# 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. 13èmes Rencontres Arithmétique de l'Informatique Mathématique (RAIM 2022), Nov 2022, Nantes, France. hal-03931125

HAL Id: hal-03931125
https://hal.science/hal-03931125
Submitted on 9 Jan 2023

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.

# RAIM 2022 

3rd November 2022

## Adaptive Precision Sparse Matrix-Vector Product

 and its Application to Krylov SolversRoméo Molina<br>LIP6, Sorbonne Université

Service Online, Département Informatique, IJCLab
Joint work with
Stef Graillat, Fabienne Jézéquel, and Theo Mary

## Today's floating-point landscape

Bits

| bfloat16 | $B$ | 8 | 8 | $10^{ \pm 38}$ | $4 \times 10^{-3}$ |
| :--- | :--- | :---: | :---: | :--- | :--- |
| fp16 | $H$ | 11 | 5 | $10^{ \pm 5}$ | $5 \times 10^{-4}$ |
| fp32 | $S$ | 24 | 8 | $10^{ \pm 38}$ | $6 \times 10^{-8}$ |
| fp64 | $D$ | 53 | 11 | $10^{ \pm 308}$ | $1 \times 10^{-16}$ |
| fp128 | Q | 113 | 15 | $10^{ \pm 4932}$ | $1 \times 10^{-34}$ |

- Low precision increasingly supported by hardware
- Great benefits:
- Reduced storage, data movement, and communications
- Reduced energy consumption ( $5 \times$ with $\mathrm{fp} 16,9 \times$ with bfloat16)
- Increased speed on emerging hardware ( $16 \times$ 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

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 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, my thesis area: CADNA / PROMISE...
© Does not need any understanding of what the code does
$\boldsymbol{\nabla}$ Does not have any understanding of what the code does
- This work: another point of view, exploit as much as possible the knowledge we have about the code


## Adaptive precision algorithms

- Given an algorithm and a prescribed accuracy $\varepsilon$, adaptively select the minimal precision for each computation
$\Rightarrow$ Why does it make sense to make the precision vary?


## Adaptive precision algorithms

- Given an algorithm and a prescribed accuracy $\varepsilon$, adaptively select the minimal precision for each computation
$\Rightarrow$ Why does it make sense to make the precision vary?
- Because not all computations are equally "important"! Example:
a
$+b$

and small elements produce small errors :

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

$\Rightarrow$ 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 at the variable level?

- Pushing adaptive precision to the extreme: can we benefit from storing each variable in a (possibly) different precision?
- Example: $A x=b$ with adaptive precision for each $A_{i j}$
- Is it worth it?

Need to have elements of widely different magnitudes

- 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)

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

$$
\begin{aligned}
& \text { for } i=1: m \text { do } \\
& \quad y_{i}=0 \\
& \quad \text { for } j \in n n z_{i}(A) \text { do } \\
& \quad y_{i}=y_{i}+a_{i j} x_{j} \\
& \text { end for } \\
& \text { end for } \\
& \hline
\end{aligned}
$$

- Standard error analysis for $y=A x$ performed in a uniform precision $\varepsilon$ gives,

$$
\left|\widehat{y}_{i}-y_{i}\right| \leq n_{i} \varepsilon \sum_{j \in n n z_{i}(A)}\left|a_{i j} x_{j}\right|
$$

- 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 \(j \in n n z_{i}(A)\) do
            if \(a_{i j} x_{j} \in B_{i k}\) then
                \(y_{i}^{(k)}=y_{i}^{(k)}+a_{i j} x_{j}\) 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 $B_{i k}$ and sum elements of $B_{i k}$ in precision $u_{k}$
- Error analysis: $\left|\widehat{y}_{i}^{(k)}-y_{i}^{(k)}\right| \leq n_{i}^{(k)} u_{k} \sum_{a_{i j} x_{j} \in B_{i k}}\left|a_{i j} x_{j}\right|$
- $\left|\widehat{y}_{i}^{(k)}-y_{i}^{(k)}\right| \leq n_{i}^{(k)} u_{k} \sum_{a_{i j} x_{j} \in B_{i k}}\left|a_{i j} x_{j}\right|$
$\Rightarrow$ Build the buckets such that $u_{k} \sum_{a_{i j} x_{j} \in B_{i k}}\left|a_{i j} x_{j}\right| \approx \varepsilon \sum_{j}\left|a_{i j} x_{j}\right|$
- By setting $B_{i k}$ to the interval $\left(\varepsilon \beta_{i} / u_{k+1}, \varepsilon \beta_{i} / u_{k}\right]$, we obtain $\left|\widehat{y}_{i}^{(k)}-y_{i}^{(k)}\right| \leq n_{i}^{(k)} \varepsilon \beta_{i}$ and so $\left|\widehat{y}_{i}-y_{i}\right| \leq n_{i} \varepsilon \beta_{i}$
- Two possible choices for $\beta_{i}$ :
- $\beta_{i}=\sum_{j}\left|a_{i j} x_{j}\right| \Rightarrow$ guarantees $O(\varepsilon)$ componentwise error:

$$
\left|\widehat{y}_{i}-y_{i}\right| \leq n \epsilon \sum_{j}\left|a_{i j} x_{j}\right| \quad \forall i \in\{1, \ldots, n\}
$$

- $\beta_{i}=\|A\|\|x\| \Rightarrow$ guarantees $O(\varepsilon)$ normwise error:
$\left\|\widehat{y}_{i}-y_{i}\right\| \leq n \epsilon\|A\|\|x\|$
- Smallest elements are simply dropped


## Visualise mixed-precision gains

Matrice dgreen


For some matrices, many elements can be dropped that leads to major gains.

## Visualise mixed-precision gains



For some matrices, many elements can be dropped that leads to major gains.

## Visualise mixed-precision gains



For some matrices, many elements can be dropped that leads to major gains.

## Visualise mixed-precision gains

Matrice nv2


For some matrices, many elements can be dropped that leads to major gains.

## Visualise mixed-precision gains



For some matrices, many elements can be dropped that leads to major gains.

## SpMV experimental settings

- 33 matrices coming from SuiteSparse collection and industrial partners with at most 166 M non-zeros


## SpMV experimental settings

- 33 matrices coming from SuiteSparse collection and industrial partners with at most 166 M non-zeros
- 3 different accuracy targets:
- $\epsilon=2^{-24}$
- $\epsilon=2^{-37}$
- $\epsilon=2^{-53}$


## SpMV experimental settings

Various sets of precision formats:

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

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

## SpMV experiments

## Maintaining componentwise accuracy



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

## SpMV experiments

## Maintaining normwise accuracy



- Unif.

Adapt. 2 prec.
Adapt. 3 prec.
Adapt. 7 prec.
$\epsilon=2^{-24}$
$\epsilon=2^{-37}$
$\epsilon=2^{-53}$

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

## SpMV experiments

Theoretical storage gains targetting $\epsilon=2^{-53}$ accuracy (fp64)


Up to $\mathbf{8 8 \%}$ of storage reduction

## SpMV experiments

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


Up to $\mathbf{8 5 \%}$ of time reduction

## SpMV experiments

Theoretical storage gains targetting $\epsilon=2^{-24}$ accuracy (fp32)


Up to $\mathbf{9 7 \%}$ of storage reduction

## SpMV experiments

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

$\square$ Unif. fp32

- Adapt. NW 2 prec.
$\square$ Adapt. CW 2 prec.

Up to $\mathbf{8 8 \%}$ of time reduction

## SpMV experiments

Theoretical storage gains targetting $\epsilon=2^{-37}$ accuracy (unavailable in hardware!)

Accuracy target: $\epsilon=2^{-37}$


We are able to target any kind of accuracy with only natively supported precisions.

## SpMV experiments

Actual time gains targetting $\epsilon=2^{-37}$ accuracy (unavailable in hardware!)


We are able to target any kind of accuracy with only natively supported precisions.

## Applying adaptive GMRES to iterative solvers

## Performance of GMRES rely on SpMV

```
\(r=b-A x_{0}\)
\(\beta=\|r\|_{2}\)
\(q_{1}=r / \beta\)
for \(k=1,2, \ldots\) do
    \(y=A q_{k}\)
    for \(j=1: k\) do
        \(h_{j k}=q_{j}^{T} y\)
        \(y=y-h_{j k} q_{j}\)
    end for
    \(h_{k+1, k}=\|y\|_{2}\)
    \(q_{k+1}=y / h_{k+1, k}\)
    Solve the least squares problem \(\min _{c_{k}}\left\|H c_{k}-\beta e_{1}\right\|_{2}\)
    \(x_{k}=x_{0}+Q_{k} c_{k}\)
end for
```

How does the adaptive method affect the convergence?

- Adaptive SpMV achieve larger speedups for lower accuracy targets
- GMRES-based iterative refinement particularly attractive

```
1: for i=1,2,\ldots do
2:
3: Solve Ad
4: }\quad\mp@subsup{x}{i}{}=\mp@subsup{x}{i-1}{}+\mp@subsup{d}{i}{
5: end for
```

- SpMV line 2 in CW adaptive, target $\epsilon=2^{-53}$
- GMRES line 3 experimented with multiple variants
- Use of simple Jacobi preconditioner
- Fixed outside threshold $\epsilon_{\text {out }}=2^{-53}$
- Varying inside threshold $\epsilon_{i n}=2^{-24}$
- Fixed inside restart to 80 iterations


## Application to GMRES: convergence scheme experiments

GMRES convergence for matrix ML_Laplace


- With reasonable accuracy targets, adaptive SpMV, does not affect the confergence scheme
- Choose the best compromise between the iteration cost and the number of iterations


## Application to GMRES: convergence scheme experiments

GMRES convergence for matrix CoupCons3D


- Unif. fp32
. Unif. bfloat16 (50\%)
$\cdots \quad$ Adapt. CW $\epsilon_{i n}=2^{-24}(68 \%)$
Adapt. CW $\epsilon_{\text {in }}=2^{-20}(56 \%)$
$\longrightarrow$ Adapt. CW $\epsilon_{i n}=2^{-16}(46 \%)$
. Adapt. NW $\epsilon_{i n}=2^{-24}(36 \%)$
$\rightarrow$ Adapt. NW $\epsilon_{i n}=2^{-16}(1 \%)$
Adapt. NW $\epsilon_{i n}=2^{-20}(17 \%)$
- For a given $\epsilon_{i n}$ value, NW variants achieve a lower cost but a slower convergence than CW ones
- $\epsilon_{i n}=2^{-24}$ leads to the best NW variant, which converges in 1040 iterations with an SpMV cost of $36 \%$ of the fp32 uniform one
- $\epsilon_{\text {in }}=2^{-20}$ leads to the best CW variant, which converges in 320 iterations with a corresponding SpMV cost of $56 \%$
18/1 Both options should be considered.


## Application to GMRES: convergence scheme experiments



- Surprising behavior
- Consistently reproduced and occurs for several other matrices in our set
- Aggressive dropping of small coefficients might lead to a "nicer" matrix for which GMRES can converge quickly.


## Conclusion: take-home messages

- Adaptive precision SpMV algorithm
- Ruilds computing buckets according to the elements magnitude
- Targets any accuracy
- Matrix-dependent gains
- Application to Krylov solvers
- Reasonable accuracy targets preserves convergence scheme

More info


Thank you! Any questions?

