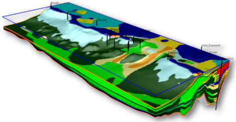
Mixed Precision Strategies for Solving Sparse Linear Systems with BiCGStab

Hugo Dorfsman (LIP6/IFPEN)
Ani Anciaux-Sedrakian Thomas Guignon Fabienne Jézéquel Théo Mary

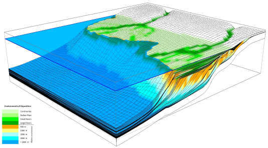
CANUM 2024 - 46ème Congrès National d'Analyse Numérique

31st May, 2024

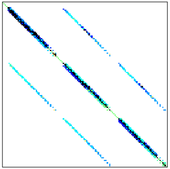
Iterative solvers for sparse linear systems



(a) Stratigraphy



(b) Sedimentary deposition



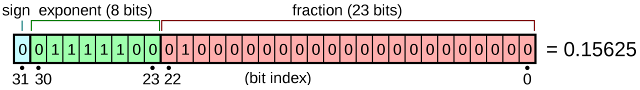
(c) Sparse matrix

- Many applications lead to require solving large ill-conditioned sparse systems
- We rely on iterative solvers for performance, which require a good preconditioner
- Scalability of iterative solvers in HPC is limited by communication costs
- Growing support and availability of lower precision hardware (especially GPUs)

Floating point precision

Representing floating point numbers:

- High precision numbers
  - better range and accuracy (up to 1×10^{-16} for double 64bit)
  - slow operations and expensive to transfer
- Low precision numbers
  - smaller range and accuracy (only up to 6×10^{-8} for single 32bit)
  - faster operations and memory transfers



Larger errors are introduced at each operation but can be mitigated by mixing the different precisions in an suitable way [1,2]

[1] A. Abdelfattah et al. (2020), DOI: 10.48550/ARXIV.2007.06674

[2] N. J. Higham and T. Mary (2022), *Acta Numer.*, DOI: 10/gscz6w

BiCGStab

Algorithms for solving linear systems with square non symmetric matrices:

- BiCGStab or GMRES both iterative Krylov subspace methods

Characteristics of BiCGStab:

👍 only needs a fixed amount of memory

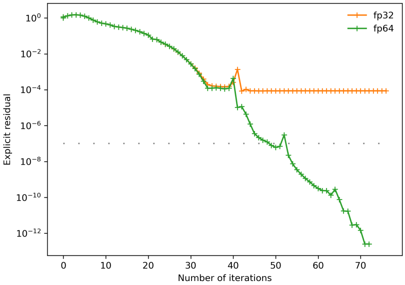
👎 convergence is not monotone

👎 no guarantee of convergence even in exact arithmetic

$x_0, \bar{r}_0$ arbitrary
$p_0 = r_0 \leftarrow b - Ax_0$
for $(i = 1$ **to** $i_{\max})$
  Solve: $M\hat{p} = p_{i-1}$
  $\alpha = (\bar{r}_0, r_{i-1})/(\bar{r}_0, A\hat{p})$
  $s = r_{i-1} - \alpha A\hat{p}$
  Solve: $M\hat{s} = s$
  $\omega = (A\hat{s}, s)/(A\hat{s}, A\hat{s})$
  $x_i = x_{i-1} + \alpha\hat{p} + \omega\hat{s}$
  $r_i = s - \omega A\hat{s}$
  if $(\|r_i\| < \epsilon)$ **then** exit loop
  $\beta = \alpha(\bar{r}_0, r_i)/\omega(\bar{r}_0, r_{i-1})$
  $p_i = r_i + \beta(p_{i-1} - \omega A\hat{p})$

Single precision preconditioner and Flexible BiCGStab

- Single precision degrades attainable precision and sometimes convergence speed



Matrix: SparseSuite HB/sherman4

Preconditioner: BlockJacobi 20x20

$$\text{ExpRes} = \frac{\|Ax - b\|}{\|b\|}$$

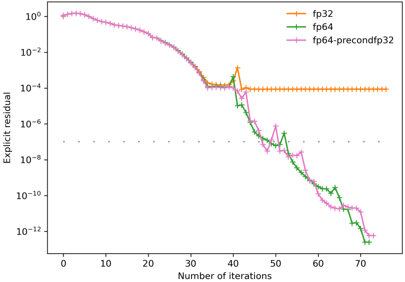
[3] J. Chen et al. (2016), *J. Sci. Comput.*, DOI: 10/gs775m

Single precision preconditioner and Flexible BiCGStab

- Single precision degrades attainable precision and sometimes convergence speed

Some possible approaches:

- Compute the preconditioner in fp32
- Apply the preconditioner in fp32 $\Longrightarrow$ variable preconditioner: but BiCGStab is a flexible solver [3]



Matrix: SparseSuite HB/sherman4

Preconditioner: BlockJacobi 20x20

$$\text{ExpRes} = \frac{||Ax - b||}{||b||}$$

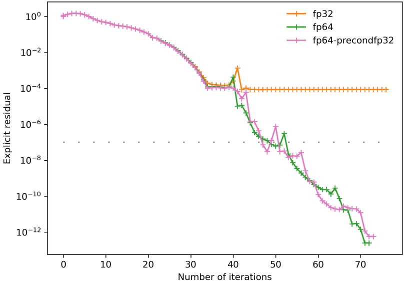
[3] J. Chen et al. (2016), *J. Sci. Comput.*, DOI: 10/gs775m

Single precision preconditioner and Flexible BiCGStab

- Single precision degrades attainable precision and sometimes convergence speed

Some possible approaches:

- Compute the preconditioner in fp32
- Apply the preconditioner in fp32 $\Longrightarrow$ variable preconditioner: but BiCGStab is a flexible solver [3]



Matrix: SparseSuite HB/sherman4

Preconditioner: BlockJacobi 20x20

$$\text{ExpRes} = \frac{||Ax - b||}{||b||}$$

SpMV and dot products remain in fp64, can we improve performance again?

[3] J. Chen et al. (2016), *J. Sci. Comput.*, DOI: 10/gs775m

Iterative Refinement (IR)

- At each iteration we correct x in higher precision.
- Historically used with GMRES-IR (equivalent to restarted GMRES) and shown to work well in mixed precision [4]
- Restarted BiCGStab is less common.

y_0 and $\bar{r}_0$ arbitrary, $x_0 = 0$
for ($j = 1$ **to** $j_{\max}$)
  $R \leftarrow b - Ay_{j-1}$
  $p_0 = r_0 \leftarrow R - Ax_0$
  for ($i = 1$ **to** $i_{\max}$)
    Solve: $M\hat{p} = p_{i-1}$
    $\alpha = (\bar{r}_0, r_{i-1})/(\bar{r}_0, A\hat{p})$
    $s = r_{i-1} - \alpha A\hat{p}$
    Solve: $M\hat{s} = s$
    $\omega = (A\hat{s}, s)/(A\hat{s}, A\hat{s})$
    $x_i = x_{i-1} + \alpha\hat{p} + \omega\hat{s}$
    $r_i = s - \omega A\hat{s}$
    if ($\|r_i\| < \epsilon$) **then** exit loop
    $\beta = \alpha(\bar{r}_0, r_i)/\omega(\bar{r}_0, r_{i-1})$
    $p_i = r_i + \beta(p_{i-1} - \omega A\hat{p})$
  $y_j \leftarrow y_{j-1} + x_i$
return y_j

[4] E. Carson and N. J. Higham (2018), *SIAM J. Sci. Comput.*, DOI: 10/gs77zr

Iterative Refinement (IR)

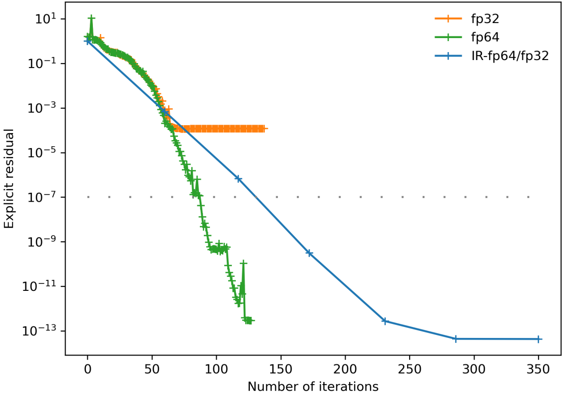
- At each iteration we correct x in higher precision.
- Historically used with GMRES-IR (equivalent to restarted GMRES) and shown to work well in mixed precision [4]
- Restarted BiCGStab is less common.

→ BiCGStab-IR in mixed precision

y_0 and $\bar{r}_0$ arbitrary, $x_0 = 0$
for ($j = 1$ **to** $j_{\max}$)
  $R \leftarrow b - Ay_{j-1}$ (Prec u_{high})
  $p_0 = r_0 \leftarrow R - Ax_0$
  for ($i = 1$ **to** $i_{\max}$)
    Solve: $M\hat{p} = p_{i-1}$
    $\alpha = (\bar{r}_0, r_{i-1})/(\bar{r}_0, A\hat{p})$
    $s = r_{i-1} - \alpha A\hat{p}$
    Solve: $M\hat{s} = s$
    $\omega = (A\hat{s}, s)/(A\hat{s}, A\hat{s})$ (Prec u_{low})
    $x_i = x_{i-1} + \alpha\hat{p} + \omega\hat{s}$
    $r_i = s - \omega A\hat{s}$
    if ($\|r_i\| < \epsilon$) **then** exit loop
    $\beta = \alpha(\bar{r}_0, r_i)/\omega(\bar{r}_0, r_{i-1})$
    $p_i = r_i + \beta(p_{i-1} - \omega A\hat{p})$
  $y_j \leftarrow y_{j-1} + x_i$ (Prec u_{high})
return y_j

[4] E. Carson and N. J. Higham (2018), *SIAM J. Sci. Comput.*, DOI: 10/gs77zr

Iterative Refinement (IR)



Matrix: SparseSuite HB/sherman4
Preconditioner: None

$$\text{ExpRes} = \frac{\|Ax - b\|}{\|b\|}$$

IR converges to double precision accuracy but degrades convergence speed.
Similar issues observed in recent work [5]

[5] Y. Zhao et al. (2023), *J. Inf. Process.*, DOI: 10.2197/ipsjjip.31.860

Flying Restart (FR)

FR is a restarted version of BiCGStab described in [6] that keeps a memory of previously calculated direction vectors.
Design objectives of FR :

- enhance attainable accuracy in fixed precision
- allow for a better stopping criteria

x_0 and $\bar{r}_0$ arbitrary, $y = x_0$
$p_0 = r_0 \leftarrow b - Ax_0$
for $(i = 1$ **to** $i_{\max})$
  Solve: $M\hat{p} = p_{i-1}$
  $\alpha = (\bar{r}_0, r_{i-1})/(\bar{r}_0, A\hat{p})$
  $s = r_{i-1} - \alpha A\hat{p}$
  Solve: $M\hat{s} = s$
  $\omega = (A\hat{s}, s)/(A\hat{s}, A\hat{s})$
  $x_i = x_{i-1} + \alpha\hat{p} + \omega\hat{s}$
  $r_i = s - \omega A\hat{s}$
  if (*flying restart*) **then**
    $r_i = b - Ax_i$
    $y = y + x_i$
    $x_i = 0; \quad b = r_i$
  if $(\|r_i\| < \epsilon)$ **then** exit loop
  $\beta = \alpha(\bar{r}_0, r_i)/\omega(\bar{r}_0, r_{i-1})$
  $p_i = r_i + \beta(p_{i-1} - \omega A\hat{p})$
return $y + x_i$

[6] G. L. G. Sleijpen and H. A. van der Vorst (1996), *Computing*, DOI: 10.1007/bf02309342

Flying Restart (FR)

FR is a restarted version of BiCGStab described in [6] that keeps a memory of previously calculated direction vectors.

Design objectives of FR :

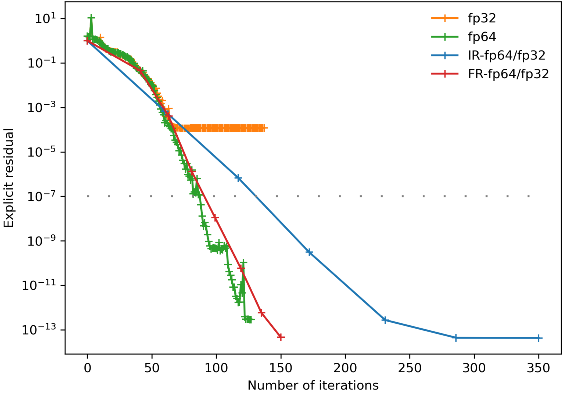
- enhance attainable accuracy in fixed precision
- allow for a better stopping criteria

Can we use FR to improve mixed precision BiCGStab?

x_0 and $\bar{r}_0$ arbitrary, $y = x_0$
$p_0 = r_0 \leftarrow b - Ax_0$
for $(i = 1$ **to** $i_{\max})$
  Solve: $M\hat{p} = p_{i-1}$
  $\alpha = (\bar{r}_0, r_{i-1})/(\bar{r}_0, A\hat{p})$
  $s = r_{i-1} - \alpha A\hat{p}$
  Solve: $M\hat{s} = s$ (Prec u_{low})
  $\omega = (A\hat{s}, s)/(A\hat{s}, A\hat{s})$
  $x_i = x_{i-1} + \alpha\hat{p} + \omega\hat{s}$
  $r_i = s - \omega A\hat{s}$
  if (*flying restart*) **then**
    $r_i = b - Ax_i$
    $y = y + x_i$ (Prec u_{high})
    $x_i = 0; \quad b = r_i$
  if $(\|r_i\| < \epsilon)$ **then** exit loop
  $\beta = \alpha(\bar{r}_0, r_i)/\omega(\bar{r}_0, r_{i-1})$ (Prec u_{low})
  $p_i = r_i + \beta(p_{i-1} - \omega A\hat{p})$
return $y + x_i$

[6] G. L. G. Sleijpen and H. A. van der Vorst (1996), *Computing*, DOI: 10.1007/bf02309342

Flying Restart (FR)



Matrix: SparseSuite HB/sherman4
Preconditioner: None

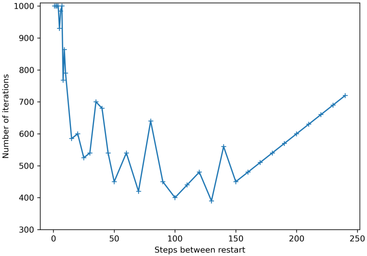
$$\text{ExpRes} = \frac{\|Ax-b\|}{\|b\|}$$

FR converges to double precision and is closer to fp64 convergence speed

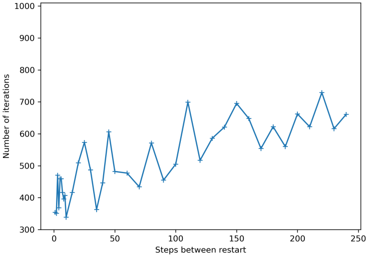
Choice of parameters

The speed of convergence depends on the choice of parameters for the restart:

$$\textbf{if} \quad \underbrace{(\text{res} < \epsilon)}_{\text{restart tolerance}} \quad \text{or} \quad \underbrace{(n_r > \text{maxiter})}_{\text{steps since last restart}} \quad \textbf{then} \text{ restart}$$



(a) IR



(b) FR

Matrix: SparseSuite Wang/wang3 Preconditioner: None ExpRes $= \frac{||Ax-b||}{||b||}$

An ϵ too small or too big can degrade the performance for IR
The parameter choice for FR tends to be more linear

Experimental setting

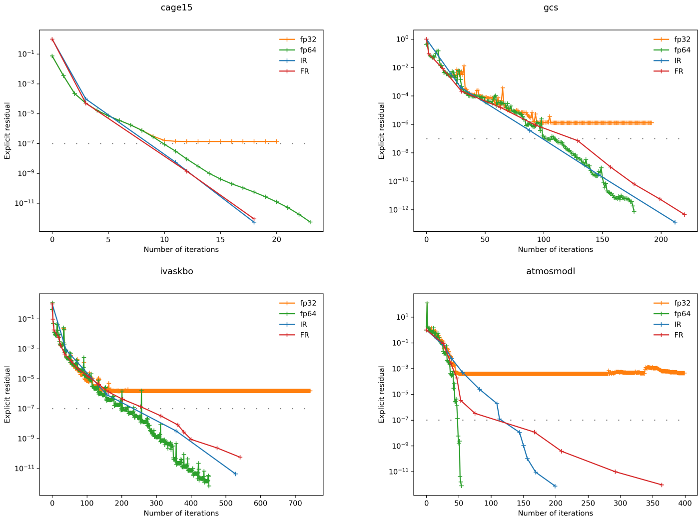
Matrices:

Origin	Matrix	n	# nonzeros	Block size
SuiteSparse	vanHeukelum/cage15	5 154 859	99 199 551	1x1
SuiteSparse	Wang/wang3	26 064	177 168	1x1
IFPEN	IvaskBO-N1-I1-F1	49 572	472 927	3x3
IFPEN	GCS-400p-N1-I1-F2	185 498	1 094 385	3x3
SuiteSparse	Bourchtein/atmosmodl	1 489 752	10 319 760	1x1
SuiteSparse	HB/sherman4	1104	3786	1x1

Parameters: Tuned reasonably with $\epsilon \in \left[10^{-1}, 10^{-7}\right]$
Software: MCGSolver
Clusters:

- IFPEN Ener: 2 CPU Intel Skylake G-6140 - 2.3 GHz, 96 GB Memory
- TGCC Topaze: 2 CPU AMD EPYC Milan 7763 - 2.45 GHz, 4 GPU Nvidia A100, 256 GB Memory
- TGCC Irene: 1 CPU Fujitsu A64FX - 1.8 GHz, 32 GB Memory

Convergence speed comparison



Preconditioner: None ExpRes $= \frac{||Ax-b||}{||b||}$ Kernel: CPUAVX2 Cluster: IFPEN Ener

Sparse Matrix Vector Products Flops Benchmarks

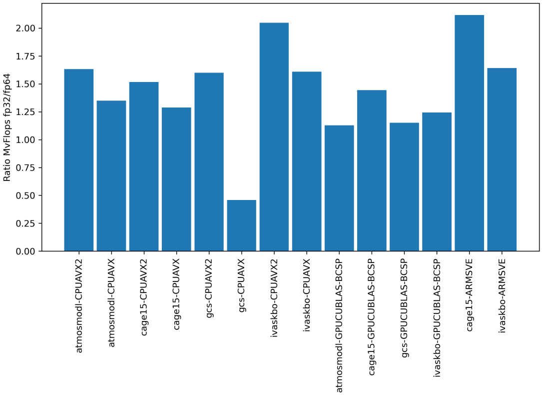
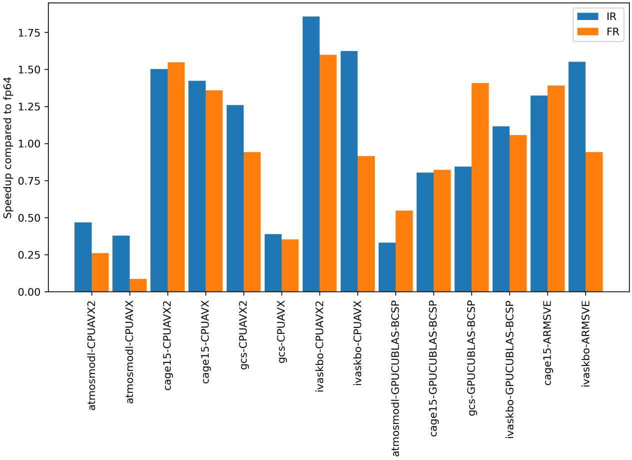


Figure: Ratio of fp32 MvFlops compared to fp64 on different kernels and matrices

Speed of matrix vector products depends on:

- floating point precision: fp32 / fp64
- but also matrices
- and computation kernel, computer architecture, compiler versions...

Performance comparison



Preconditioner: None

Conclusion

- BiCGStab fp32: potential to gain speed but accuracy is bad
- Iterative Refinement (IR):
  - 👍 can take advantage of fp32 speed
  - 👍 converges to fp64 accuracy
  - 👎 restarts tends to hinder convergence speed
- Flying Restart (FR):
  - 👍 can take advantage of fp32 speed
  - 👍 converges to fp64 accuracy
  - 👍 convergence speed usually preserved

Perspectives

- compare the performance when using a preconditioner
- 16bit precision on GPUs