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

Roméo Molina[1,2] Stef Graillat[1] Fabienne Jézéquel[1,3] Theo Mary[1]

[1]Sorbonne Université, CNRS, LIP6, Paris, France [2]Laboratoire de Physique des 2 infinis Irène Joliot-Curie – IJCLab, CNRS, Orsay, France [3]Université Paris-Panthéon-Assas, France

For more details:

Today's floating-point landscape

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

- Low precision increasingly supported
- **Great benefits:**
  - Reduced **storage**
  - Reduced **energy**
  - Increased **speed**
- **Some limitations:**
  - Low accuracy (large u)
  - Narrow range

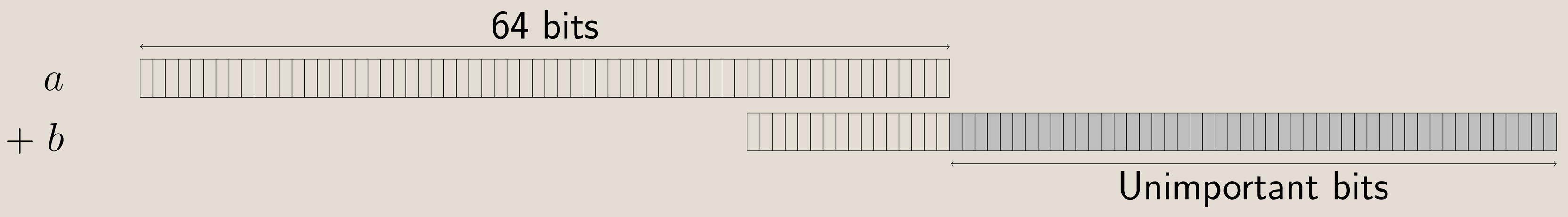
Mixed precision algorithms

Objectives:

- Performance benefits of low precisions
- Accuracy and stability of high precisions

Opportunity for mixed precision:

⇒ **all computations are not equally "important"!**



⇒ We can adapt the precisions to the data at hand

Uniform vs adaptive precision SpMV

$$A \in \mathbb{R}^{m \times n}, x \in R^n$$

Uniform precision SpMV

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

$$|\widehat{y}_i - y_i| \le \#\mathrm{nnz}_i(A)\varepsilon \sum_{j \in nnz_i(A)} |a_{ij}x_j|$$

Idea: For each row i of A, split the set of j-indices into q buckets B_{ik} and sum the corresponding elements a_{ij} in precision u_k

Adaptive precision SpMV

for $i = 1: m$ **do**
  for $k = 1: q$ **do**
    $y_i^{(k)} = 0$
    for $j \in B_{ik}$ **do**
      $y_i^{(k)} \leftarrow y_i^{(k)} + a_{ij}x_j$ in precision u_k
    end for
  end for
  $y_i = \sum_{k=1}^{q} y_i^{(k)}$ in precision u_1
end for

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

Theorem 1: Adaptive SpMV error bound

Given an accuracy target ϵ the buckets B_{ik} must be built as

$$B_{ik} = \{j \in \mathrm{nnz}_i(A) : |a_{ij}x_j| \in (\varepsilon\beta_i/u_{k+1}, \varepsilon\beta_i/u_k]\}$$

to get

$$|\widehat{y}_i - y_i| \le n_i\varepsilon\beta_i$$

with:

- $\beta_i = \sum_j |a_{ij}x_j|$ *for componentwise (CW) error:* $\forall i\ |\widehat{y}_i - y_i| \le O(\epsilon)\sum_j |a_{ij}x_j|$
- $\beta_i = \|A\|\|x\|$ *for normwise (NW) error:* $\|\widehat{y} - y\| \le O(\epsilon)\|A\|\|x\|$

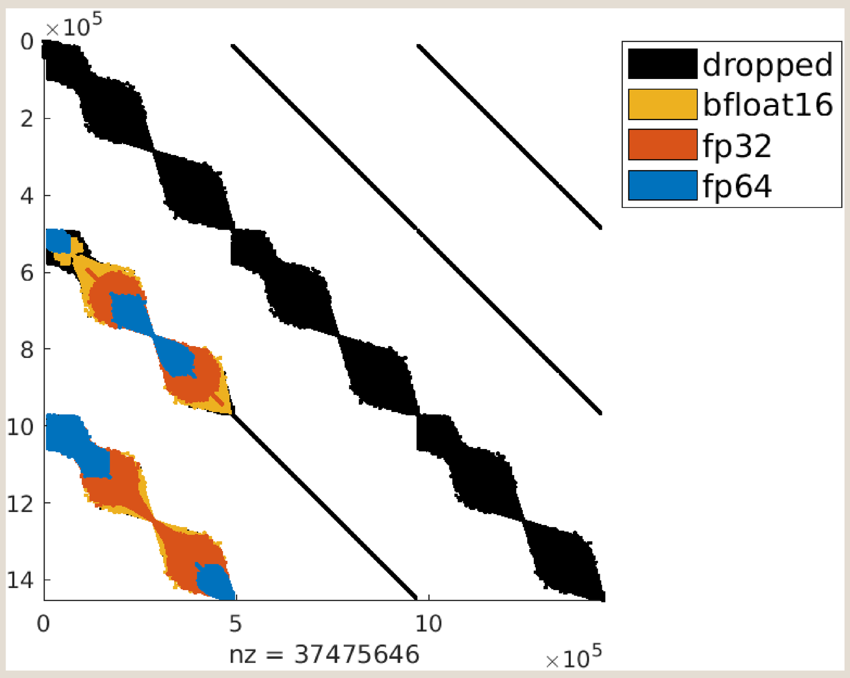
Experimental settings

- 34 matrices from SuiteSparse collection and industrial partners with at most 166M non-zeros.
- Machine: Intel Xeon E5-2690v3, 24 cores @2.60GHz, 193Go Memory
- Different **accuracy targets**
  - fp32
  - fp48
  - fp64
- Different **sets of precision formats**
  - 2 precisions: fp32, fp64
  - 3 precisions: bfloat16, fp32, fp64
  - 7 precisions: bfloat16, fp24, fp32, fp40, fp48, fp56, fp64

Emulated formats

	Bits			
Format	Signif. (t)	Exponent	Range	$u = 2^{-t}$
fp24	16	8	$10^{\pm 38}$	2×10^{-5}
fp40	29	11	$10^{\pm 308}$	2×10^{-9}
fp48	37	11	$10^{\pm 308}$	8×10^{-12}
fp56	45	11	$10^{\pm 308}$	3×10^{-14}

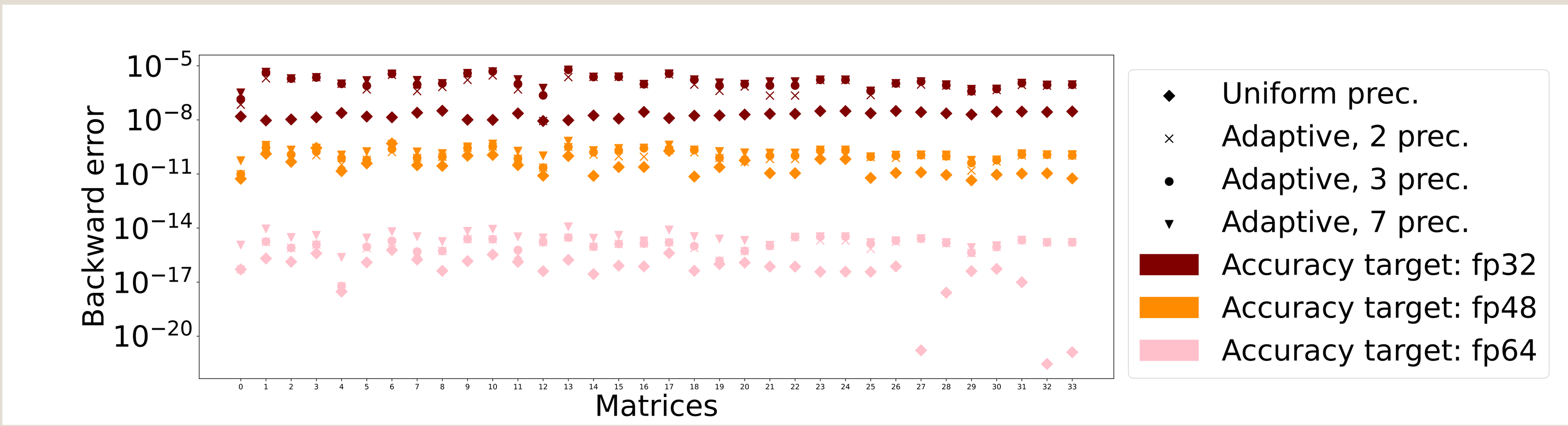
An example



Targeting an fp64 accuracy using precisions bfloat16, fp32 and fp64

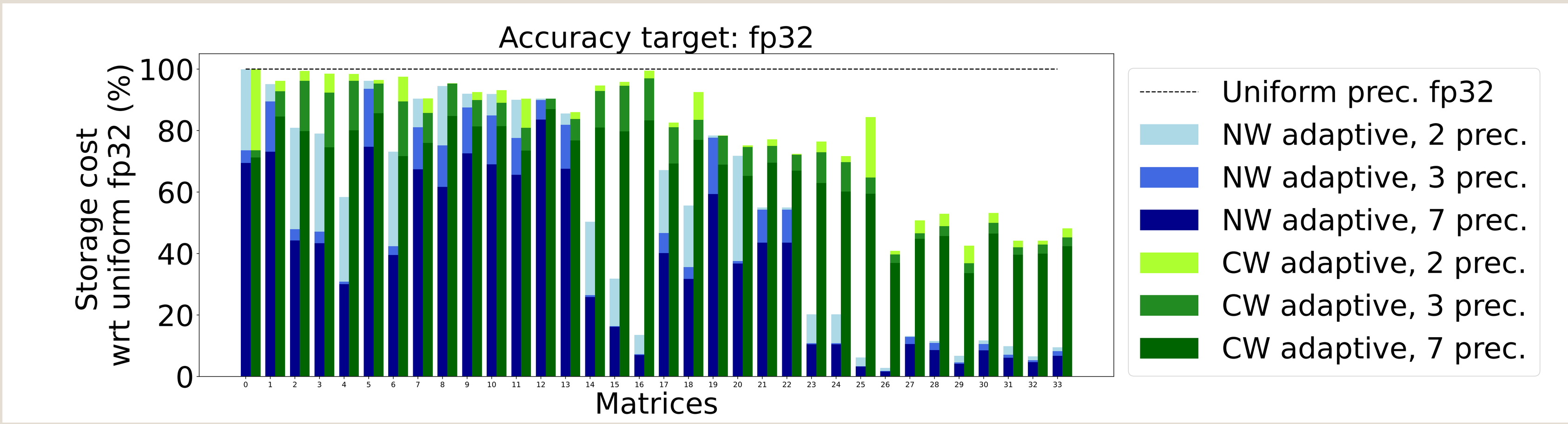
- 66% elements dropped
- 11% converted to bf16
- 12% converted to fp32
- only 11% stay in fp64

Measured error



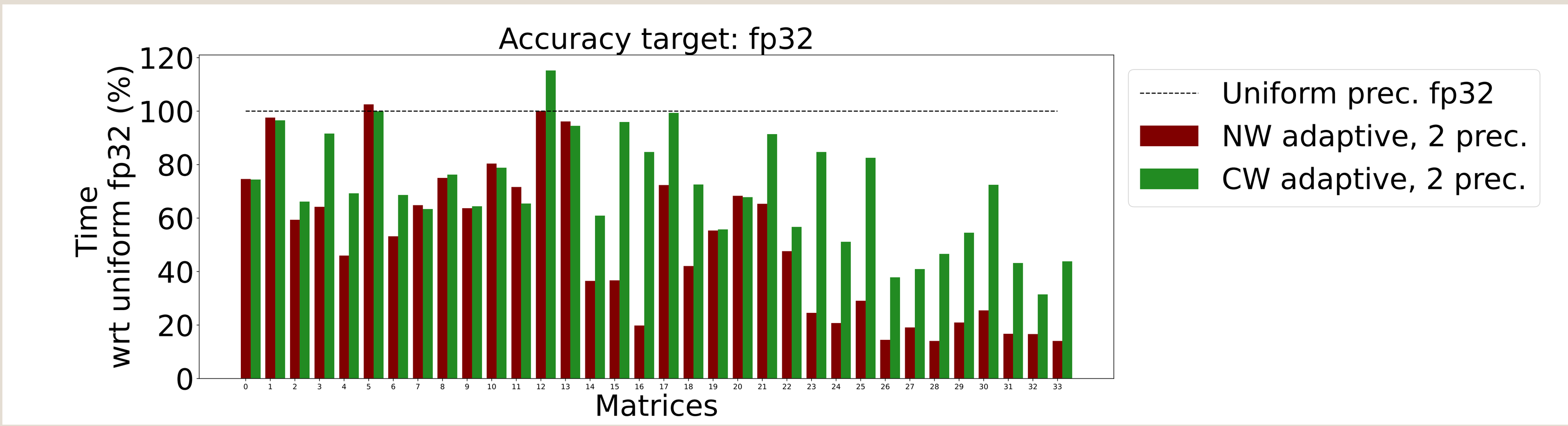
Adaptive methods achieve an **accuracy similar** to that of the uniform ones

Storage gains



Adaptive methods achieve **significant storage gains**, up to a factor 36×

Time gains



Storage gains translate into **time gains**, up to a factor 7×

Adaptive SpMV within GMRES

GMRES

$r = b - Ax_0$
$\beta = \|r\|_2$
$q_1 = r/\beta$
for $k = 1, 2, \dots$ **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

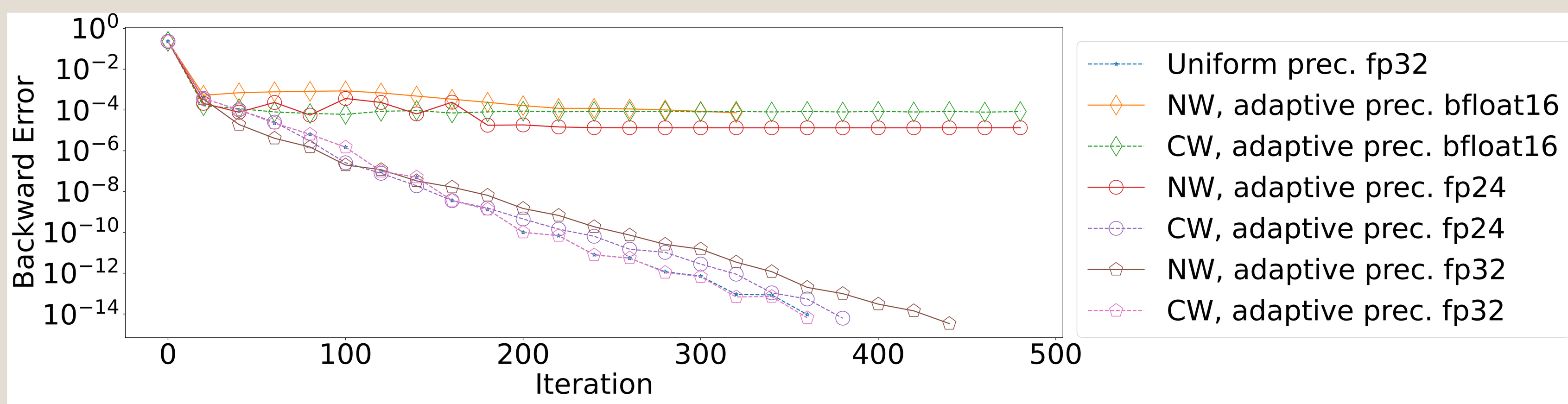
GMRES-based iterative refinement

for $i = 1, 2, \dots$ **do**
  $r_i = b - Ax_{i-1}$ in high precision
  Solve $Ad_i = r_i$ by GMRES in lower precision
  $x_i = x_{i-1} + d_i$
end for

The bottleneck of GMRES is SpMV

⇒ But **how does the adaptive method affect the convergence?**

Impact on GMRES convergence



For reasonable accuracy targets, adaptive SpMV **does not affect the convergence scheme**