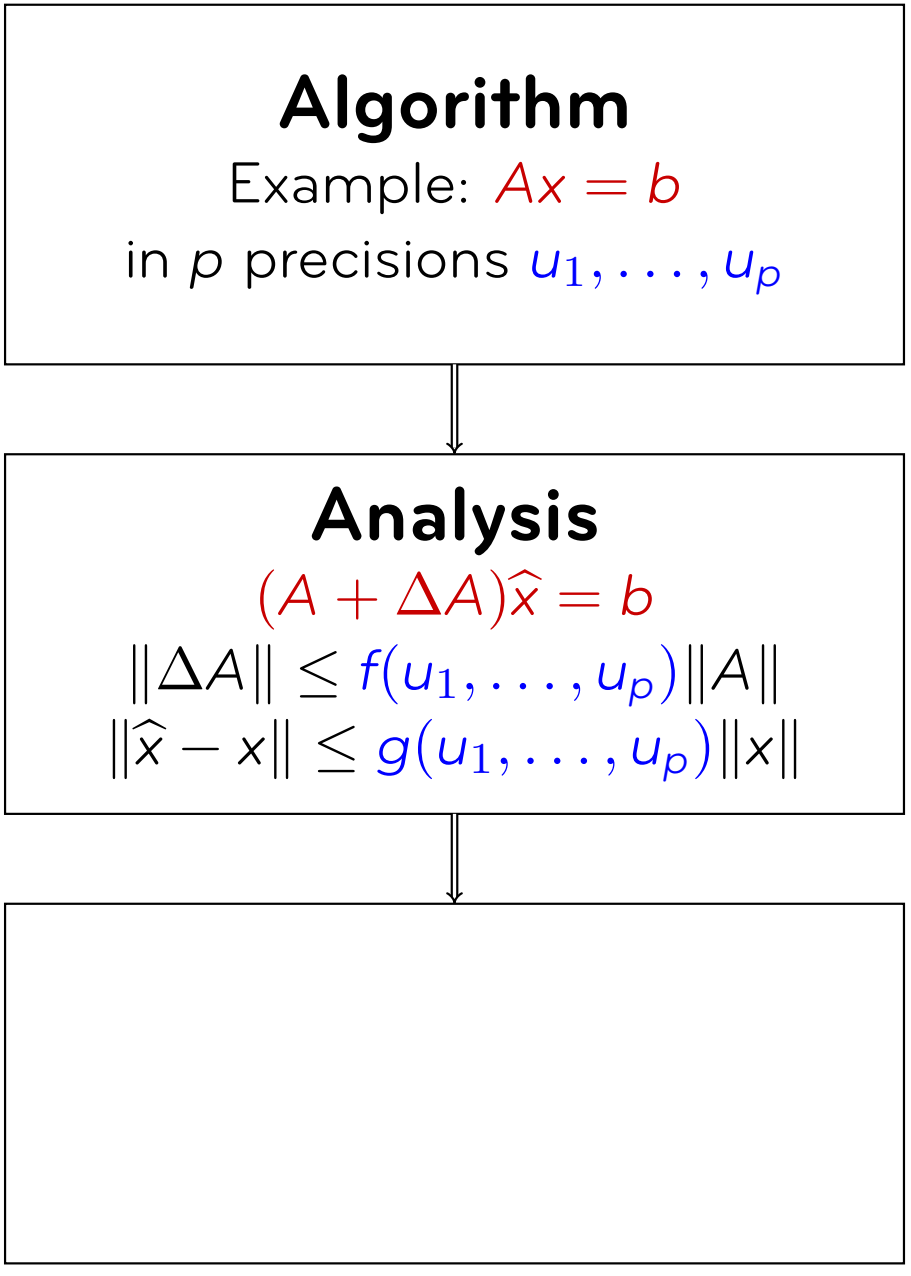
Error analysis of iterative refinement

Factorize $A = LU$ **at precision $\mathbf{u_f}$**
Solve $Ax_1 = b$ via $x_1 = U^{-1}(L^{-1}b)$ **at precision $\mathbf{u_f}$**
repeat
  $r_i = b - Ax_i$ **at precision $\mathbf{u_r}$**
  Solve $Ad_i = r_i$ with preconditioned GMRES
    at precision $\mathbf{u_g}$ except matvecs **at precision $\mathbf{u_p}$**
  $x_{i+1} = x_i + d_i$ **at precision $\mathbf{u}$**
until converged

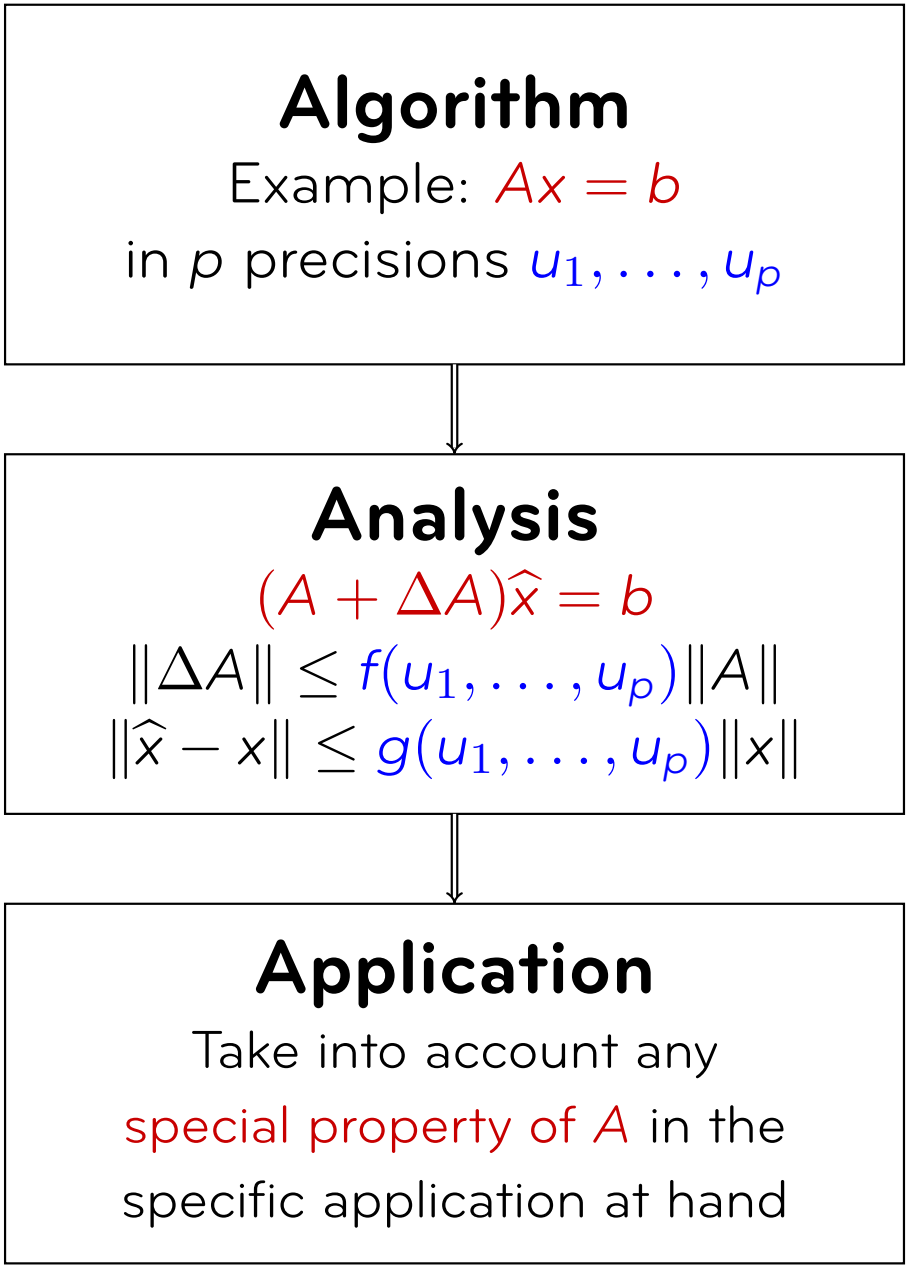
- Theorem from Carson and Higham (2018) : provided that $\kappa(A)\mathbf{u_f} < 1$, we reach $\|\widehat{x} - x\| \leq (\mathbf{u} + \mathbf{u_r}\kappa(A))\|x\|$
  $\Rightarrow$ **Up to $\kappa(A) = O(10^3)$ for fp16 factorization**

- Theorem from Amestoy et al. (2021) : provided that $(\mathbf{u_g} + \kappa(A)\mathbf{u_p})\kappa(A)^2{\mathbf{u_f}}^2 < 1$ we reach the same accuracy
  $\Rightarrow$ **Up to $\kappa(A) = O(10^{11})$ for fp16 factorization**

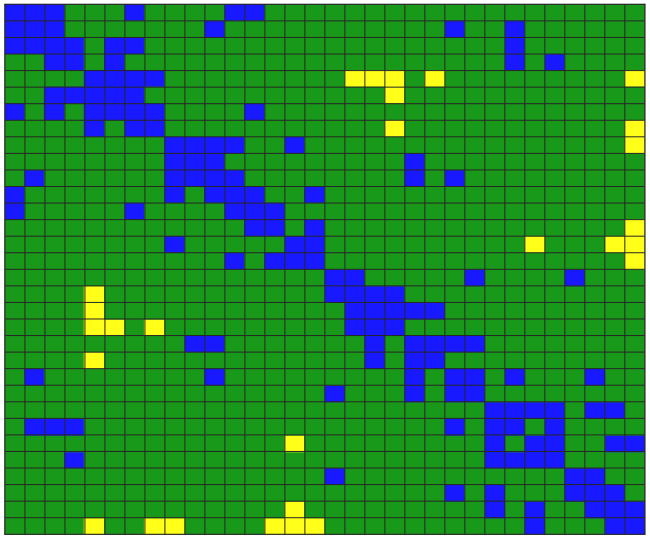
The AAA approach (3/3)



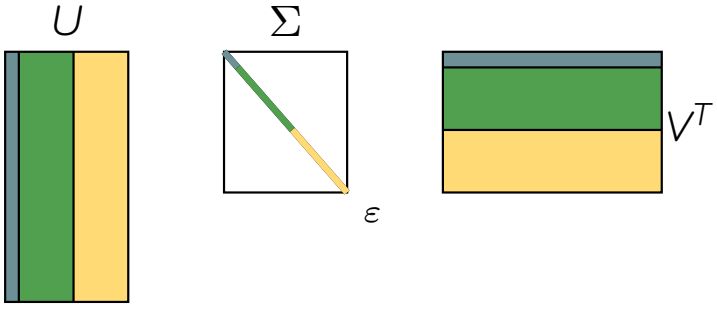
The AAA approach (3/3)



Block low-rank (BLR) matrices



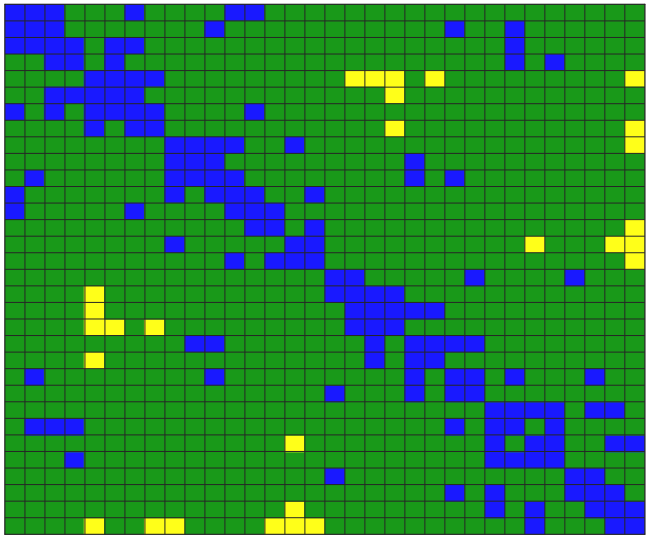
double single half



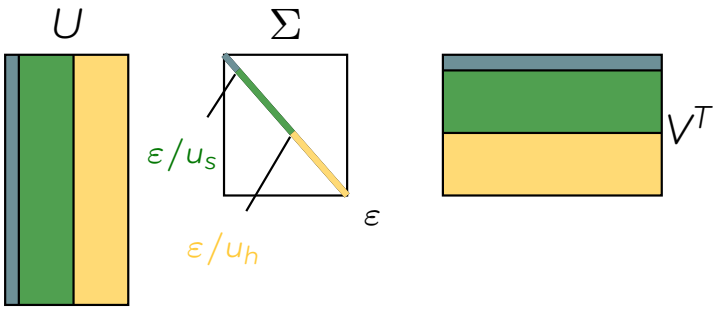
- BLR matrices arise in several applications (PDEs, covariance matrices, etc.)
- Blocks are compressed with low-rank approximations, e.g. via truncated SVD

Why does mixed precision work here ?

Block low-rank (BLR) matrices



double single half



- BLR matrices arise in several applications (PDEs, covariance matrices, etc.)
- Blocks of norm less than ε/u_i can be stored in precision u_i
- Blocks are compressed with low-rank approximations, e.g. via truncated SVD
- Singular vectors associated with singular values less than ε/u_i can be stored in precision u_i

Why does mixed precision work here ?

Amestoy et al. (2021b)

Data-driven adaptive precision algorithms

- Why can we store "less important" data in lower precision ? Because small elements produce small errors :

$$|\mathrm{fl}(a \text{ op } b) - a \text{ op } b| \leq u|a \text{ op } b|, \qquad \text{op} \in \{+, -, *, \div\}$$

⇒ **Opportunity for mixed precision !**

- BLR matrices Amestoy et al. (2021b) Abdulah et al. (2021)
- Low-rank approximations, SVD Amestoy et al. (2021b)
- Randomized SVD Connolly, Higham, and Pranesh (SIAM AN 2021)
- Quantized dot product Diffenderfer, Osei-Kuffuor, & Menon (2021)
- SpMV Ahmad, Sundar and Hall (2020)
- Multiword matrix multiplication Fasi et al. (SIAM LA 2021)
- Random matrix multiplication Higham and Mary (2020)
- Block Jacobi and SPAI preconditioners Anzt et al. (2019)
- Runge Kutta Croci and de Souza (2021)

Adaptive precision at the variable level ?

- Pushing adaptive precision to the extreme: can we benefit from storing **each variable** in a (possibly) different precision?
- Same granularity as precision autotuning, but different method to select precisions
- Example: $Ax = b$ with adaptive precision for each A_{ij}
  - **Is it worth it ?**
    Need to have elements of widely different magnitudes, and yet not structured in any obvious way (by blocks or columns, etc.)
  - **Is it practical ?**
    Probably not for compute-bound applications, but could it work for memory-bound ones?

  $\Rightarrow$ Natural candidate: **sparse matrices**

Sparse matrix–vector product (SpMV)

$$y = Ax,\ A \in \mathbb{R}^{m \times n}$$

for $i = 1 : m$ **do**
  $y_i = 0$
  for $j \in nnz_i(A)$ **do**
    $y_i = y_i + a_{ij} x_j$
  end for
end for

- Standard backward error analysis: if $y = Ax$ is performed in a uniform precision ε, we obtain

$$|\widehat{y}_i - y_i| \leq n_i \varepsilon \sum_{j \in nnz_i(A)} |a_{ij} x_j|$$

- **Idea:** store elements of A in a precision inversely proportional to their magnitude (**smaller elements in lower precision**)

Adaptive precision SpMV

for $i = 1 : m$ **do**
  $y_i = 0$
  for $k = 1 : p$ **do**
    $y_i^{(k)} = 0$
    for $j \in nnz_i(A)$ **do**
      if $a_{ij}x_j \in B_{ik}$ **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 row i of A into p buckets B_{ik} and sum elements of B_{ik} in precision u_k
- Backward error analysis: $|\widehat{y}_i^{(k)} - y_i^{(k)}| \leq n_i^{(k)} u_k \sum_{a_{ij}x_j \in B_{ik}} |a_{ij}x_j|$

Building the buckets

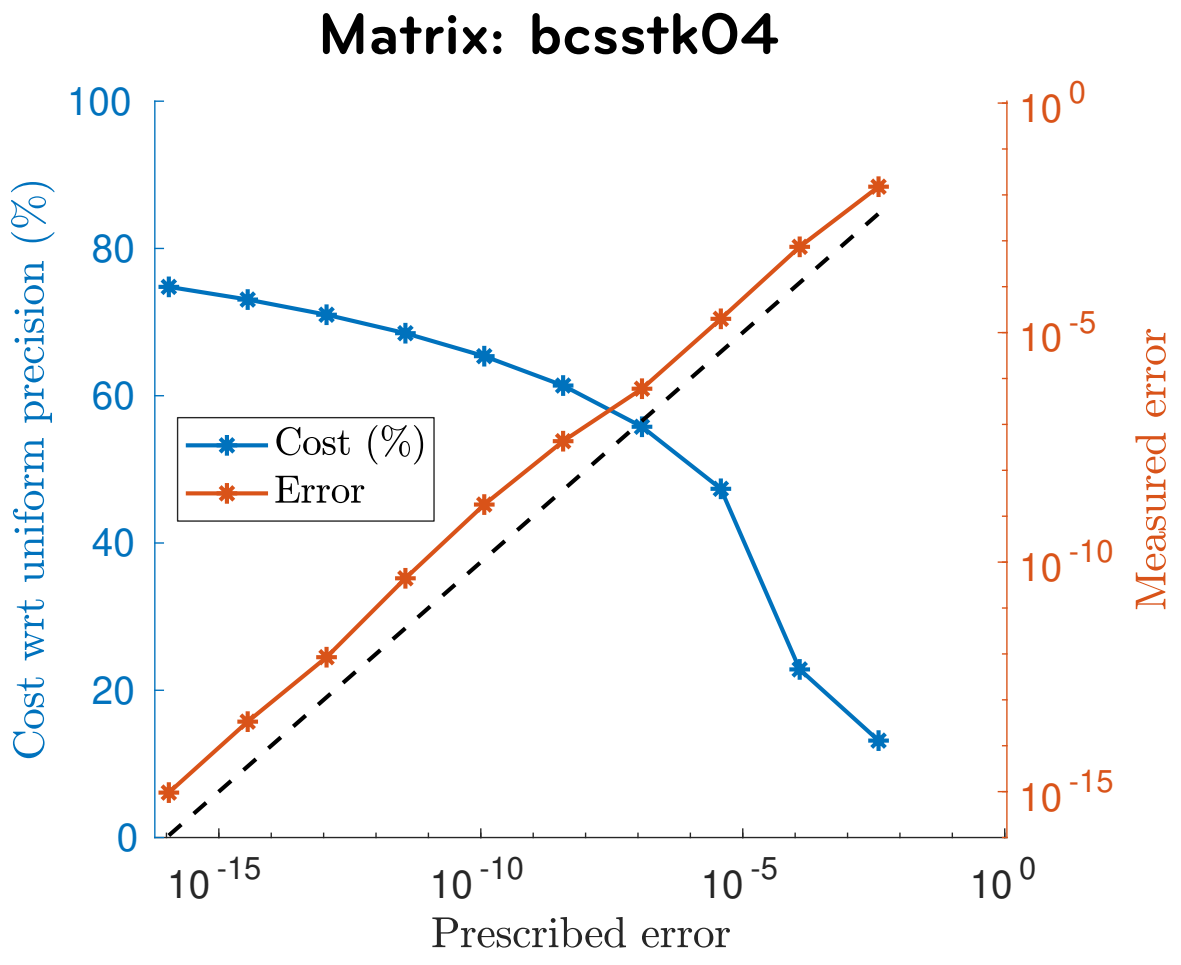
- $|\widehat{y}_i^{(k)} - y_i^{(k)}| \leq n_i^{(k)} u_k \sum_{a_{ij}x_j \in B_{ik}} |a_{ij}x_j|$

$\Rightarrow$ Build the buckets such that $u_k \sum_{a_{ij}x_j \in B_{ik}} |a_{ij}x_j| \approx \varepsilon \sum_j |a_{ij}x_j|$

- By setting B_{ik} to the interval $(\varepsilon\beta_i/u_{k+1}, \varepsilon\beta_i/u_k]$, we obtain $|\widehat{y}_i^{(k)} - y_i^{(k)}| \leq n_i^{(k)} \varepsilon\beta_i$ and so $|\widehat{y}_i - y_i| \leq n_i \varepsilon\beta_i$

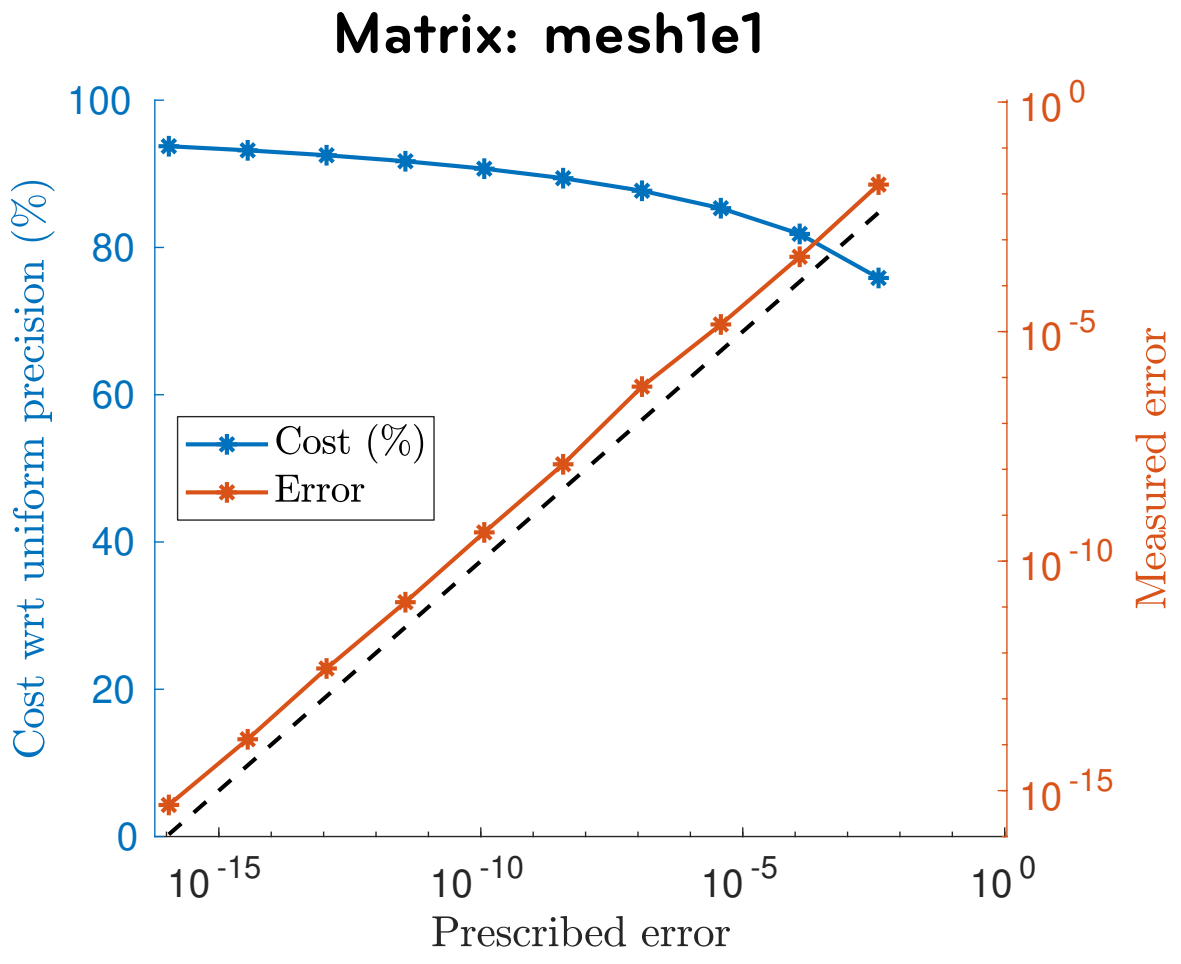
- Two possible choices for β_i:
  - $\beta_i = \sum_j |a_{ij}x_j| \Rightarrow$ guarantees $O(\varepsilon)$ componentwise backward error
  - $\beta_i = \|A\|\|x\| \Rightarrow$ guarantees $O(\varepsilon)$ normwise backward error

SpMV: experiments



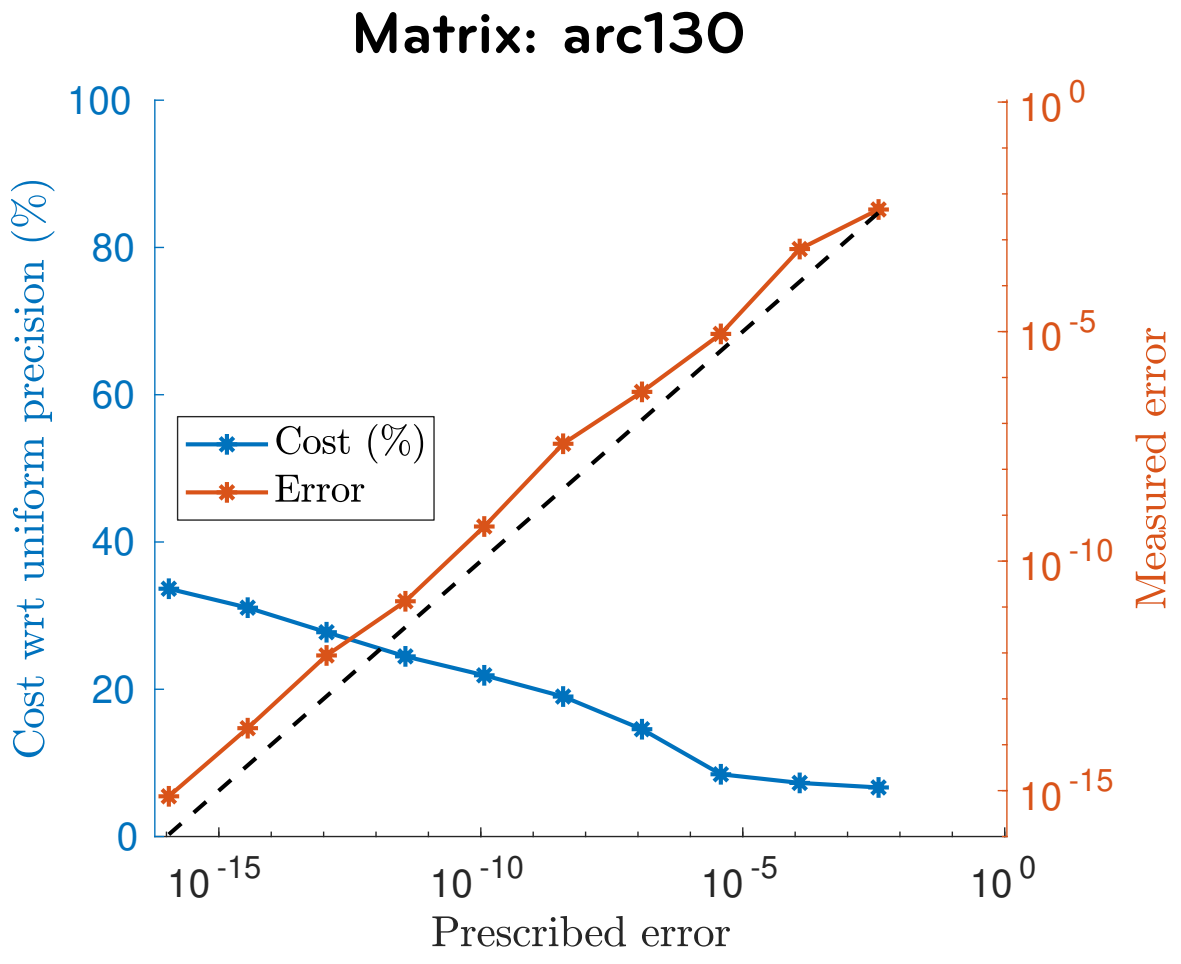
Results for $x = (1, \ldots, 1)^T$

SpMV: experiments



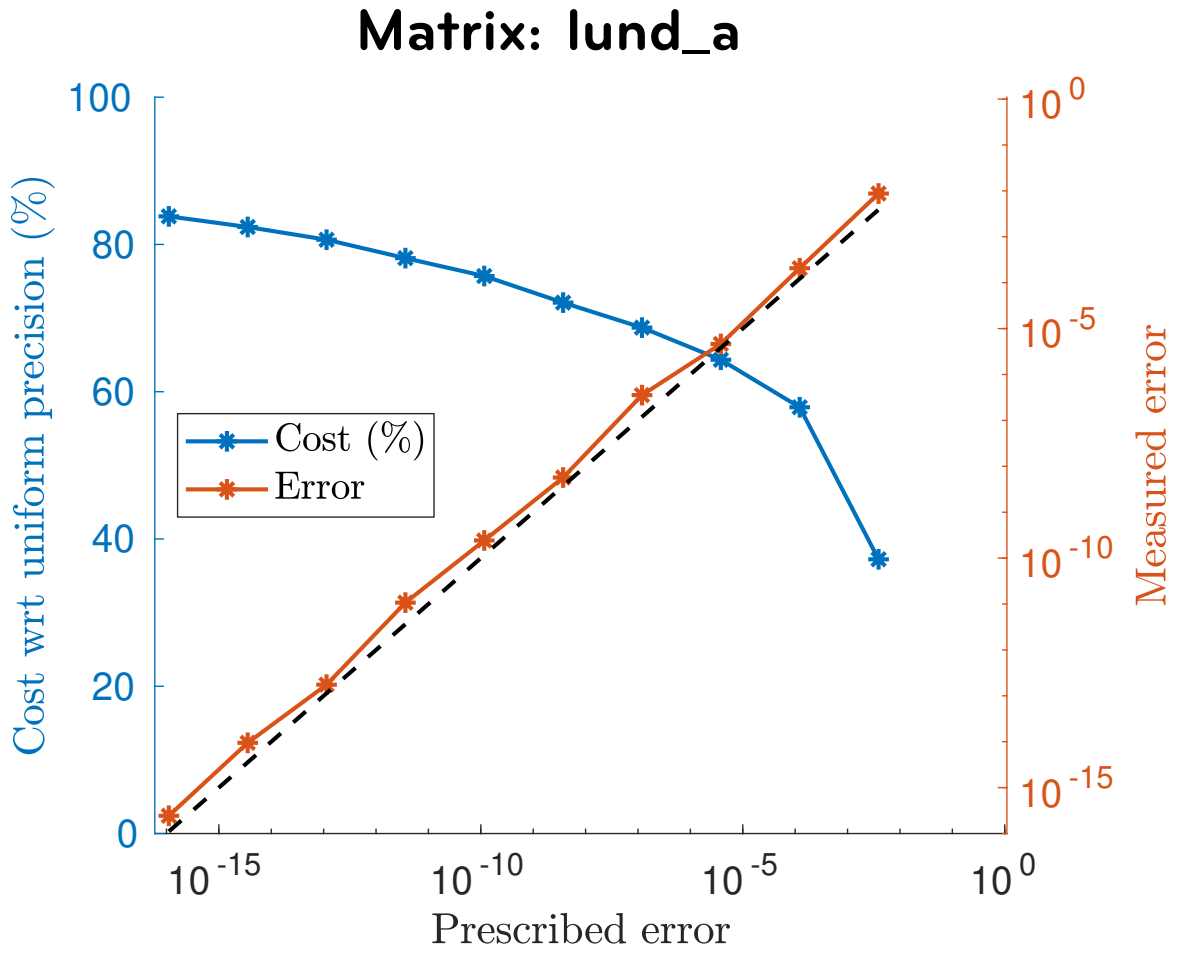
Results for $x = (1, \ldots, 1)^T$

SpMV: experiments



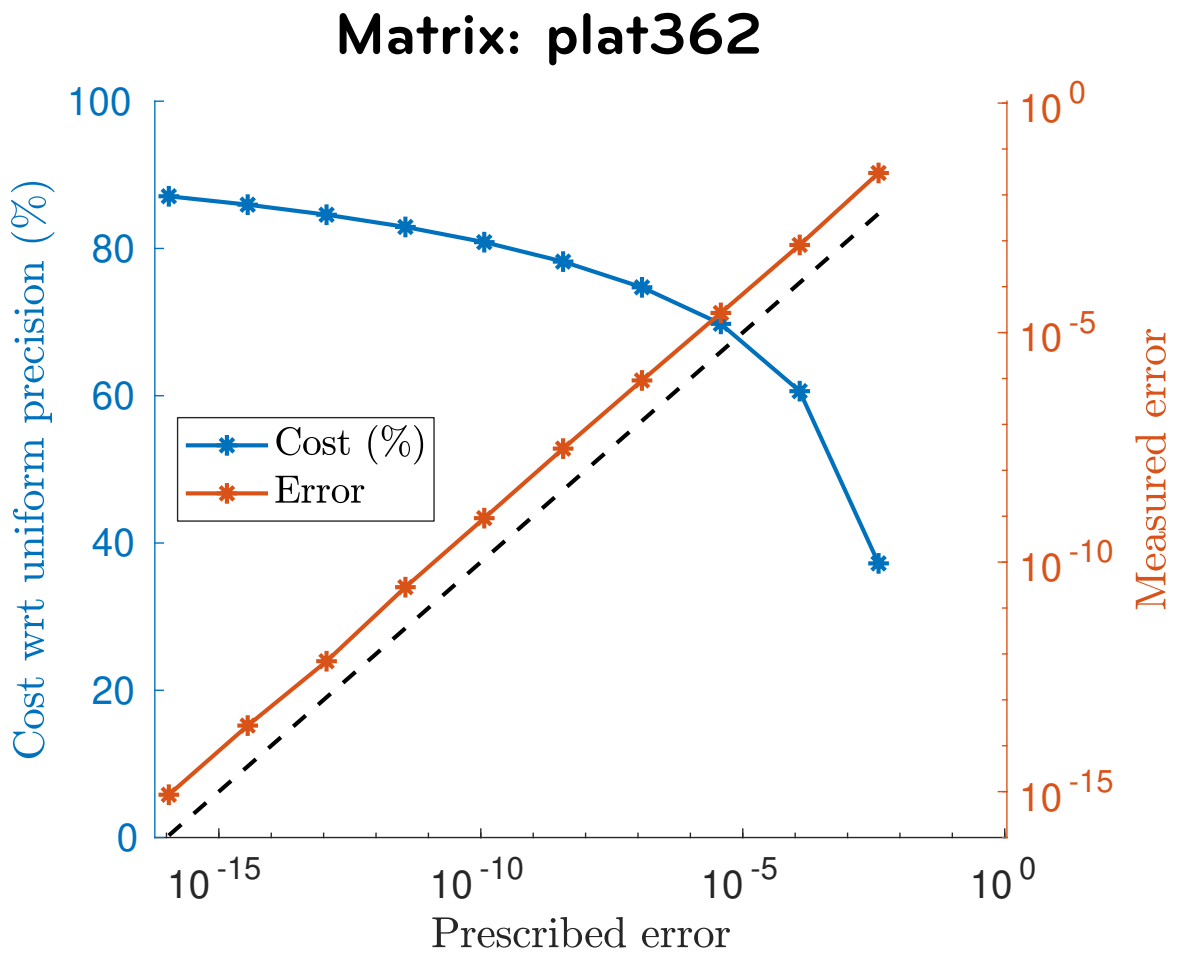
Results for $x = (1, \ldots, 1)^T$

SpMV: experiments



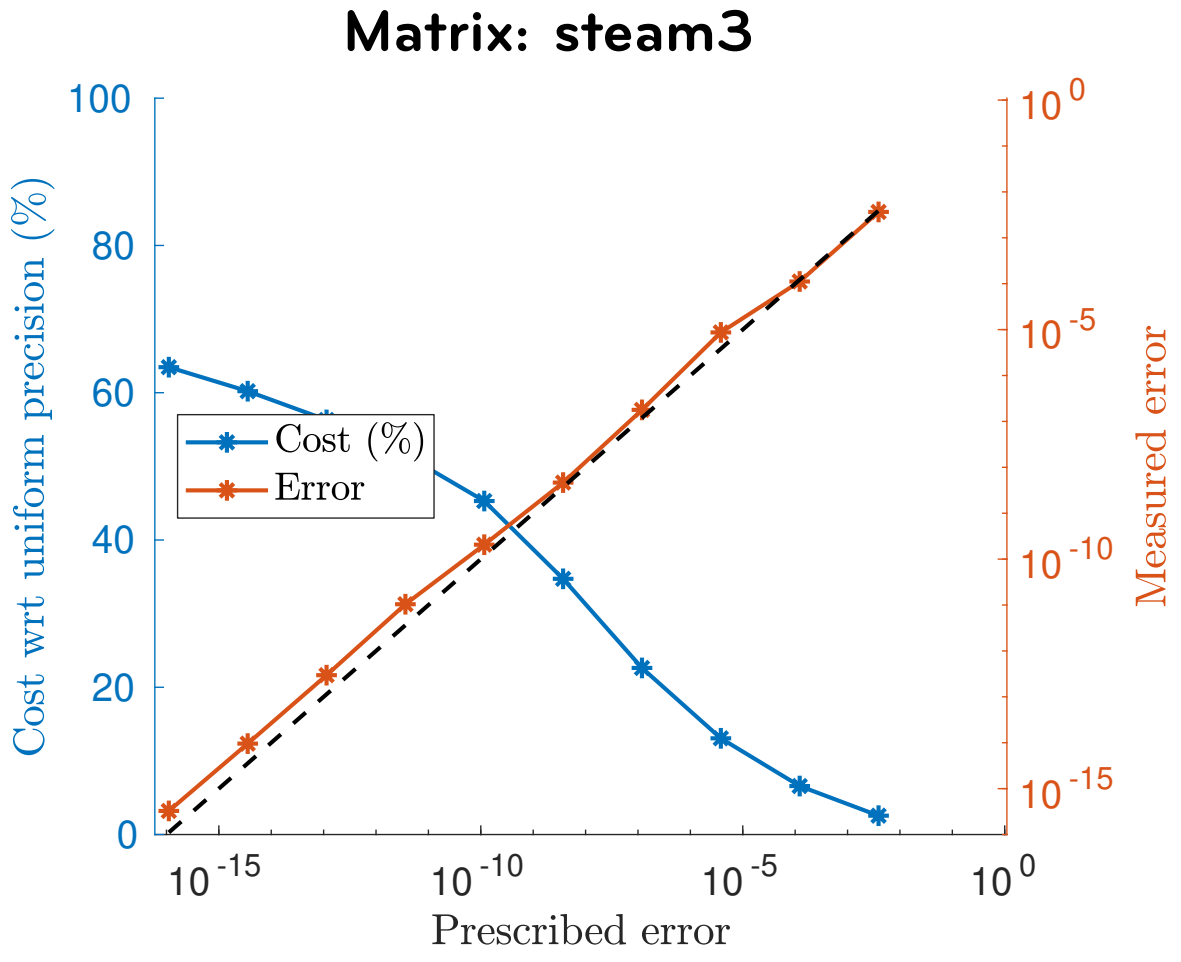
Results for $x = (1, \ldots, 1)^T$

SpMV: experiments



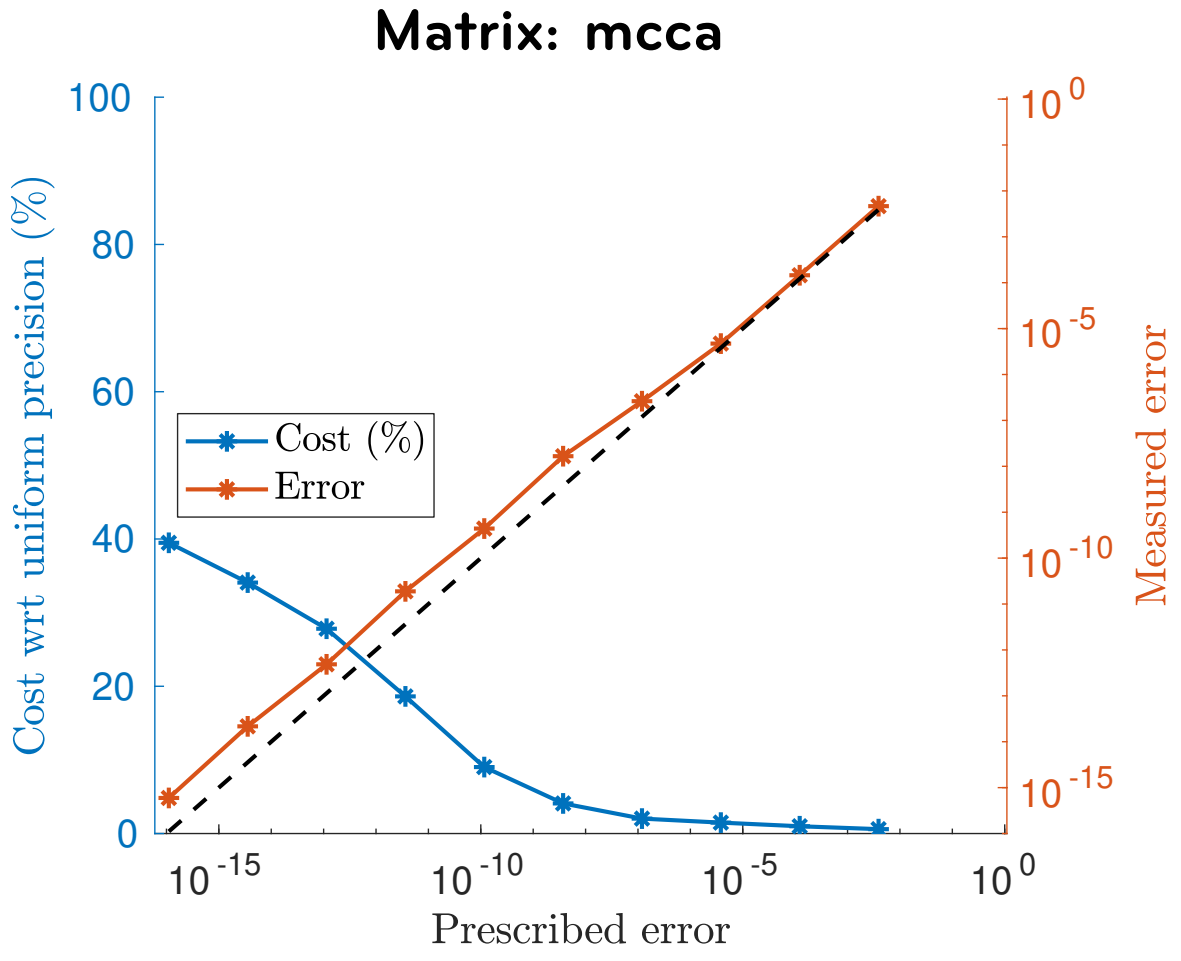
Results for $x = (1, \ldots, 1)^T$

SpMV: experiments



Results for $x = (1, \ldots, 1)^T$

SpMV: experiments



Results for $x = (1, \ldots, 1)^T$

Role of vector x

- Critical issue: accuracy of SpMV depends on x, but not practical to change precision of A based on x
- Can still use it and cross fingers ...
- More promising avenue: use it in a setting where x is guaranteed to be "nice"

Role of vector x

- Critical issue: accuracy of SpMV depends on x, but not practical to change precision of A based on x
- Can still use it and cross fingers ...
- More promising avenue: use it in a setting where x is guaranteed to be "nice"

$r = b - Ax_0$
$\beta = \|r\|_2$
$q_1 = r/\beta$
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 the least squares problem $\min_{c_k} \|Hc_k - \beta e_1\|_2$
  $x_k = x_0 + Q_k c_k$
end for

$\Rightarrow$ **Krylov solvers**! In GMRES, x is orthonormal

Application to GMRES: experimental setting

Difficulties in assessing the potential of adaptive precision for GMRES:

- **Highly matrix dependent**, need to cover a wide range of applications
- For a given matrix, **hard to know what a good accuracy is**
  - What storage precision?
  - What tolerance threshold for GMRES convergence?
  - Normwise or componentwise stable SpMV?
    We have both $\eta_{\text{fwd}} \leq \kappa(A)\eta_{\text{bwd}}^{\text{nrm}}$ and $\eta_{\text{fwd}} \leq \text{cond}(A)\eta_{\text{bwd}}^{\text{cmp}}$, where $\text{cond}(A) = \||A^{-1}||A|\| \leq \kappa(A) = \|A\|\|A^{-1}\|$
    If $\text{cond}(A) \ll \kappa(A)$, componentwise stability may be critical
  - How small should the forward error be?
- Comparison further muddled by possible use of
  - Preconditioners
  - Iterative refinement (i.e., restarted GMRES)

Application to GMRES: an example

- Results with matrix arc130 ($n = 130$, $\kappa(A) = 1.2 \times 10^{12}$, $\mathrm{cond}(A) = 2.2 \times 10^{6}$)
- Use unpreconditioned unrestarted GMRES with A **stored in precision** ε and with **convergence tolerance** τ
- Compare three variants
  - U: store A in uniform precision ε
  - AC: store A in adaptive precision with $\eta_{\mathrm{bwd}}^{\mathrm{cmp}} = \varepsilon$ (compwise stability)
  - AN: store A in adaptive precision with $\eta_{\mathrm{bwd}}^{\mathrm{nrm}} = \varepsilon$ (normwise stability)

Application to GMRES: an example

ε	τ	Iter.	Cost (% U) AC	Cost (% U) AN	$\eta_{\text{bwd}}^{\text{nrm}}$ U/AC/AN	$\eta_{\text{bwd}}^{\text{cmp}}$ U/AC/AN	η_{fwd} U/AC/AN
2^{-53}	10^{-14}	15	57	37	10^{-16}	10^{-10}	10^{-5}
	10^{-12}	13	57	37	10^{-13}	10^{-7}	10^{-3}
	10^{-10}	10	57	37	10^{-11}	10^{-5}	10^{-1}
					U/AC AN	U/AC AN	U/AC AN
2^{-37}	10^{-14}	15	50	28	10^{-16} 10^{-11}	10^{-9} 10^{-5}	10^{-5} 10^{-1}
	10^{-12}	13	50	28	10^{-13} 10^{-11}	10^{-7} 10^{-5}	10^{-3} 10^{-1}
	10^{-10}	10	50	28	10^{-11} 10^{-11}	10^{-5} 10^{-5}	10^{-1} 10^{-1}
2^{-24}	10^{-14}	15	43	20	10^{-12} 10^{-7}	10^{-7} 10^{-1}	10^{-2} 10^{2}
	10^{-12}	13	43	20	10^{-12} 10^{-7}	10^{-7} 10^{-1}	10^{-2} 10^{2}
	10^{-10}	10	43	20	10^{-11} 10^{-7}	10^{-5} 10^{-1}	10^{-1} 10^{2}

- Use of AC/AN does not increase iterations
- Both AC/AN achieve significant gains, AN more so
- As ε increases, U/AC become more accurate than AN
- As τ increases, gap between U/AC and AN closes

$\Rightarrow$ **No clear winner**

Application to GMRES: an example

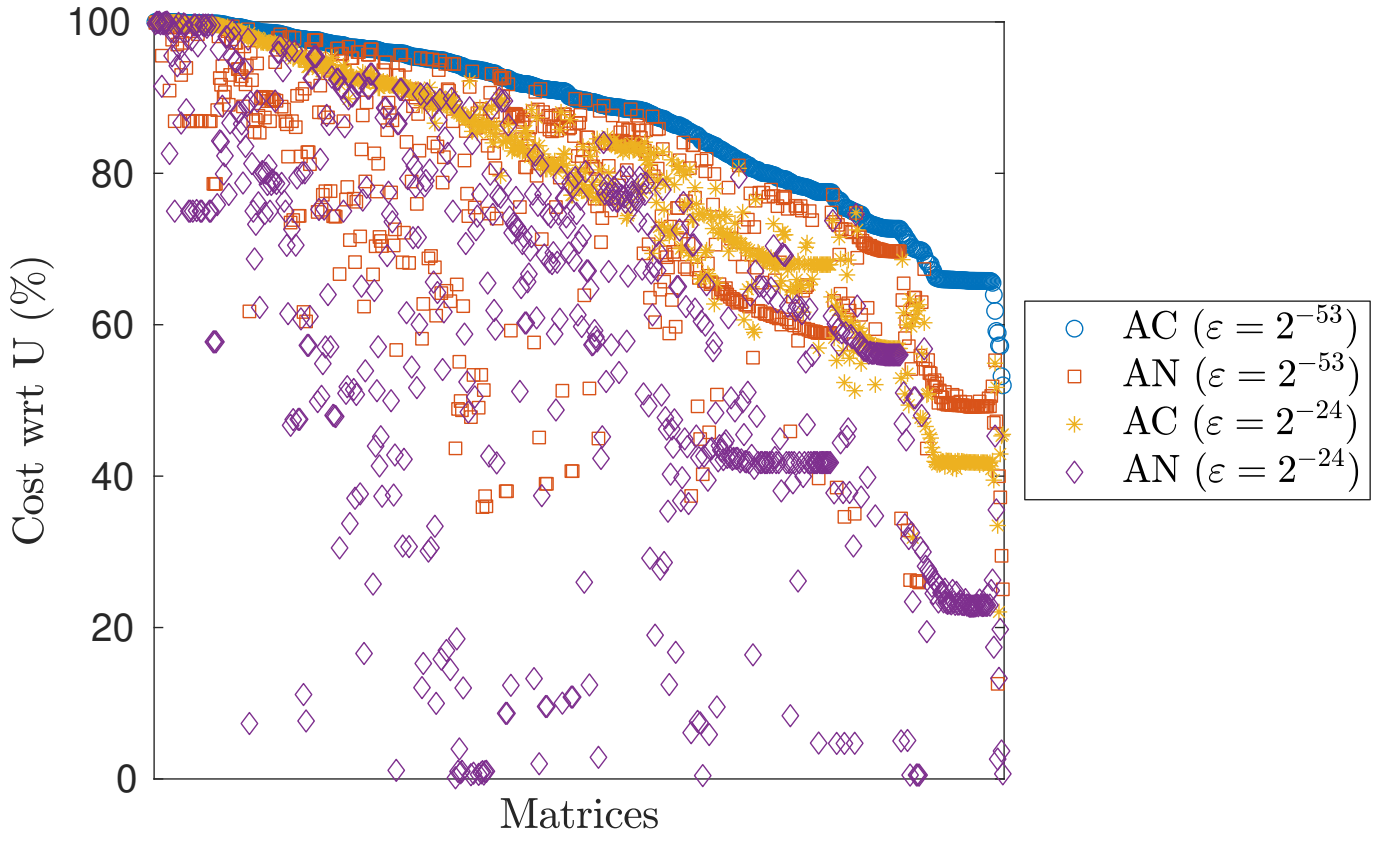
ε	τ	Iter.	Cost (% U) AC	Cost (% U) AN	$\eta_{\text{bwd}}^{\text{nrm}}$ U/AC/AN	$\eta_{\text{bwd}}^{\text{cmp}}$ U/AC/AN	η_{fwd} U/AC/AN
2^{-53}	10^{-14}	15	57	37	10^{-16}	10^{-10}	10^{-5}
	10^{-12}	13	57	37	10^{-13}	10^{-7}	10^{-3}
	10^{-10}	10	57	37	10^{-11}	10^{-5}	10^{-1}
					U/AC AN	U/AC AN	U/AC AN
2^{-37}	10^{-14}	15	50	28	10^{-16} 10^{-11}	10^{-9} 10^{-5}	10^{-5} 10^{-1}
	10^{-12}	13	50	28	10^{-13} 10^{-11}	10^{-7} 10^{-5}	10^{-3} 10^{-1}
	10^{-10}	10	50	28	10^{-11} 10^{-11}	10^{-5} 10^{-5}	10^{-1} 10^{-1}
2^{-24}	10^{-14}	15	43	20	10^{-12} 10^{-7}	10^{-7} 10^{-1}	10^{-2} 10^{2}
	10^{-12}	13	43	20	10^{-12} 10^{-7}	10^{-7} 10^{-1}	10^{-2} 10^{2}
	10^{-10}	10	43	20	10^{-11} 10^{-7}	10^{-5} 10^{-1}	10^{-1} 10^{2}

- If target $\eta_{\text{fwd}} = 10^{-5}$:
  - AN costs $53 \times 15 \times 37$, AC costs $37 \times 15 \times 50$

  $\Rightarrow$ Similar cost, both about 50% better than U
- If target $\eta_{\text{fwd}} = 10^{-1}$:
  - AN costs $37 \times 10 \times 28$, AC costs $24 \times 10 \times 43$

  $\Rightarrow$ Similar cost, both about 57% better than U

Application to GMRES: results on real-life matrices



- Results on 659 SuiteSparse matrices of order $n \in [100, 5000]$
- Keep only matrices for which AC ($\varepsilon = 2^{-53}$) achieves some gain (70% of the matrices)

Conclusion: take-home messages

- The AAA approach:
  - Tailor the use of mixed precision to a specific algorithm class
  - **Use error analysis as a guide** to choose the precisions
  - **Adapt precisions to the data** at hand

- Illustration of this methodology for NLA
  - **Adaptive precision SpMV**
  - Application to Krylov solvers: significant reductions of the data movement at equivalent accuracy
  - Article in preparation

Thank you! Any questions?