



HAL
open science

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

Roméo Molina, Stef Graillat, Fabienne Jézéquel, Théo Mary

► **To cite this version:**

Roméo Molina, Stef Graillat, Fabienne Jézéquel, Théo Mary. Adaptive Precision Sparse Matrix-Vector Product and its Application to Krylov Solvers. Sparse Days Meeting 2022, Jun 2022, Saint-Girons, France. . hal-03780522

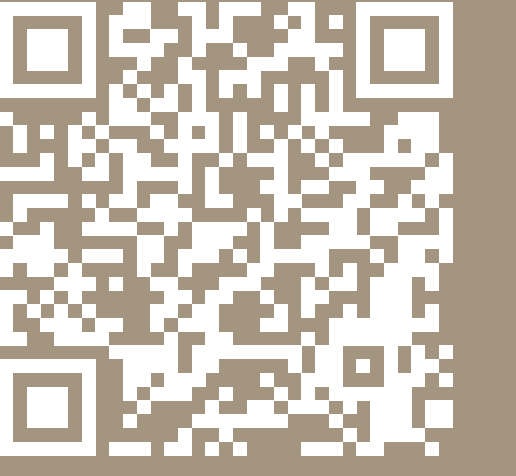
HAL Id: hal-03780522

<https://hal.science/hal-03780522v1>

Submitted on 19 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

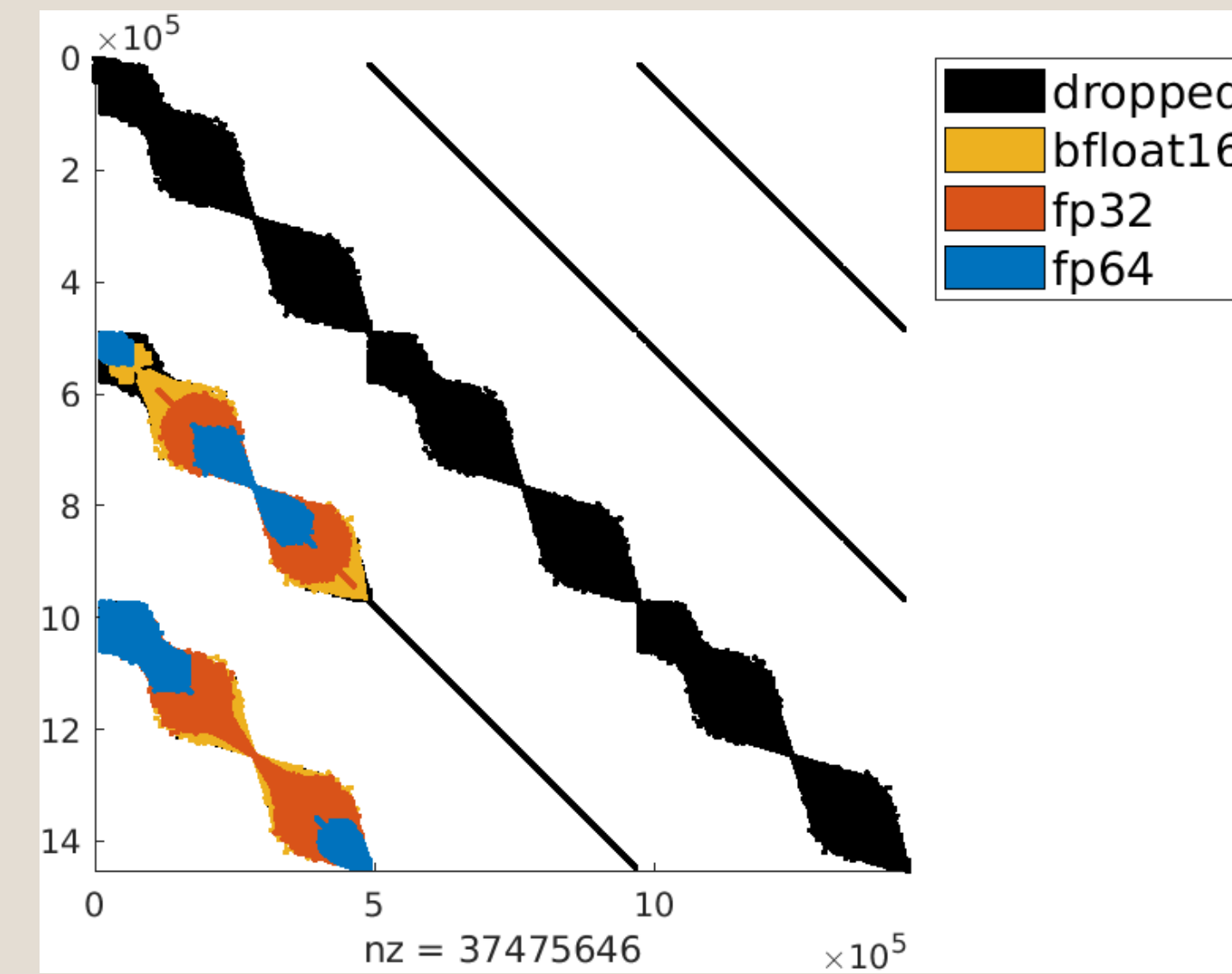


Today's floating-point landscape

	Bits		$u = 2^{-t}$	
	Signif. (t)	Exp. Range		
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

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

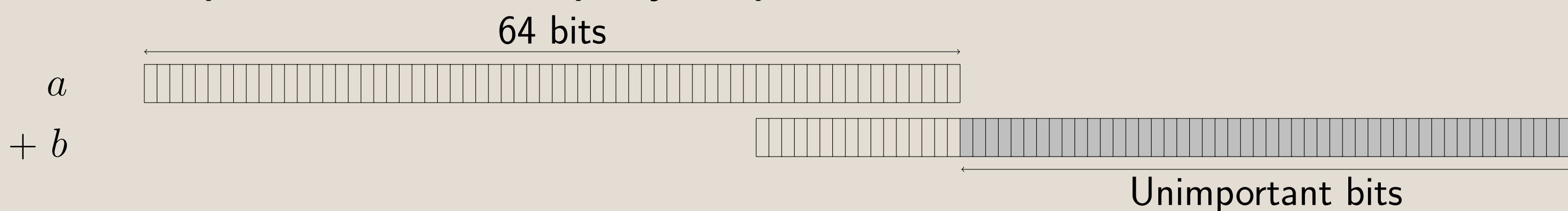
Mixed precision algorithms

Objectives:

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

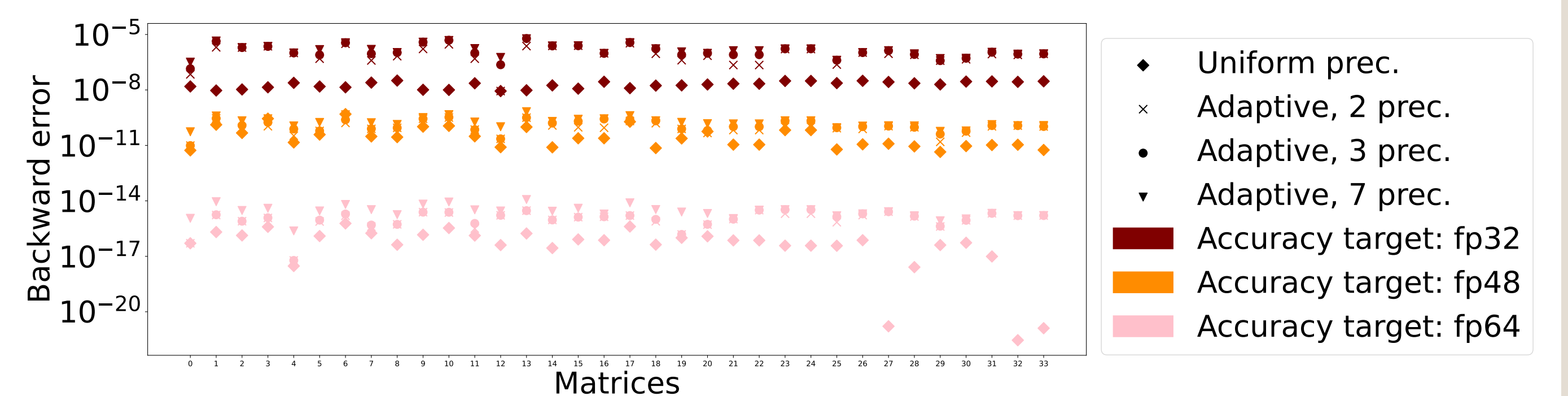
Opportunity for mixed precision:

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



⇒ We can **adapt the precisions to the data at hand**

Measured error



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

Uniform vs adaptive precision SpMV

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

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
```

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

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)} ← 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_q
end for
```

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

Theorem 1: Adaptive SpMV error bound

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

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

to get

$$|\hat{y}_i - y_i| \leq n_i \epsilon \beta_i$$

with:

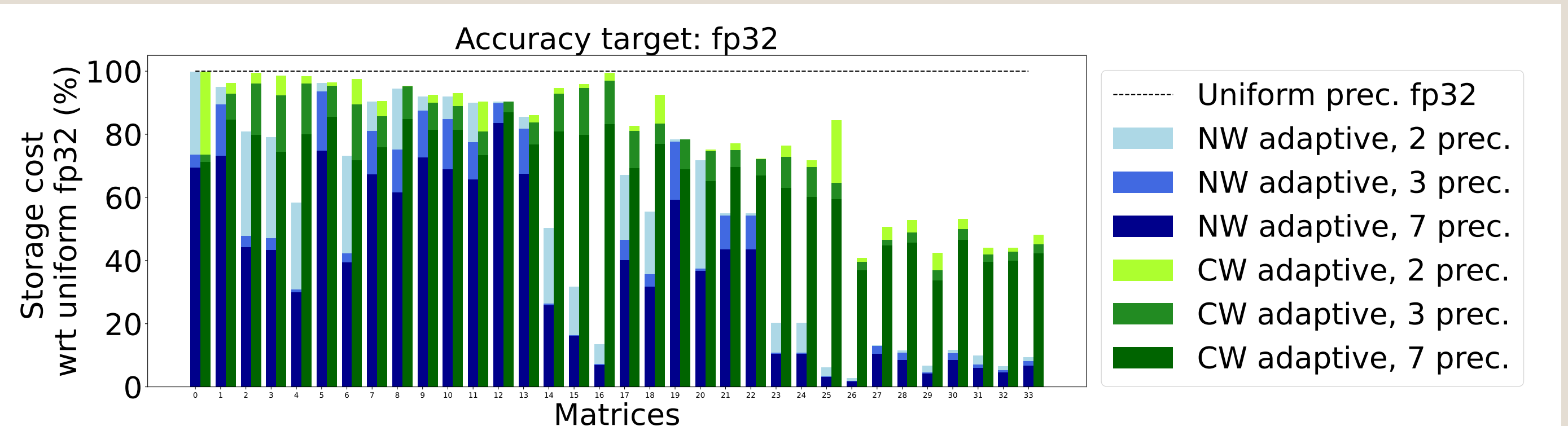
- $\beta_i = \sum_j |a_{ij} x_j|$ for **componentwise (CW) error**: $\forall i |\hat{y}_i - y_i| \leq O(\epsilon) \sum_j |a_{ij} x_j|$
- $\beta_i = \|A\| \|x\|$ for **normwise (NW) error**: $\|\hat{y} - y\| \leq O(\epsilon) \|A\| \|x\|$

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

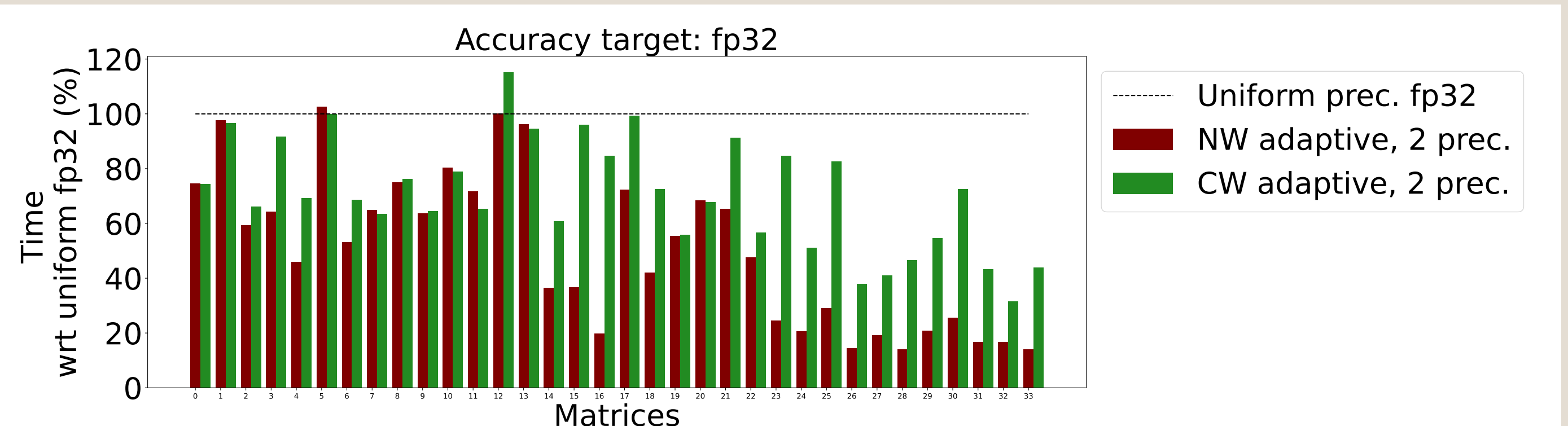
Emulated formats				
	Bits		$u = 2^{-t}$	
Format	Signif. (t)	Exponent Range		
fp24	16	8	$10^{\pm 38}$	2×10^{-5}
fp40	29	11	$10^{\pm 308}$	2×10^{-9}
fp48	37	11	$10^{\pm 308}$	8×10^{-12}
fp56	45	11	$10^{\pm 308}$	3×10^{-14}

Storage gains



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

Time gains



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

Adaptive SpMV within GMRES

GMRES

```
r = b - Ax_0
β = ||r||_2
q_1 = r / β
for k = 1, 2, ... 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 - βe_1||_2
  x_k = x_0 + Q_k c_k
end for
```

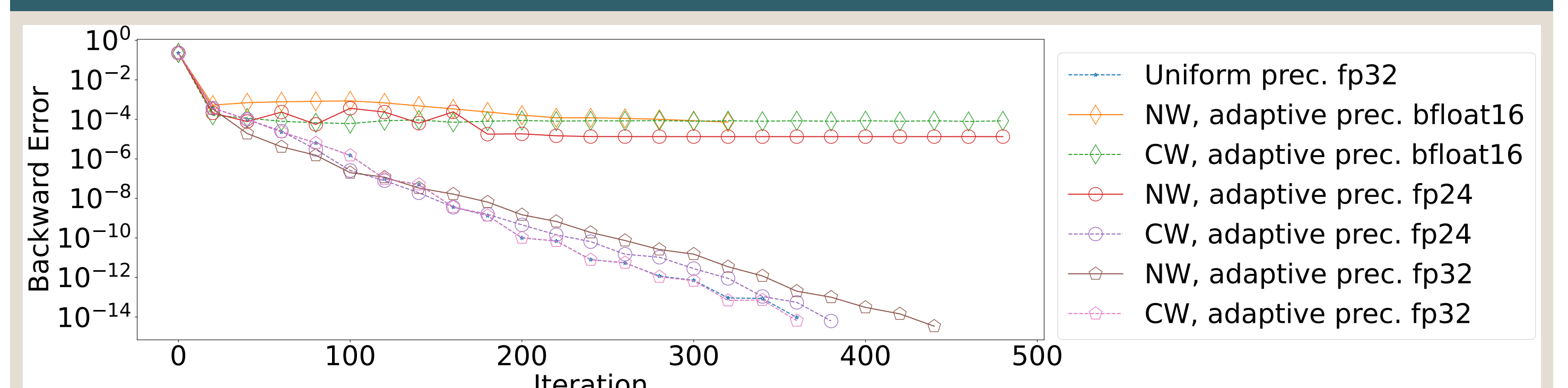
GMRES-based iterative refinement

```
for i = 1, 2, ... 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**