Improving the Numerical Behavior of Communication-Avoiding Krylov Subspace Methods

Erin Carson
Charles University

September 1, 2022
Faculty of Computer Science, University of Vienna

The Conjugate Gradient (CG) Method

$r_0 = b - Ax_0, \quad p_0 = r_0$

for $i = 1$:nmax

$$\alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T A p_{i-1}}$$

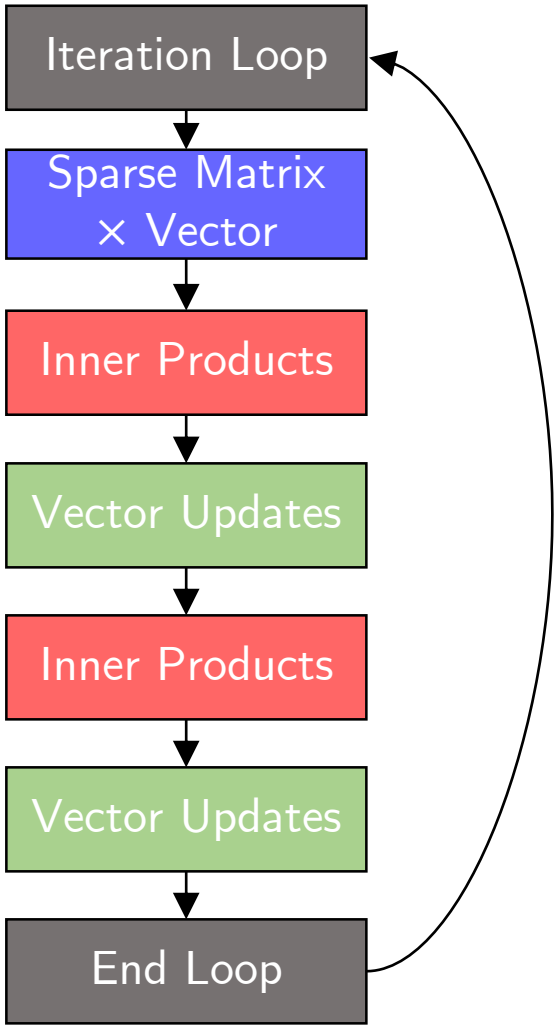
$$x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$$

$$r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$$

$$\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$$

$$p_i = r_i + \beta_i p_{i-1}$$

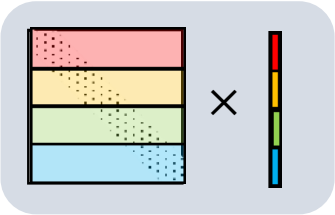
end



Cost Per Iteration

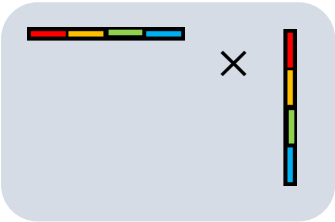
→ Sparse matrix-vector multiplication (SpMV)

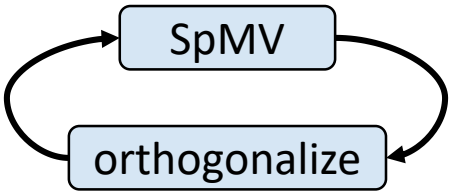
- $O(\text{nnz})$ flops
- Must communicate vector entries w/neighboring processors (nearest neighbor MPI collective)
- Must read A/vector from slow memory



→ Inner products

- $O(N)$ flops
- global synchronization (MPI_Allreduce)
  - all processors must exchange data and wait for *all* communication to finish before proceeding
- Multiple reads/writes to slow memory





Low computation/communication ratio

⇒ **Performance is communication-bound**

Synchronization-reducing variants

Motivated many approaches to reducing synchronization (increasing ratio of computation to communication) in CG:

- Early work: CG with a single synchronization point per iteration
  - 3-term recurrence CG
  - Using modified computation of recurrence coefficients
  - Using auxiliary vectors
- Pipelined Krylov subspace methods
  - Uses modified coefficients and auxiliary vectors to reduce synchronization points to 1 per iteration
  - Modifications also allow decoupling of SpMV and inner products - enables overlapping (MPI non-blocking collectives)
- s-step Krylov subspace methods
  - Compute iterations in blocks of s using a different Krylov subspace basis
  - Enables one synchronization per s iterations

s-step CG

Key observation: After iteration i, for $j \in \{0, .., s\}$,

$$x_{i+j} - x_i, \quad r_{i+j}, \quad p_{i+j} \quad \in \quad \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$$

s-step CG

Key observation: After iteration i, for $j \in \{0, .., s\}$,

$$x_{i+j} - x_i, \quad r_{i+j}, \quad p_{i+j} \quad \in \quad \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$$

s steps of s-step CG:

s-step CG

Key observation: After iteration i, for $j \in \{0, .., s\}$,

$$x_{i+j} - x_i, \quad r_{i+j}, \quad p_{i+j} \quad \in \quad \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$$

s steps of s-step CG:

Expand solution space s dimensions at once

Compute "basis" matrix $\mathcal{Y}$ such that $\operatorname{span}(\mathcal{Y}) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$ according to the recurrence $A\underline{\mathcal{Y}} = \mathcal{Y}\,\mathcal{B}$

s-step CG

Key observation: After iteration i, for $j \in \{0, .., s\}$,

$$x_{i+j} - x_i, \quad r_{i+j}, \quad p_{i+j} \quad \in \quad \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$$

s steps of s-step CG:

Expand solution space s dimensions at once

Compute "basis" matrix $\mathcal{Y}$ such that $\operatorname{span}(\mathcal{Y}) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$ according to the recurrence $A\underline{\mathcal{Y}} = \mathcal{Y}\,\mathcal{B}$

Compute inner products between basis vectors in one synchronization

$$\mathcal{G} = \mathcal{Y}^T \mathcal{Y}$$

s-step CG

Key observation: After iteration i, for $j \in \{0, .., s\}$,

$$x_{i+j} - x_i, \quad r_{i+j}, \quad p_{i+j} \quad \in \quad \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$$

s steps of s-step CG:

Expand solution space s dimensions at once

Compute "basis" matrix $\mathcal{Y}$ such that $\operatorname{span}(\mathcal{Y}) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$ according to the recurrence $A\underline{\mathcal{Y}} = \mathcal{Y}\,\mathcal{B}$

Compute inner products between basis vectors in one synchronization

$$\mathcal{G} = \mathcal{Y}^T \mathcal{Y}$$

Compute s iterations of vector updates

Perform s iterations of vector updates by updating coordinates in basis $\mathcal{Y}$:

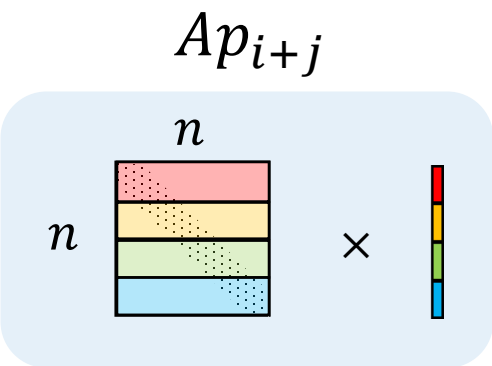
$$x_{i+j} - x_i = \mathcal{Y} x'_j, \qquad r_{i+j} = \mathcal{Y} r'_j, \qquad p_{i+j} = \mathcal{Y} p'_j$$

s-step CG

For s iterations of updates, inner products and SpMVs (in basis $\mathcal{Y}$) can be computed by independently by each processor without communication:

s-step CG

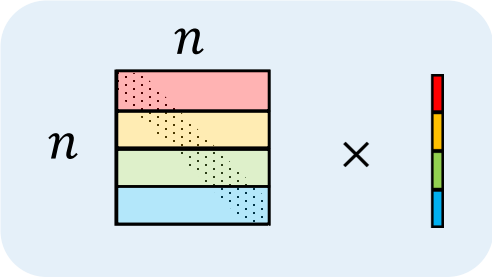
For s iterations of updates, inner products and SpMVs (in basis $\mathcal{Y}$) can be computed by independently by each processor without communication:



s-step CG

For s iterations of updates, inner products and SpMVs (in basis $\mathcal{Y}$) can be computed by independently by each processor without communication:

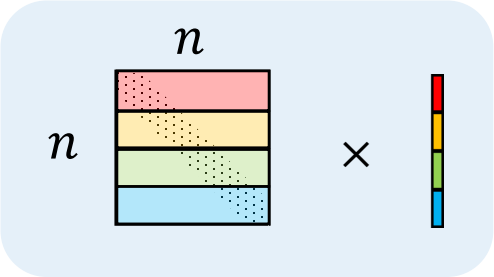
$$Ap_{i+j} \quad = \quad A\underline{\mathcal{Y}}p'_j$$



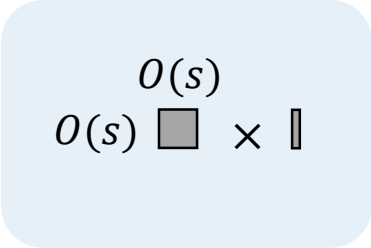
s-step CG

For s iterations of updates, inner products and SpMVs (in basis $\mathcal{Y}$) can be computed by independently by each processor without communication:

$$Ap_{i+j} \quad = \quad A\underline{\mathcal{Y}}p'_j \quad = \quad \mathcal{Y}(\mathcal{B}p'_j)$$



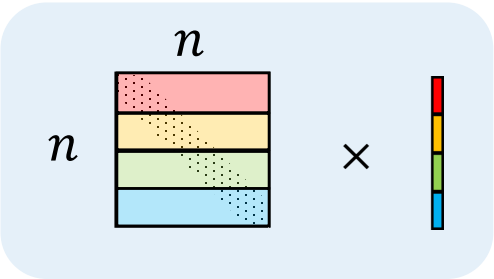
$\rightarrow$



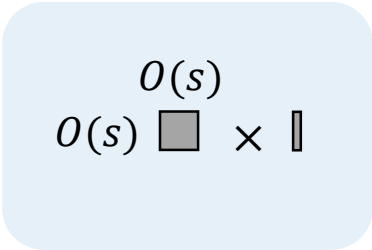
s-step CG

For s iterations of updates, inner products and SpMVs (in basis $\mathcal{Y}$) can be computed by independently by each processor without communication:

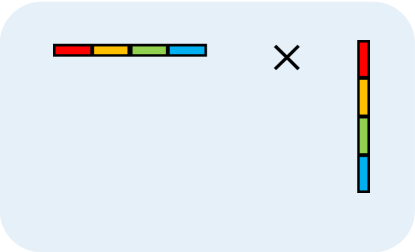
$$Ap_{i+j} \quad = \quad A\underline{\mathcal{Y}}p'_j \quad = \quad \mathcal{Y}(\mathcal{B}p'_j)$$



$\rightarrow$



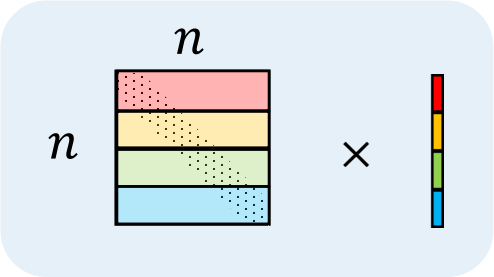
$$(r_{i+j}, r_{i+j})$$



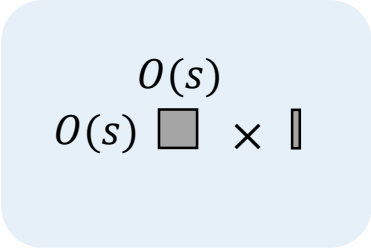
s-step CG

For s iterations of updates, inner products and SpMVs (in basis $\mathcal{Y}$) can be computed by independently by each processor without communication:

$$Ap_{i+j} = A\underline{\mathcal{Y}}p'_j = \mathcal{Y}(\mathcal{B}p'_j)$$



$\rightarrow$

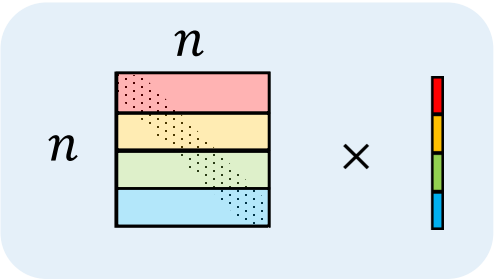


$$(r_{i+j}, r_{i+j}) = r_j'^T \mathcal{Y}^T \mathcal{Y} r'_j$$

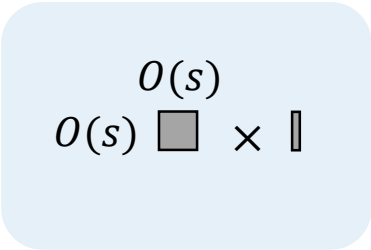
s-step CG

For s iterations of updates, inner products and SpMVs (in basis $\mathcal{Y}$) can be computed by independently by each processor without communication:

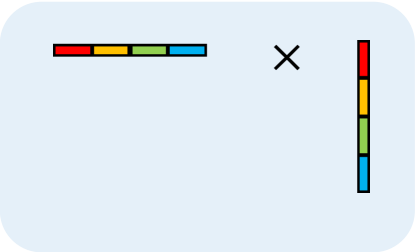
$$Ap_{i+j} \quad = \quad A\underline{\mathcal{Y}}p'_j \quad = \quad \mathcal{Y}(\mathcal{B}p'_j)$$



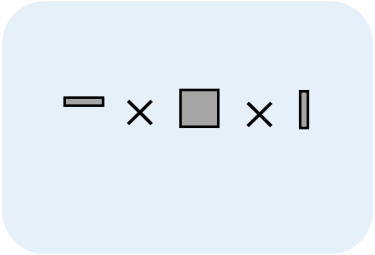
$\rightarrow$



$$(r_{i+j}, r_{i+j}) \quad = \quad r_j'^T \mathcal{Y}^T \mathcal{Y} r'_j \quad = \quad r_j'^T \mathcal{G} r'_j$$



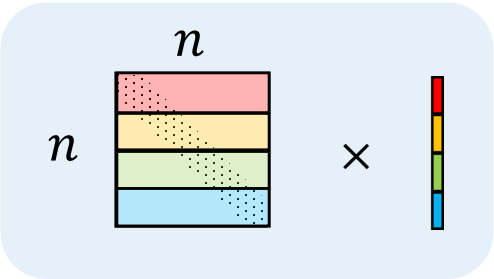
$\rightarrow$



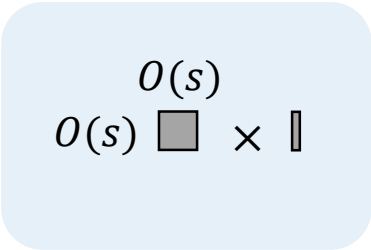
s-step CG

For s iterations of updates, inner products and SpMVs (in basis $\mathcal{Y}$) can be computed by independently by each processor without communication:

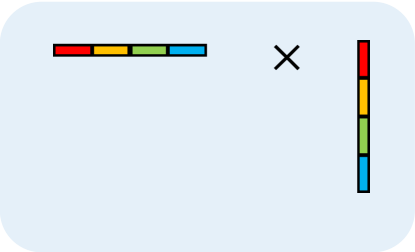
$$Ap_{i+j} = A\underline{\mathcal{Y}}p'_j = \mathcal{Y}(\mathcal{B}p'_j)$$



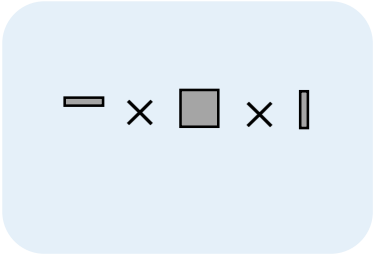
$\rightarrow$



$$(r_{i+j}, r_{i+j}) = r_j'^T \mathcal{Y}^T \mathcal{Y} r'_j = r_j'^T \mathcal{G} r'_j$$



$\rightarrow$



s-step CG

$r_0 = b - Ax_0, p_0 = r_0$

for $k = 0{:}\text{nmax}/s$

Compute $\mathcal{Y}_k$ and $\mathcal{B}_k$ such that $A\underline{\mathcal{Y}_k} = \mathcal{Y}_k\mathcal{B}_k$ and

$\text{span}(\mathcal{Y}_k) = \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$

$\mathcal{G}_k = \mathcal{Y}_k^T\mathcal{Y}_k$

$x_0' = 0, r_0' = e_{s+2}, p_0' = e_1$

for $j = 1{:}\,s$

$$\alpha_{sk+j-1} = \frac{r_{j-1}'^T\mathcal{G}_k r_{j-1}'}{p_{j-1}'^T\mathcal{G}_k\mathcal{B}_k p_{j-1}'}$$

$$x_j' = x_{j-1}' + \alpha_{sk+j-1}p_{j-1}'$$

$$r_j' = r_{j-1}' - \alpha_{sk+j-1}\mathcal{B}_k p_{j-1}'$$

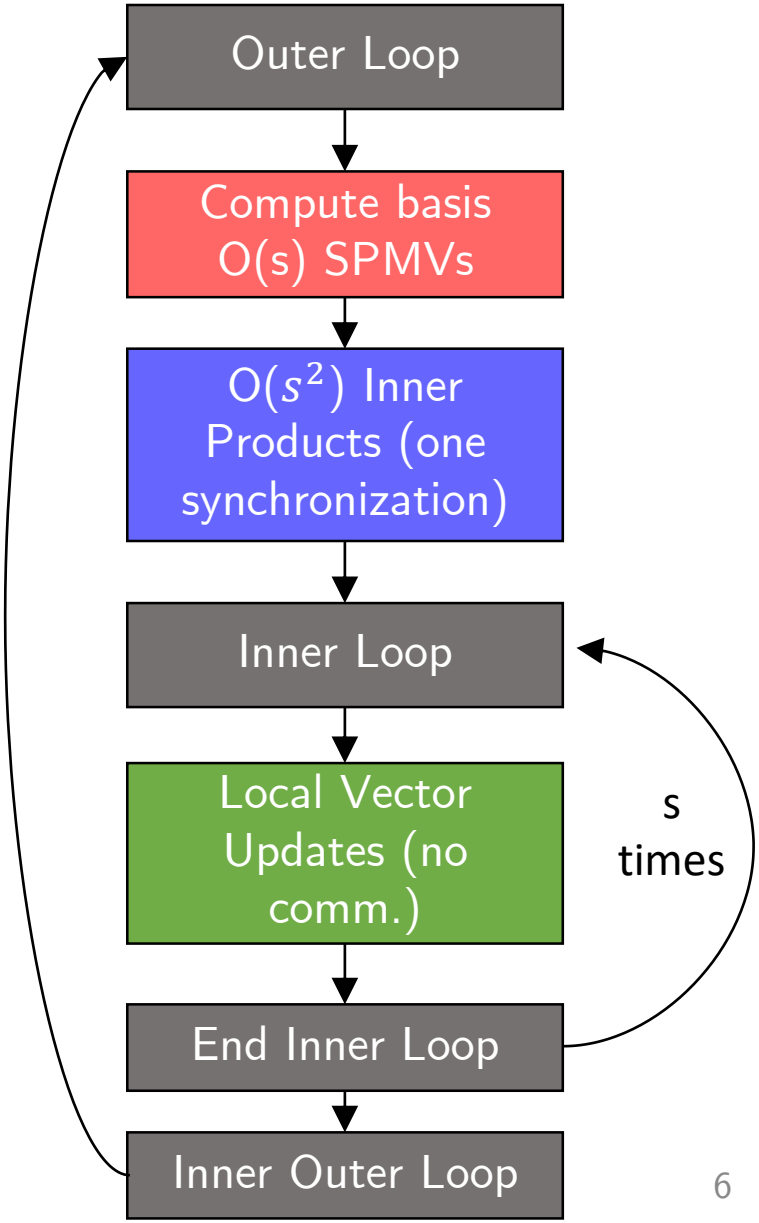
$$\beta_{sk+j} = \frac{r_j'^T\mathcal{G}_k r_j'}{r_{j-1}'^T\mathcal{G}_k r_{j-1}'}$$

$$p_j' = r_j' + \beta_{sk+j}p_{j-1}'$$

end

$[x_{s(k+1)} - x_{sk}, r_{s(k+1)}, p_{s(k+1)}] = \mathcal{Y}_k[x_s', r_s', p_s']$

end

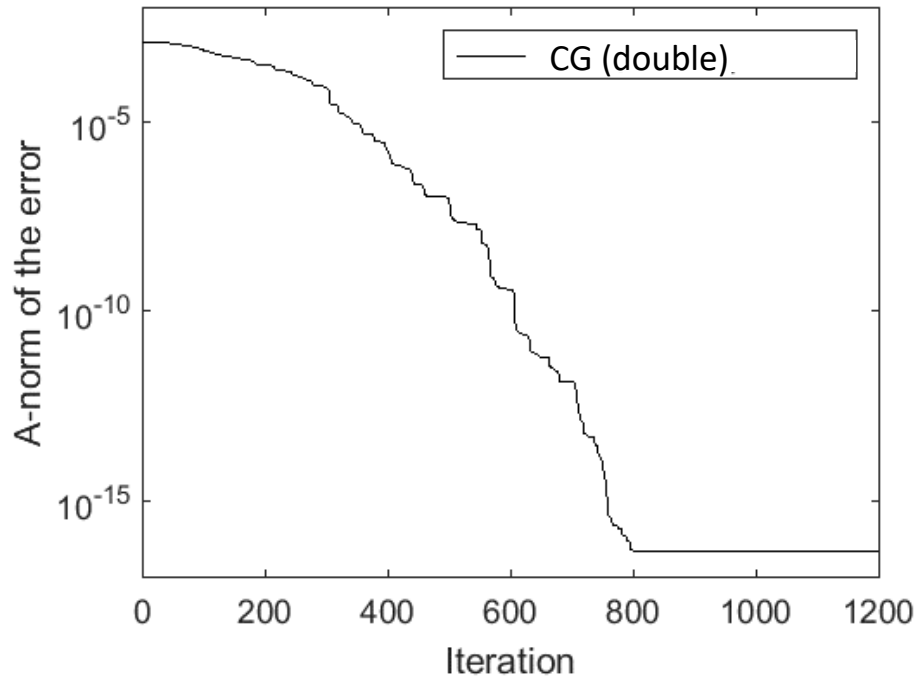


The effects of finite precision

Well-known that roundoff error has two effects:

1. Delay of convergence
   - No longer have exact Krylov subspace
   - Can lose numerical rank deficiency
   - Residuals no longer orthogonal - Minimization of $\|x - x_i\|_A$ no longer exact

2. Loss of attainable accuracy
   - Rounding errors cause true residual $b - Ax_i$ and updated residual r_i deviate!



A: bcsstk03 from SuiteSparse,
b: equal components in the eigenbasis of $A, \|b\| = 1$
$N = 112, \kappa(A) \approx 7e6$

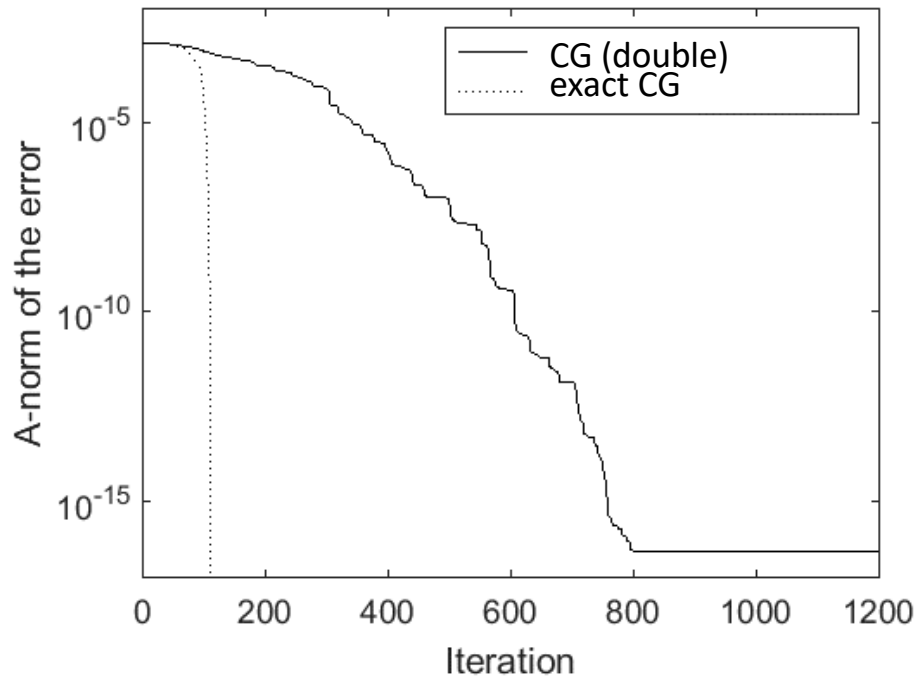
Much work on these results for CG; See Meurant and Strakoš (2006) for a thorough summary of early developments in finite precision analysis of Lanczos and CG

The effects of finite precision

Well-known that roundoff error has two effects:

1. Delay of convergence
   - No longer have exact Krylov subspace
   - Can lose numerical rank deficiency
   - Residuals no longer orthogonal - Minimization of $\|x - x_i\|_A$ no longer exact

2. Loss of attainable accuracy
   - Rounding errors cause true residual $b - Ax_i$ and updated residual r_i deviate!



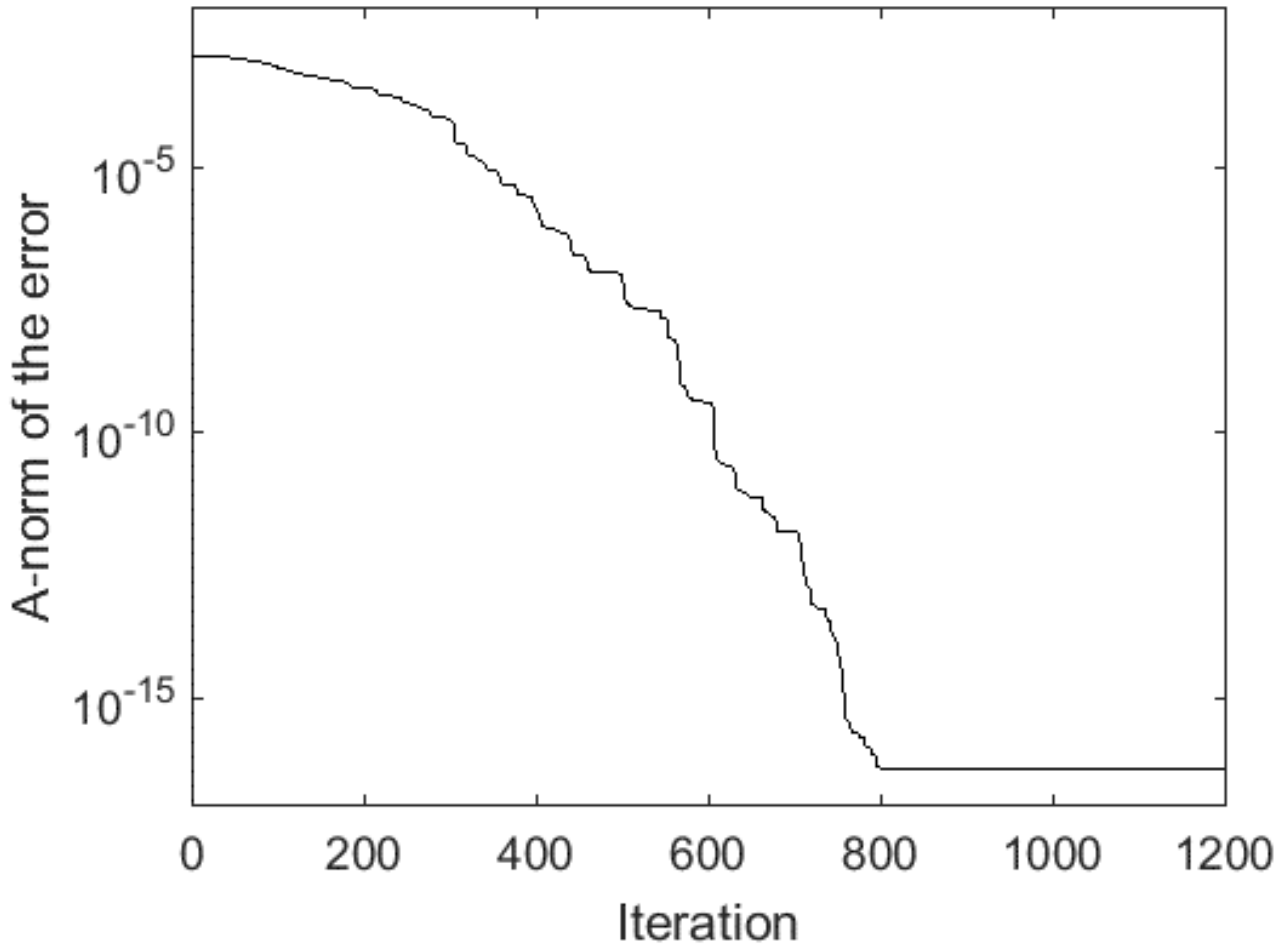
A: bcsstk03 from SuiteSparse,
b: equal components in the eigenbasis of $A, \|b\| = 1$
$N = 112, \kappa(A) \approx 7e6$

Much work on these results for CG; See Meurant and Strakoš (2006) for a thorough summary of early developments in finite precision analysis of Lanczos and CG

Conjugate Gradient method for solving Ax = b
double precision ($\varepsilon = 2^{-53}$)

$$\|x_i - x\|_A = \sqrt{(x_i - x)^T A (x_i - x)}$$

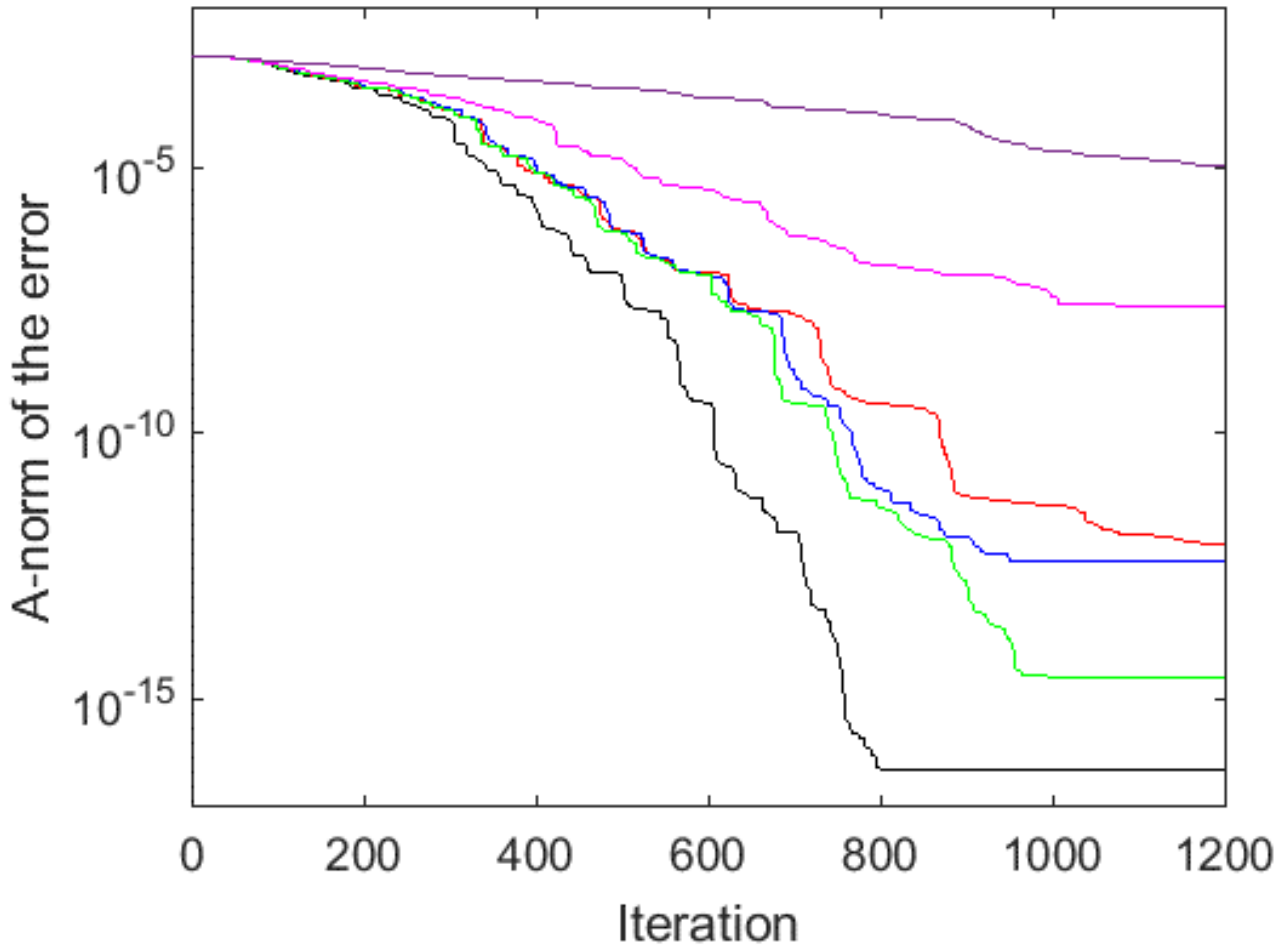
$$x_i = x_{i-1} + \alpha_i p_i$$
$$r_i = r_{i-1} - \alpha_i A p_i$$
$$p_i = r_i + \beta_i p_i$$



Conjugate Gradient method for solving Ax = b
double precision ($\varepsilon = 2^{-53}$)

$$\|x_i - x\|_A = \sqrt{(x_i - x)^T A (x_i - x)}$$

$$x_i = x_{i-1} + \alpha_i p_i$$
$$r_i = r_{i-1} - \alpha_i A p_i$$
$$p_i = r_i + \beta_i p_i$$



Lanczos Convergence Analysis [Paige, 1976]

Finite precision Lanczos process: (A is $n \times n$ with at most N nonzeros per row)

$$A\hat{V}_m = \hat{V}_m \hat{T}_m + \hat{\beta}_{m+1}\hat{v}_{m+1}e_m^T + \delta\hat{V}_m$$

$$\hat{V}_m = [\hat{v}_1, \dots, \hat{v}_m], \qquad \delta\hat{V}_m = [\delta\hat{v}_1, \dots, \delta\hat{v}_m], \qquad \hat{T}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & & \\ \hat{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \hat{\beta}_m \\ & & \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}$$

for $i \in \{1, \dots, m\}$,

$$\|\delta\hat{v}_i\|_2 \le \varepsilon_1\sigma$$
$$\hat{\beta}_{i+1}\left|\hat{v}_i^T\hat{v}_{i+1}\right| \le 2\varepsilon_0\sigma$$
$$\left|\hat{v}_{i+1}^T\hat{v}_{i+1} - 1\right| \le \varepsilon_0/2$$
$$\left|\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A\hat{v}_i\|_2^2\right| \le 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2$$

where $\sigma \equiv \|A\|_2$, and $\theta\sigma \equiv \||A|\|_2$

Classical Lanczos (Paige, 1976):

$$\varepsilon_0 = O(\varepsilon n)$$
$$\varepsilon_1 = O(\varepsilon N\theta)$$

Lanczos Convergence Analysis [Paige, 1976]

Finite precision Lanczos process: (A is $n \times n$ with at most N nonzeros per row)

$$A\hat{V}_m = \hat{V}_m \hat{T}_m + \hat{\beta}_{m+1}\hat{v}_{m+1}e_m^T + \delta\hat{V}_m$$

$$\hat{V}_m = [\hat{v}_1, \dots, \hat{v}_m], \qquad \delta\hat{V}_m = [\delta\hat{v}_1, \dots, \delta\hat{v}_m], \qquad \hat{T}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & & \\ \hat{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \hat{\beta}_m \\ & & \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}$$

for $i \in \{1, \dots, m\}$,

$$\|\delta\hat{v}_i\|_2 \le \varepsilon_1\sigma$$
$$\hat{\beta}_{i+1}\left|\hat{v}_i^T\hat{v}_{i+1}\right| \le 2\varepsilon_0\sigma$$
$$\left|\hat{v}_{i+1}^T\hat{v}_{i+1} - 1\right| \le \varepsilon_0/2$$
$$\left|\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A\hat{v}_i\|_2^2\right| \le 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2$$

where $\sigma \equiv \|A\|_2$, and $\theta\sigma \equiv \||A|\|_2$

Classical Lanczos (Paige, 1976):

$$\varepsilon_0 = O(\varepsilon n)$$
$$\varepsilon_1 = O(\varepsilon N\theta)$$

s-step Lanczos (C., 2015):

$$\varepsilon_0 = O\left(\varepsilon n \boldsymbol{\Gamma^2}\right)$$
$$\varepsilon_1 = O(\varepsilon N\theta\boldsymbol{\Gamma})$$

$$\Gamma = \max_{\ell \le k} \|\mathcal{Y}_\ell^+\|_2 \cdot \||\mathcal{Y}_\ell|\|_2$$

Paige's Results for Classical Lanczos (1980)

Using bounds on local rounding errors in Lanczos, showed that

1. The computed eigenvalues always lie between the extreme eigenvalues of A to within a small multiple of machine precision.
2. At least one small interval containing an eigenvalue of A is found by the nth iteration.
3. The algorithm behaves numerically like Lanczos with full reorthogonalization until a very close eigenvalue approximation is found.
4. The loss of orthogonality among basis vectors follows a rigorous pattern and implies that some computed eigenvalues have converged.

Results for s-step Lanczos

- Do Paige's results, e.g.,

  loss of orthogonality $\rightarrow$ eigenvalue convergence

  hold for s-step Lanczos?
- The answer is **YES!** …but
- Only if:
  - $\varepsilon_0 \equiv 2\varepsilon(n+11s+15)\,\Gamma^2 \leq \frac{1}{12}$
    - i.e., $\Gamma \leq \left(24\varepsilon(n+11s+15)\right)^{-1/2} = O\left(\frac{1}{\sqrt{n\varepsilon}}\right)$

Results for s-step Lanczos

- Do Paige's results, e.g.,

  loss of orthogonality $\rightarrow$ eigenvalue convergence

  hold for s-step Lanczos?
- The answer is **YES!** ...but
- With the additional caveat:
- Paige's results say: orthogonality is not lost until an eigenvalue has stabilized to within $O(\varepsilon)$ of an eigenvalue of A

Results for s-step Lanczos

- Do Paige's results, e.g.,

  loss of orthogonality $\rightarrow$ eigenvalue convergence

  hold for s-step Lanczos?
- The answer is **YES!** ...but

- With the additional caveat:
- Paige's results say: orthogonality is not lost until an eigenvalue has stabilized to within $O(\varepsilon)$ of an eigenvalue of A
- For s-step Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $O(\varepsilon\Gamma^2)$ of an eigenvalue of A
  - So the result is weaker: an eigenvalue is considered to be "stabilized" within a larger radius for the s-step case, and thus orthogonality is lost sooner
    - This explains the worse convergence behavior!

The case for extended precision

- The term Γ enters the bounds due to computation in the computed s-step basis
  - SpMVs cause Γ terms in the bounds
  - Inner products (computed using the Gram matrix) cause Γ^2 terms in the bounds

- Idea: use extended precision in computing and applying the Gram matrix
  - Computation only happens once every s iterations (doubles the size of the Allreduce)
  - Applying to vector happens every iteration, but the matrix is very small (sxs, fits in cache)

Mixed Precision Lanczos Analysis

Finite precision Lanczos process: (A is $n \times n$ with at most N nonzeros per row)

$$A\hat{V}_m = \hat{V}_m \hat{T}_m + \hat{\beta}_{m+1}\hat{v}_{m+1}e_m^T + \delta\hat{V}_m$$

$$\hat{V}_m = [\hat{v}_1, \dots, \hat{v}_m], \qquad \delta\hat{V}_m = [\delta\hat{v}_1, \dots, \delta\hat{v}_m], \qquad \hat{T}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & & \\ \hat{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \hat{\beta}_m \\ & & \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}$$

for $i \in \{1, \dots, m\}$,

$$\|\delta\hat{v}_i\|_2 \le \varepsilon_1\sigma$$
$$\hat{\beta}_{i+1}\left|\hat{v}_i^T\hat{v}_{i+1}\right| \le 2\varepsilon_0\sigma$$
$$\left|\hat{v}_{i+1}^T\hat{v}_{i+1} - 1\right| \le \varepsilon_0/2$$
$$\left|\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A\hat{v}_i\|_2^2\right| \le 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2$$

where $\sigma \equiv \|A\|_2$, and $\theta\sigma \equiv \||A|\|_2$

Classical Lanczos (Paige, 1976):

$$\varepsilon_0 = O(\varepsilon n)$$
$$\varepsilon_1 = O(\varepsilon N\theta)$$

s-step Lanczos (C., 2015):

$$\varepsilon_0 = O\left(\varepsilon n \boldsymbol{\Gamma}^2\right)$$
$$\varepsilon_1 = O(\varepsilon N\theta\boldsymbol{\Gamma})$$

$$\Gamma = \max_{\ell \le k} \|\mathcal{Y}_\ell^+\|_2 \cdot \||\mathcal{Y}_\ell|\|_2$$

Mixed Precision Lanczos Analysis

Finite precision Lanczos process: (A is $n \times n$ with at most N nonzeros per row)

$$A\hat{V}_m = \hat{V}_m \hat{T}_m + \hat{\beta}_{m+1}\hat{v}_{m+1}e_m^T + \delta\hat{V}_m$$

$$\hat{V}_m = [\hat{v}_1, \dots, \hat{v}_m], \qquad \delta\hat{V}_m = [\delta\hat{v}_1, \dots, \delta\hat{v}_m], \qquad \hat{T}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & & \\ \hat{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \hat{\beta}_m \\ & & \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}$$

for $i \in \{1, \dots, m\}$,

$$\|\delta\hat{v}_i\|_2 \le \varepsilon_1\sigma$$
$$\hat{\beta}_{i+1}\left|\hat{v}_i^T\hat{v}_{i+1}\right| \le 2\varepsilon_0\sigma$$
$$\left|\hat{v}_{i+1}^T\hat{v}_{i+1} - 1\right| \le \varepsilon_0/2$$
$$\left|\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A\hat{v}_i\|_2^2\right| \le 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2$$

where $\sigma \equiv \|A\|_2$, and $\theta\sigma \equiv \||A|\|_2$

Classical Lanczos (Paige, 1976):

$$\varepsilon_0 = O(\varepsilon n)$$
$$\varepsilon_1 = O(\varepsilon N\theta)$$

s-step Lanczos (C., 2015):

$$\varepsilon_0 = O\left(\varepsilon n \boldsymbol{\Gamma}^2\right)$$
$$\varepsilon_1 = O(\varepsilon N\theta\Gamma)$$

Mixed precision s-step Lanczos (C., Gergelits, Yamazaki, 2021):

$$\varepsilon_0 = O(\varepsilon \boldsymbol{\Gamma})$$
$$\varepsilon_1 = O(\varepsilon N\theta\Gamma)$$

$$\Gamma = \max_{\ell \le k} \|\mathcal{Y}_\ell^+\|_2 \cdot \||\mathcal{Y}_\ell|\|_2$$

Mixed precision s-step Lanczos analysis

Classical Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $\boldsymbol{O}(\boldsymbol{\varepsilon})$ of an eigenvalue of A

Uniform precision s-step Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $\boldsymbol{O}(\boldsymbol{\varepsilon\Gamma^2})$ of an eigenvalue of A

Results hold if $\boldsymbol{\Gamma} \leq \boldsymbol{O}\left(\frac{\mathbf{1}}{\sqrt{\boldsymbol{n\varepsilon}}}\right)$

Mixed precision s-step Lanczos analysis

Classical Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $\boldsymbol{O}(\boldsymbol{\varepsilon})$ of an eigenvalue of A

Uniform precision s-step Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $\boldsymbol{O}(\boldsymbol{\varepsilon\Gamma^2})$ of an eigenvalue of A

Results hold if $\boldsymbol{\Gamma} \leq \boldsymbol{O}\left(\frac{\mathbf{1}}{\sqrt{\boldsymbol{n\varepsilon}}}\right)$

Mixed precision s-step Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $\boldsymbol{O}(\boldsymbol{\varepsilon\Gamma})$ of an eigenvalue of A

Results hold if $\boldsymbol{\Gamma} \leq \boldsymbol{O}\left(\frac{\mathbf{1}}{\boldsymbol{n\varepsilon}}\right)$

⇒ For mixed precision case, expect orthogonality (and thus convergence behavior) to be somewhere between classical and (uniform precision) s-step Lanczos

⇒ Expect mixed precision algorithm can handle more ill-conditioned bases versus uniform precision algorithm

Extension to s-step CG

- s-step CG based on underlying s-step Lanczos procedure
- Better Ritz value accuracy and orthogonality in s-step Lanczos → better convergence behavior of mixed precision s-step CG

Extension to s-step CG

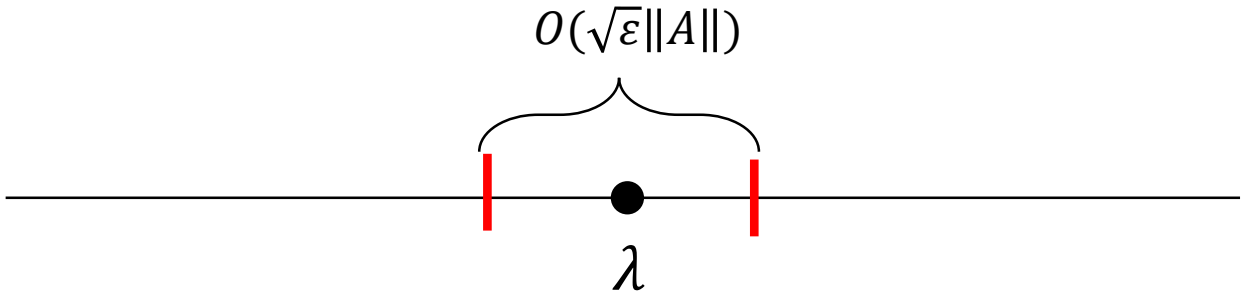
- s-step CG based on underlying s-step Lanczos procedure
- Better Ritz value accuracy and orthogonality in s-step Lanczos → better convergence behavior of mixed precision s-step CG
- But: extended precision computations in Gram matrix computations will not improve attainable accuracy
  - determined by precision in matrix-vector products

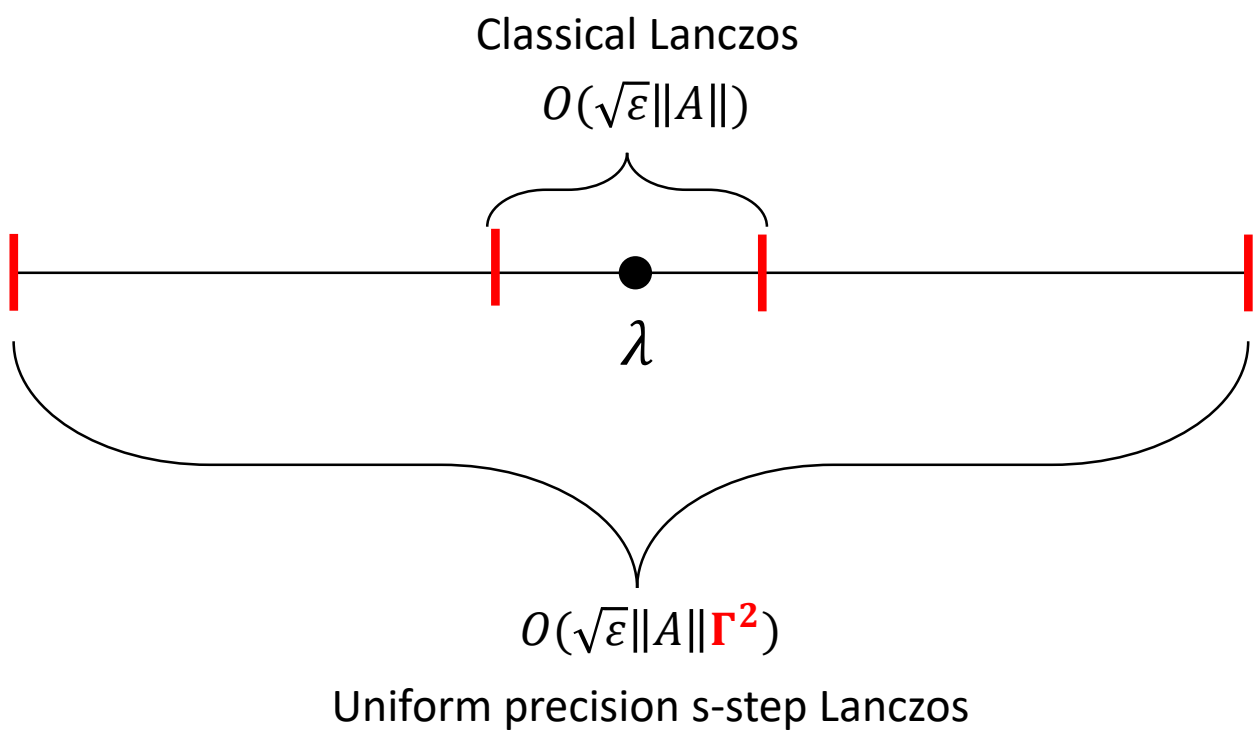
Extension to s-step CG

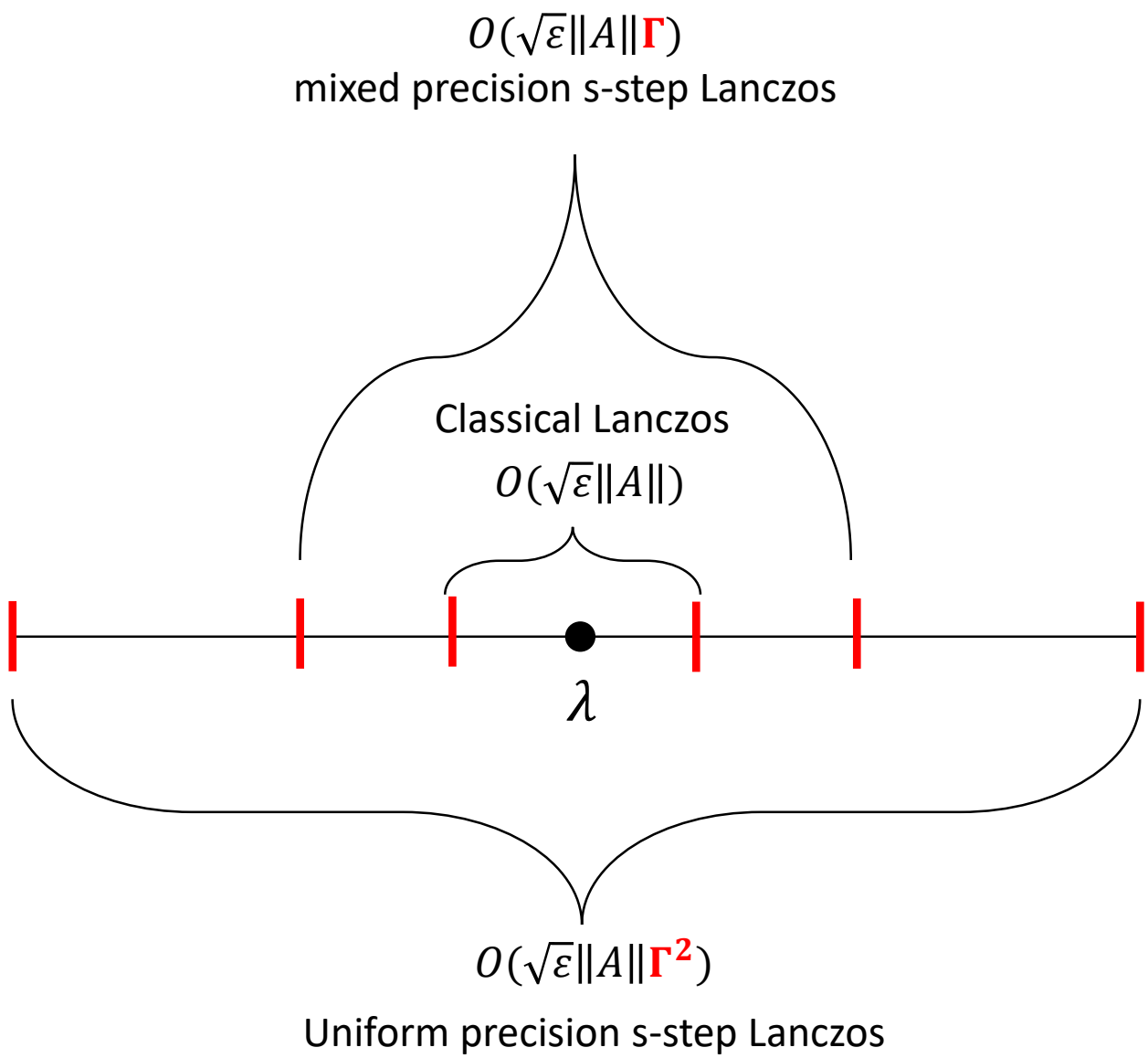
- s-step CG based on underlying s-step Lanczos procedure
- Better Ritz value accuracy and orthogonality in s-step Lanczos → better convergence behavior of mixed precision s-step CG
- But: extended precision computations in Gram matrix computations will not improve attainable accuracy
  - determined by precision in matrix-vector products

- Greenbaum (1989): finite precision classical CG behaves like exact CG applied to a larger matrix whose eigenvalues are in tight clusters around the eigenvalues of A.
- Can we extend this analysis?
  - Prediction: Cluster radius will contain a Γ^2 term for the uniform precision case, Γ term for the mixed precision case

Classical Lanczos



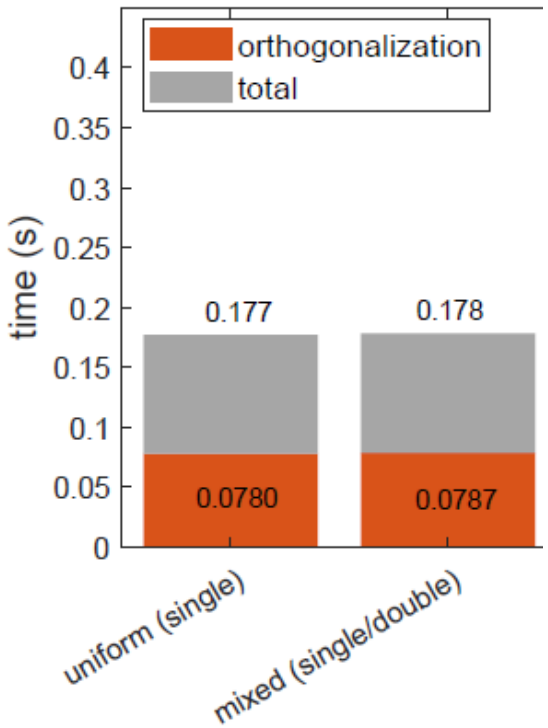

Classical Lanczos
$O(\sqrt{\varepsilon}\|A\|)$
λ
$O(\sqrt{\varepsilon}\|A\|\boldsymbol{\Gamma}^{\mathbf{2}})$
Uniform precision s-step Lanczos


$O(\sqrt{\varepsilon}\|A\|\boldsymbol{\Gamma})$
mixed precision s-step Lanczos
Classical Lanczos
$O(\sqrt{\varepsilon}\|A\|)$
λ
$O(\sqrt{\varepsilon}\|A\|\boldsymbol{\Gamma}^2)$
Uniform precision s-step Lanczos

What is the overhead?

- 3D Laplace matrix with $n = 100^3$
- 500 iterations of s-step CG with $s = 5$ on an NVIDIA V100 GPU
- Single/double: Uses KokkosBlas::DotBasedGemm for Gram matrix, computes $C = \alpha A^T B + \beta C$
  - Do not compute multiplication with α $(= 1)$
  - Only compute upper triangular part of C since symmetric
  - Input cast to double before being passed in

[Yamazaki, C., Kelley, 2022]



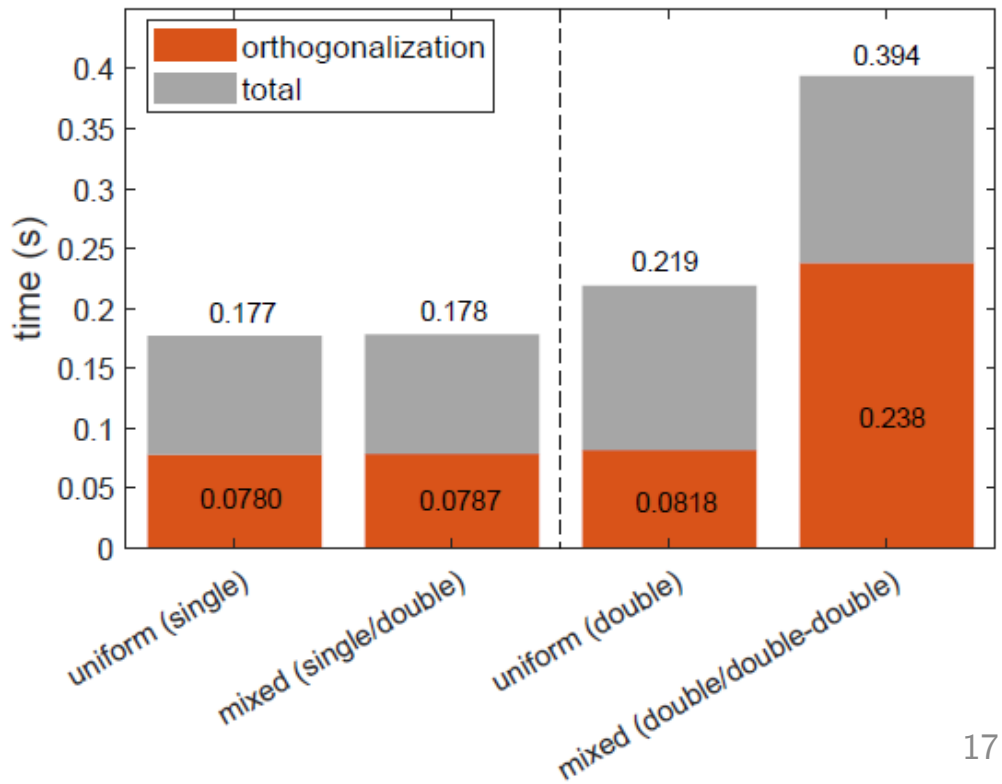
What is the overhead?

- 3D Laplace matrix with $n = 100^3$

[Yamazaki, C., Kelley, 2022]

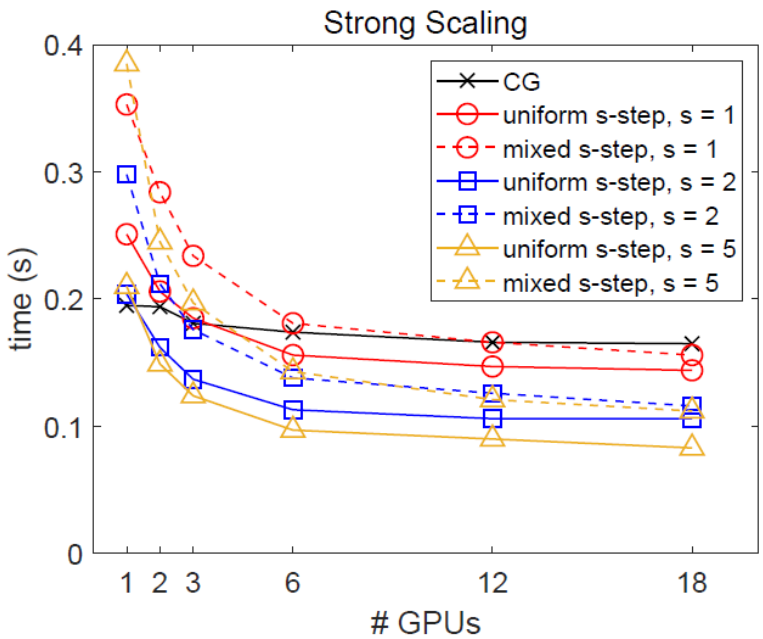
- 500 iterations of s-step CG with $s = 5$ on an NVIDIA V100 GPU
- Single/double: Uses KokkosBlas::DotBasedGemm for Gram matrix, computes $C = \alpha A^T B + \beta C$
  - Do not compute multiplication with α $(= 1)$
  - Only compute upper triangular part of C since symmetric
  - Input cast to double before being passed in

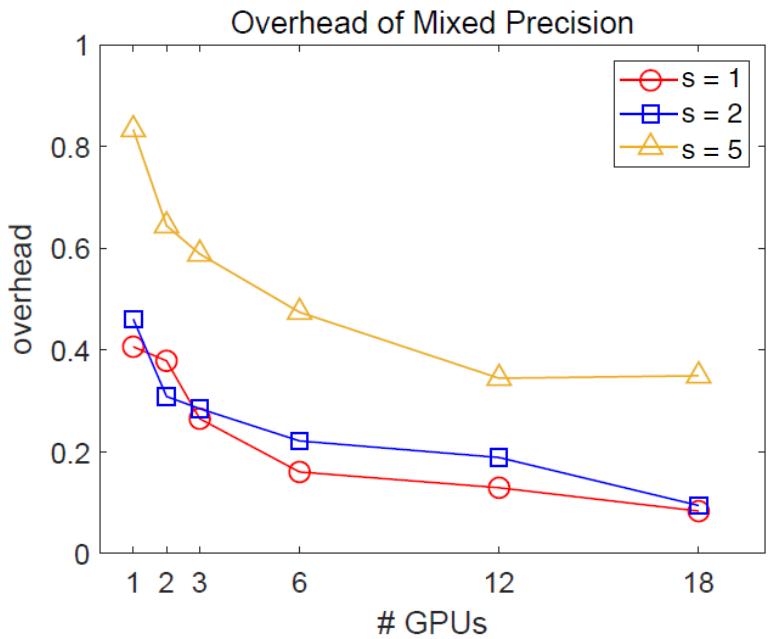
- Double/double-double: Software implementation of double-double (inner products require 17-21.5x more flops)
  - SinceKokkos does not support double-double arithmetic, our implementation uses a custom reducer for mixed-precision inner products on a GPU
  - For small double-double computations with the Gram matrix, we use multiprecision BLAS on the host CPU



Strong Scaling

- Same problem [Yamazaki, C., Kelley, 2022]
- Strong scaling up to 18 GPUs on Summit (6 GPUs per node)
- Using double/double-double



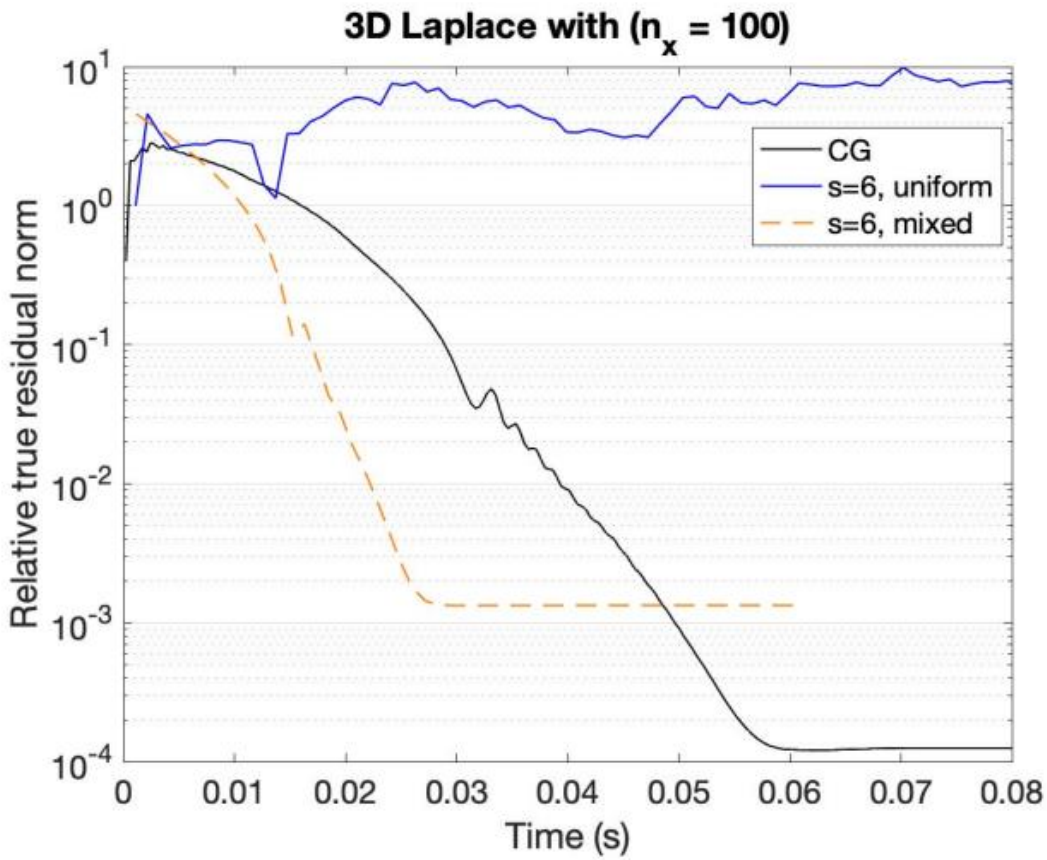


- Overhead of using software-implemented precision decreases as we scale up the hardware
  - Likely because latency becomes more dominant?

Time to Solution for Laplace Problem

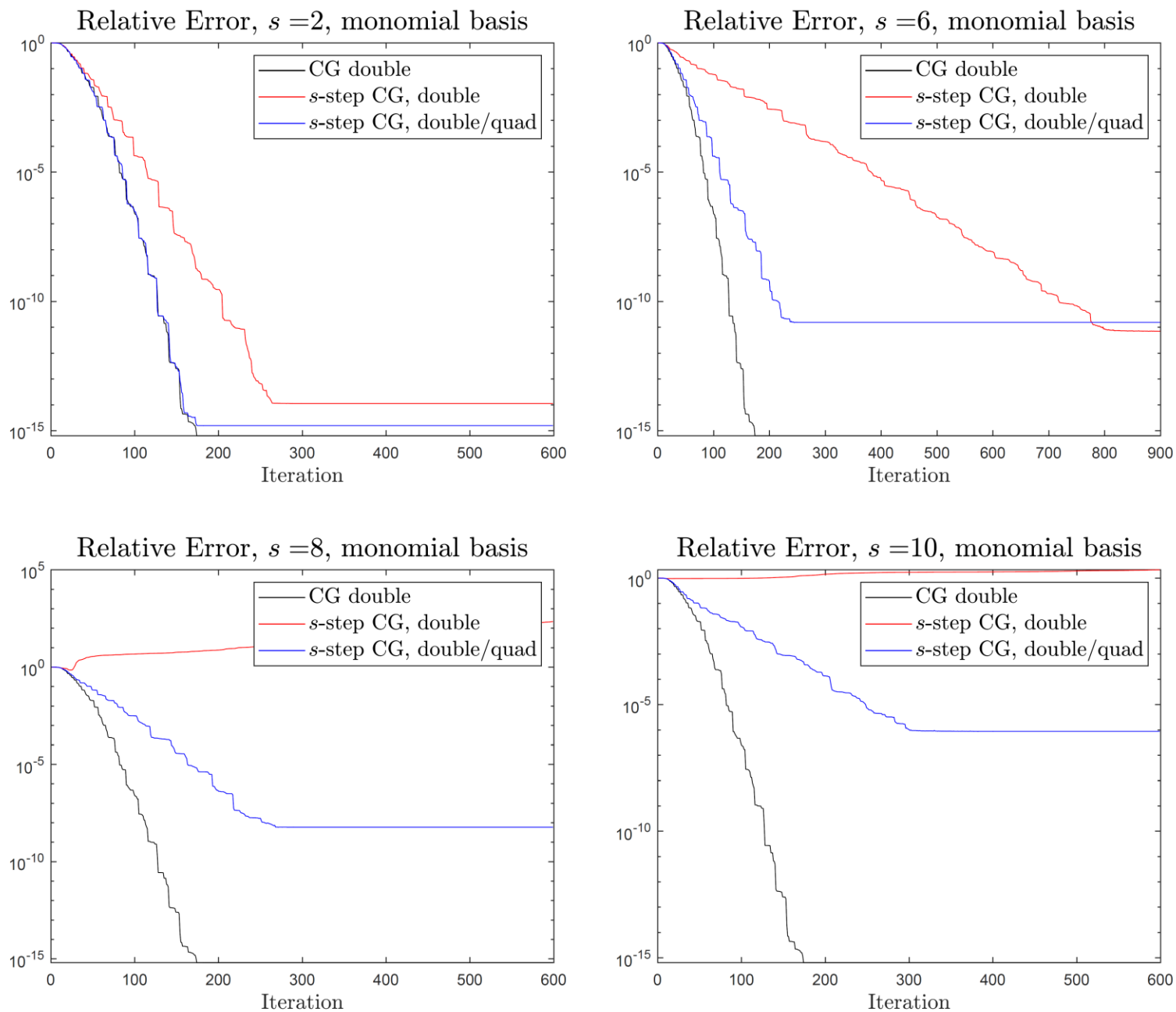
$\|b\|_2 = 1$, equal entries

[Yamazaki, C., Kelley, 2022]

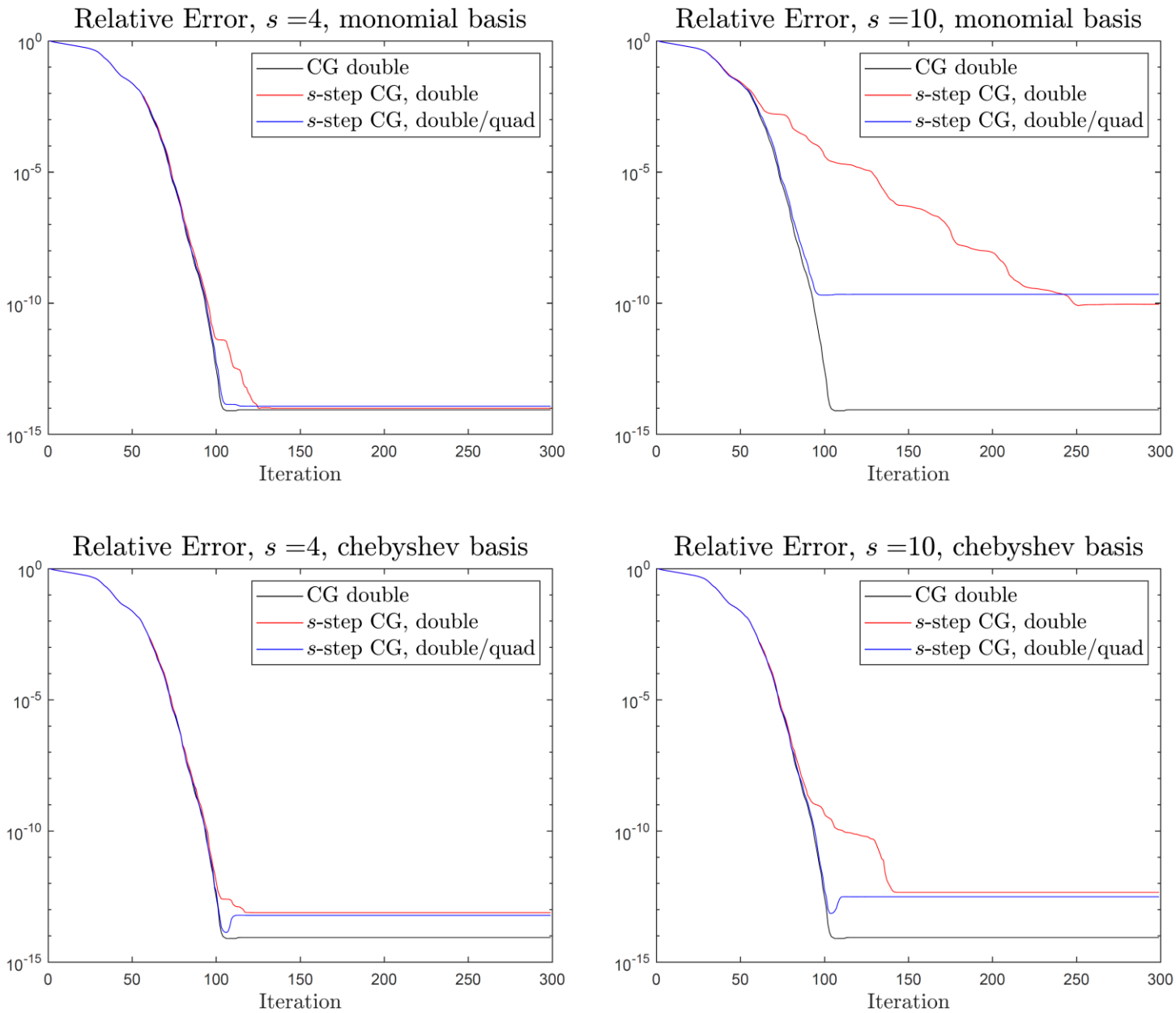


6 NVIDIA 100 GPUs, single working precision

Diagonal test problem, $n = 100$, $\kappa(A) = 10^5$, clustered eigenvalues



nos4 from SuiteSparse



Maximum attainable accuracy

- Accuracy $\|x - \hat{x}_i\|$ generally not computable, *but* $x - \hat{x}_i = A^{-1}(b - A\hat{x}_i)$
- Size of the true residual, $\|b - A\hat{x}_i\|$, used as computable measure of accuracy

Maximum attainable accuracy

- Accuracy $\|x - \hat{x}_i\|$ generally not computable, *but* $x - \hat{x}_i = A^{-1}(b - A\hat{x}_i)$
- Size of the true residual, $\|b - A\hat{x}_i\|$, used as computable measure of accuracy
- Rounding errors cause the **true residual,** $\boldsymbol{b - A\hat{x}_i}$, and the **updated residual,** $\boldsymbol{\hat{r}_i}$, to deviate

Maximum attainable accuracy

- Accuracy $\|x - \hat{x}_i\|$ generally not computable, *but* $x - \hat{x}_i = A^{-1}(b - A\hat{x}_i)$
- Size of the true residual, $\|b - A\hat{x}_i\|$, used as computable measure of accuracy
- Rounding errors cause the **true residual,** $\boldsymbol{b - A\hat{x}_i}$, and the **updated residual,** $\boldsymbol{\hat{r}_i}$, to deviate

- Writing $b - A\hat{x}_i = \hat{r}_i + b - A\hat{x}_i - \hat{r}_i$,

$$\|b - A\hat{x}_i\| \leq \|\hat{r}_i\| + \|b - A\hat{x}_i - \hat{r}_i\|$$

Maximum attainable accuracy

- Accuracy $\|x - \hat{x}_i\|$ generally not computable, *but* $x - \hat{x}_i = A^{-1}(b - A\hat{x}_i)$
- Size of the true residual, $\|b - A\hat{x}_i\|$, used as computable measure of accuracy
- Rounding errors cause the **true residual, $\boldsymbol{b - A\hat{x}_i}$**, and the **updated residual, $\boldsymbol{\hat{r}_i}$**, to deviate

- Writing $b - A\hat{x}_i = \hat{r}_i + b - A\hat{x}_i - \hat{r}_i$,

$$\|b - A\hat{x}_i\| \leq \|\hat{r}_i\| + \|b - A\hat{x}_i - \hat{r}_i\|$$

  - As $\|\hat{r}_i\| \to 0$, $\|b - A\hat{x}_i\|$ depends on $\|b - A\hat{x}_i - \hat{r}_i\|$

Maximum attainable accuracy

- Accuracy $\|x - \hat{x}_i\|$ generally not computable, *but* $x - \hat{x}_i = A^{-1}(b - A\hat{x}_i)$
- Size of the true residual, $\|b - A\hat{x}_i\|$, used as computable measure of accuracy
- Rounding errors cause the **true residual,** $\boldsymbol{b - A\hat{x}_i}$, and the **updated residual,** $\boldsymbol{\hat{r}_i}$, to deviate

- Writing $b - A\hat{x}_i = \hat{r}_i + b - A\hat{x}_i - \hat{r}_i$,

$$\|b - A\hat{x}_i\| \leq \|\hat{r}_i\| + \|b - A\hat{x}_i - \hat{r}_i\|$$

  - As $\|\hat{r}_i\| \to 0$, $\|b - A\hat{x}_i\|$ depends on $\|b - A\hat{x}_i - \hat{r}_i\|$

- Many results on bounding attainable accuracy, e.g.: Greenbaum (1989, 1994, 1997), Sleijpen, van der Vorst and Fokkema (1994), Sleijpen, van der Vorst and Modersitzki (2001), Björck, Elfving and Strakoš (1998) and Gutknecht and Strakoš (2000).

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \boldsymbol{\delta x_i}$$

and

$$\hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \boldsymbol{\delta r_i}$$

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \boldsymbol{\delta x_i} \quad \text{and} \quad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \boldsymbol{\delta r_i}$$

- Let $f_i \equiv b - A\hat{x}_i - \hat{r}_i$

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \boldsymbol{\delta x_i} \quad \text{and} \quad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \boldsymbol{\delta r_i}$$

- Let $f_i \equiv b - A\hat{x}_i - \hat{r}_i$

$$f_i = b - A(\hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \delta x_i) - (\hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \delta r_i)$$

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \boldsymbol{\delta x_i} \quad \text{and} \quad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \boldsymbol{\delta r_i}$$

- Let $f_i \equiv b - A\hat{x}_i - \hat{r}_i$

$$\begin{aligned} f_i &= b - A(\hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \delta x_i) - (\hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \delta r_i) \\ &= f_{i-1} + A\delta x_i + \delta r_i \end{aligned}$$

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \boldsymbol{\delta x_i} \quad \text{and} \quad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \boldsymbol{\delta r_i}$$

- Let $f_i \equiv b - A\hat{x}_i - \hat{r}_i$

$$\begin{aligned} f_i &= b - A(\hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \delta x_i) - (\hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \delta r_i) \\ &= f_{i-1} + A\delta x_i + \delta r_i \\ &= f_0 + \textstyle\sum_{m=1}^{i}(A\delta x_m + \delta r_m) \end{aligned}$$

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \boldsymbol{\delta x_i} \quad \text{and} \quad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \boldsymbol{\delta r_i}$$

- Let $f_i \equiv b - A\hat{x}_i - \hat{r}_i$

$$\begin{aligned} f_i &= b - A(\hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \delta x_i) - (\hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \delta r_i) \\ &= f_{i-1} + A\delta x_i + \delta r_i \\ &= f_0 + \textstyle\sum_{m=1}^{i}(A\delta x_m + \delta r_m) \end{aligned}$$

$\|f_i\| \leq O(\varepsilon) \sum_{m=0}^{i} N_A \|A\| \|\hat{x}_m\| + \|\hat{r}_m\|$ van der Vorst and Ye, 2000

$\|f_i\| \leq O(\varepsilon)\|A\| \left(\|x\| + \max_{m=0,\ldots,i} \|\hat{x}_m\|\right)$ Greenbaum, 1997

$\|f_i\| \leq O(\varepsilon) N_A \||A|\| \|A^{-1}\| \sum_{m=0}^{i} \|\hat{r}_m\|$ Sleijpen and van der Vorst, 1995

Sources of local roundoff error in s-step CG

Computing the s-step Krylov subspace basis:

$$A\underline{\hat{\mathcal{Y}}}_k = \hat{\mathcal{Y}}_k \mathcal{B}_k + \Delta \mathcal{Y}_k$$

Updating coordinate vectors in the inner loop:

$$\hat{x}'_{k,j} = \hat{x}'_{k,j-1} + \hat{q}'_{k,j-1} + \xi_{k,j}$$
$$\hat{r}'_{k,j} = \hat{r}'_{k,j-1} - \mathcal{B}_k\, \hat{q}'_{k,j-1} + \eta_{k,j}$$
$$\text{with} \quad \hat{q}'_{k,j-1} = \mathrm{fl}(\hat{\alpha}_{sk+j-1}\hat{p}'_{k,j-1})$$

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+j} = \hat{\mathcal{Y}}_k \hat{x}'_{k,j} + \hat{x}_{sk} + \phi_{sk+j}$$
$$\hat{r}_{sk+j} = \hat{\mathcal{Y}}_k \hat{r}'_{k,j} + \psi_{sk+j}$$

Sources of local roundoff error in s-step CG

Computing the s-step Krylov subspace basis:

$$A\underline{\hat{\mathcal{Y}}}_k = \hat{\mathcal{Y}}_k \mathcal{B}_k + \Delta\mathcal{Y}_k$$

Error in computing s-step basis

Updating coordinate vectors in the inner loop:

$$\hat{x}'_{k,j} = \hat{x}'_{k,j-1} + \hat{q}'_{k,j-1} + \xi_{k,j}$$

$$\hat{r}'_{k,j} = \hat{r}'_{k,j-1} - \mathcal{B}_k\, \hat{q}'_{k,j-1} + \eta_{k,j}$$

$$\text{with} \quad \hat{q}'_{k,j-1} = \mathrm{fl}(\hat{\alpha}_{sk+j-1}\hat{p}'_{k,j-1})$$

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+j} = \hat{\mathcal{Y}}_k \hat{x}'_{k,j} + \hat{x}_{sk} + \phi_{sk+j}$$

$$\hat{r}_{sk+j} = \hat{\mathcal{Y}}_k \hat{r}'_{k,j} + \psi_{sk+j}$$

Sources of local roundoff error in s-step CG

Computing the s-step Krylov subspace basis:

$$A\underline{\hat{\mathcal{Y}}}_k = \hat{\mathcal{Y}}_k \mathcal{B}_k + \Delta\mathcal{Y}_k$$

Error in computing s-step basis

Updating coordinate vectors in the inner loop:

$$\hat{x}'_{k,j} = \hat{x}'_{k,j-1} + \hat{q}'_{k,j-1} + \xi_{k,j}$$

$$\hat{r}'_{k,j} = \hat{r}'_{k,j-1} - \mathcal{B}_k\, \hat{q}'_{k,j-1} + \eta_{k,j}$$

$$\text{with} \quad \hat{q}'_{k,j-1} = \mathrm{fl}(\hat{\alpha}_{sk+j-1}\hat{p}'_{k,j-1})$$

Error in updating coefficient vectors

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+j} = \hat{\mathcal{Y}}_k \hat{x}'_{k,j} + \hat{x}_{sk} + \phi_{sk+j}$$

$$\hat{r}_{sk+j} = \hat{\mathcal{Y}}_k \hat{r}'_{k,j} + \psi_{sk+j}$$

Sources of local roundoff error in s-step CG

Computing the s-step Krylov subspace basis:

$$A\underline{\hat{\mathcal{Y}}}_k = \hat{\mathcal{Y}}_k \mathcal{B}_k + \Delta\mathcal{Y}_k$$

Error in computing s-step basis

Updating coordinate vectors in the inner loop:

$$\hat{x}'_{k,j} = \hat{x}'_{k,j-1} + \hat{q}'_{k,j-1} + \xi_{k,j}$$

$$\hat{r}'_{k,j} = \hat{r}'_{k,j-1} - \mathcal{B}_k\, \hat{q}'_{k,j-1} + \eta_{k,j}$$

$$\text{with} \quad \hat{q}'_{k,j-1} = \mathrm{fl}(\hat{\alpha}_{sk+j-1}\hat{p}'_{k,j-1})$$

Error in updating coefficient vectors

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+j} = \hat{\mathcal{Y}}_k \hat{x}'_{k,j} + \hat{x}_{sk} + \phi_{sk+j}$$

$$\hat{r}_{sk+j} = \hat{\mathcal{Y}}_k \hat{r}'_{k,j} + \psi_{sk+j}$$

Error in basis change

Attainable accuracy of s-step CG

- We can write the gap between the true and updated residuals f in terms of these errors:

$$f_{sk+j} = f_0$$

$$-\sum_{\ell=0}^{k-1}\left[A\phi_{s\ell+s} + \psi_{s\ell+s} + \sum_{i=1}^{s}\left[A\hat{\mathcal{Y}}_\ell \xi_{\ell,i} + \hat{\mathcal{Y}}_\ell \eta_{\ell,i} - \Delta\mathcal{Y}_\ell \hat{q}'_{\ell,i-1}\right]\right]$$

$$-A\phi_{sk+j} - \psi_{sk+j} - \sum_{i=1}^{j}\left[A\hat{\mathcal{Y}}_k \xi_{k,i} + \hat{\mathcal{Y}}_k \eta_{k,i} - \Delta\mathcal{Y}_\ell \hat{q}'_{k,i-1}\right]$$

- Using standard rounding error results, this allows us to obtain an upper bound on $\|f_{sk+j}\|$.

Attainable accuracy of s-step CG

- We can write the gap between the true and updated residuals f in terms of these errors:

$$f_{sk+j} = f_0 - \sum_{\ell=0}^{k-1}\left[A\phi_{s\ell+s} + \psi_{s\ell+s} + \sum_{i=1}^{s}\left[A\hat{\mathcal{Y}}_\ell \xi_{\ell,i} + \hat{\mathcal{Y}}_\ell \eta_{\ell,i} - \Delta\mathcal{Y}_\ell \hat{q}'_{\ell,i-1}\right]\right] - A\phi_{sk+j} - \psi_{sk+j} - \sum_{i=1}^{j}\left[A\hat{\mathcal{Y}}_k \xi_{k,i} + \hat{\mathcal{Y}}_k \eta_{k,i} - \Delta\mathcal{Y}_\ell \hat{q}'_{k,i-1}\right]$$

- Using standard rounding error results, this allows us to obtain an upper bound on $\|f_{sk+j}\|$.

Attainable accuracy of s-step CG

- We can write the gap between the true and updated residuals f in terms of these errors:

$$f_{sk+j} = f_0$$

$$-\sum_{\ell=0}^{k-1}\left[A\phi_{s\ell+s} + \psi_{s\ell+s} + \sum_{i=1}^{s}\left[A\hat{\mathcal{Y}}_\ell \xi_{\ell,i} + \hat{\mathcal{Y}}_\ell \eta_{\ell,i} - \Delta\mathcal{Y}_\ell \hat{q}'_{\ell,i-1}\right]\right]$$

$$-A\phi_{sk+j} - \psi_{sk+j} - \sum_{i=1}^{j}\left[A\hat{\mathcal{Y}}_k \xi_{k,i} + \hat{\mathcal{Y}}_k \eta_{k,i} - \Delta\mathcal{Y}_\ell \hat{q}'_{k,i-1}\right]$$

- Using standard rounding error results, this allows us to obtain an upper bound on $\|f_{sk+j}\|$.

Attainable accuracy of s-step CG

- We can write the gap between the true and updated residuals f in terms of these errors:

$$f_{sk+j} = f_0 - \sum_{\ell=0}^{k-1}\left[A\phi_{s\ell+s} + \psi_{s\ell+s} + \sum_{i=1}^{s}\left[A\hat{\mathcal{Y}}_\ell \xi_{\ell,i} + \hat{\mathcal{Y}}_\ell \eta_{\ell,i} - \Delta\mathcal{Y}_\ell \hat{q}'_{\ell,i-1}\right]\right] - A\phi_{sk+j} - \psi_{sk+j} - \sum_{i=1}^{j}\left[A\hat{\mathcal{Y}}_k \xi_{k,i} + \hat{\mathcal{Y}}_k \eta_{k,i} - \Delta\mathcal{Y}_\ell \hat{q}'_{k,i-1}\right]$$

- Using standard rounding error results, this allows us to obtain an upper bound on $\|f_{sk+j}\|$.

Attainable accuracy of s-step CG

$f_i \equiv b - A\hat{x}_i - \hat{r}_i$

For CG:

$$\|f_i\| \leq \|f_0\| + \varepsilon \sum_{m=1}^{i} (1+N)\|A\|\|\hat{x}_m\| + \|\hat{r}_m\|$$

Attainable accuracy of s-step CG

$f_i \equiv b - A\hat{x}_i - \hat{r}_i$

For CG:

$$\|f_i\| \leq \|f_0\| + \varepsilon \sum_{m=1}^{i} (1+N)\|A\|\|\hat{x}_m\| + \|\hat{r}_m\|$$

For s-step CG: $i \equiv sk + j$

$$\|f_{sk+j}\| \leq \|f_0\| + \varepsilon \boldsymbol{c\bar{\Gamma}_k} \sum_{m=1}^{sk+j} (1+N)\|A\|\|\hat{x}_m\| + \|\hat{r}_m\|$$

where c is a low-degree polynomial in s, and

$$\bar{\Gamma}_k = \max_{\ell \leq k} \Gamma_\ell \ , \qquad \text{where} \qquad \Gamma_\ell = \|\hat{\mathcal{Y}}_\ell^+\| \cdot \||\hat{\mathcal{Y}}_\ell|\|$$

(see C., 2015)

Residual replacement strategy

- Improve accuracy by replacing **computed residual** $\hat{r}_i$ by the **true residual** $\boldsymbol{b} - \boldsymbol{A}\hat{x}_i$ in certain iterations
  - Related work for classical CG: van der Vorst and Ye (1999)

Residual replacement strategy

- Improve accuracy by replacing **computed residual $\hat{r}_i$** by the **true residual $b - A\hat{x}_i$** in certain iterations
  - Related work for classical CG: van der Vorst and Ye (1999)

- Choose when to replace $\hat{r}_i$ with $b - A\hat{x}_i$ to meet two constraints:
  1. $\|f_i\| = \|b - A\hat{x}_i - \hat{r}_i\|$ is small (relative to $\varepsilon N \|A\| \|\hat{x}_{m+1}\|$)
  2. Convergence rate is maintained (avoid large perturbations to finite precision CG recurrence)

Residual replacement strategy

- Improve accuracy by replacing **computed residual $\hat{r}_i$** by the **true residual $b - A\hat{x}_i$** in certain iterations
  - Related work for classical CG: van der Vorst and Ye (1999)

- Choose when to replace $\hat{r}_i$ with $b - A\hat{x}_i$ to meet two constraints:
  1. $\|f_i\| = \|b - A\hat{x}_i - \hat{r}_i\|$ is small (relative to $\varepsilon N\|A\|\|\hat{x}_{m+1}\|$)
  2. Convergence rate is maintained (avoid large perturbations to finite precision CG recurrence)

- Based on derived bound on deviation of residuals, can devise a residual replacement strategy for s-step CG

- Implementation has **negligible cost**

Residual replacement for s-step CG

- Use computable bound for $\|b - A\hat{x}_i - \hat{r}_i\|$ to update d_i, an estimate of error in computing r_i, in each iteration
- Set threshold $\hat{\varepsilon} \approx \sqrt{\varepsilon}$, replace whenever $d_i / \|r_i\|$ reaches threshold

Residual replacement for s-step CG

- Use computable bound for $\|b - A\hat{x}_i - \hat{r}_i\|$ to update d_i, an estimate of error in computing r_i, in each iteration
- Set threshold $\hat{\varepsilon} \approx \sqrt{\varepsilon}$, replace whenever $d_i/\|r_i\|$ reaches threshold

Pseudo-code for residual replacement with group update for s-step CG:

```
if  d_{i-1} ≤ ε̂‖r_{i-1}‖  and  d_i > ε̂‖r_i‖  and  d_i > 1.1 d_init
        z = z + Y_k x'_{k,j} + x_sk
        x_i = 0
        r_i = b − Az
        d_init = d_i = ε((1 + 2N')‖A‖‖z‖ + ‖r_i‖)
        p_i = Y_k p'_{k,j}
        break from inner loop and begin new outer loop
end
```

Residual replacement for s-step CG

- Use computable bound for $\|b - A\hat{x}_i - \hat{r}_i\|$ to update d_i, an estimate of error in computing r_i, in each iteration
- Set threshold $\hat{\varepsilon} \approx \sqrt{\varepsilon}$, replace whenever $d_i/\|r_i\|$ reaches threshold

Pseudo-code for residual replacement with group update for s-step CG:

```
if  d_{i-1} ≤ ε̂‖r_{i-1}‖  and  d_i > ε̂‖r_i‖  and  d_i > 1.1 d_init
        z = z + Y_k x'_{k,j} + x_sk
        x_i = 0
        r_i = b − Az
        d_init = d_i = ε((1 + 2N')‖A‖‖z‖ + ‖r_i‖)
        p_i = Y_k p'_{k,j}
        break from inner loop and begin new outer loop
end
```

group update of approximate solution

Residual replacement for s-step CG

- Use computable bound for $\|b - A\hat{x}_i - \hat{r}_i\|$ to update d_i, an estimate of error in computing r_i, in each iteration
- Set threshold $\hat{\varepsilon} \approx \sqrt{\varepsilon}$, replace whenever $d_i/\|r_i\|$ reaches threshold

Pseudo-code for residual replacement with group update for s-step CG:

```
if  d_{i-1} ≤ ε̂‖r_{i-1}‖  and  d_i > ε̂‖r_i‖  and  d_i > 1.1 d_init
        z = z + Y_k x'_{k,j} + x_sk
        x_i = 0
        r_i = b − Az
        d_init = d_i = ε((1 + 2N')‖A‖‖z‖ + ‖r_i‖)
        p_i = Y_k p'_{k,j}
        break from inner loop and begin new outer loop
end
```

group update of approximate solution

set residual to true residual

Residual replacement for s-step CG

- Use computable bound for $\|b - A\hat{x}_i - \hat{r}_i\|$ to update d_i, an estimate of error in computing r_i, in each iteration
- Set threshold $\hat{\varepsilon} \approx \sqrt{\varepsilon}$, replace whenever $d_i/\|r_i\|$ reaches threshold

Pseudo-code for residual replacement with group update for s-step CG:

```
if  d_{i-1} ≤ ε̂‖r_{i-1}‖  and  d_i > ε̂‖r_i‖  and  d_i > 1.1 d_init
        z = z + Y_k x'_{k,j} + x_sk
        x_i = 0
        r_i = b − Az
        d_init = d_i = ε((1 + 2N')‖A‖‖z‖ + ‖r_i‖)
        p_i = Y_k p'_{k,j}
        break from inner loop and begin new outer loop
end
```

group update of approximate solution

set residual to true residual

A computable bound

- In each iteration, update error estimate d_i $(i \equiv sk + j)$ by:

$$
\begin{aligned}
d_i \equiv\ & d_{i-1} \\
& + \varepsilon\left[(4+N')\left(\|A\|\ \left\|\left|\hat{\mathcal{Y}}_k\right|\cdot\left|\hat{x}'_{k,j}\right|\right\| + \left\|\left|\hat{\mathcal{Y}}_k\right|\cdot\left|\mathcal{B}_k\right|\cdot\left|\hat{x}'_{k,j}\right|\right\|\right) + \left\|\left|\hat{\mathcal{Y}}_k\right|\cdot\left|\hat{r}'_{k,j}\right|\right\|\right] \\
& + \varepsilon \begin{cases} \|A\|\|\hat{x}_{sk+s}\| + (2+2N')\|A\|\left\|\left|\hat{\mathcal{Y}}_k\right|\cdot\left|\hat{x}'_{k,s}\right|\right\| + N'\left\|\left|\hat{\mathcal{Y}}_k\right|\cdot\left|\hat{r}'_{k,s}\right|\right\|, & j = s \\ 0, & \text{o.w.} \end{cases}
\end{aligned}
$$

where $N' = \max(N, 2s+1)$.

A computable bound

- In each iteration, update error estimate d_i $(i \equiv sk + j)$ by:

Estimated only once

$$
\begin{aligned}
d_i \equiv\ & d_{i-1} \\
& + \varepsilon\left[(4+N')\left(\|A\| \left\| |\hat{\mathcal{Y}}_k| \cdot |\hat{x}'_{k,j}| \right\| + \left\| |\hat{\mathcal{Y}}_k| \cdot |\mathcal{B}_k| \cdot |\hat{x}'_{k,j}| \right\|\right) + \left\| |\hat{\mathcal{Y}}_k| \cdot |\hat{r}'_{k,j}| \right\|\right] \\
& + \varepsilon \begin{cases} \|A\| \|\hat{x}_{sk+s}\| + (2+2N')\|A\| \left\| |\hat{\mathcal{Y}}_k| \cdot |\hat{x}'_{k,s}| \right\| + N' \left\| |\hat{\mathcal{Y}}_k| \cdot |\hat{r}'_{k,s}| \right\|, & j = s \\ 0, & \text{o.w.} \end{cases}
\end{aligned}
$$

where $N' = \max(N, 2s+1)$.

A computable bound

- In each iteration, update error estimate d_i $(i \equiv sk + j)$ by:

$O(ns^2)$ flops per s iterations; ≤1 reduction per s iterations to compute $\left(|\hat{\mathcal{Y}}_k|^T |\hat{\mathcal{Y}}_k|\right)$

$$
\begin{aligned}
d_i \equiv\ & d_{i-1} \\
& + \varepsilon\left[(4+N')\left(\|A\| \left\| |\hat{\mathcal{Y}}_k| \cdot |\hat{x}'_{k,j}| \right\| + \left\| |\hat{\mathcal{Y}}_k| \cdot |\mathcal{B}_k| \cdot |\hat{x}'_{k,j}| \right\|\right) + \left\| |\hat{\mathcal{Y}}_k| \cdot |\hat{r}'_{k,j}| \right\|\right] \\
& + \varepsilon \begin{cases} \|A\| \|\hat{x}_{sk+s}\| + (2+2N') \|A\| \left\| |\hat{\mathcal{Y}}_k| \cdot |\hat{x}'_{k,s}| \right\| + N' \left\| |\hat{\mathcal{Y}}_k| \cdot |\hat{r}'_{k,s}| \right\|, & j = s \\ 0, & \text{o.w.} \end{cases}
\end{aligned}
$$

where $N' = \max(N, 2s+1)$.

A computable bound

- In each iteration, update error estimate d_i $(i \equiv sk + j)$ by:

$O(s^2)$ flops per s iterations; no communication

$$
\begin{aligned}
d_i \equiv\ & d_{i-1} \\
& + \varepsilon\left[(4+N')\left(\|A\|\,\||\hat{\mathcal{Y}}_k|\cdot|\hat{x}'_{k,j}|\| + \||\hat{\mathcal{Y}}_k|\cdot|\mathcal{B}_k|\cdot|\hat{x}'_{k,j}|\|\right) + \||\hat{\mathcal{Y}}_k|\cdot|\hat{r}'_{k,j}|\|\right] \\
& + \varepsilon \begin{cases} \|A\|\,\|\hat{x}_{sk+s}\| + (2+2N')\|A\|\,\||\hat{\mathcal{Y}}_k|\cdot|\hat{x}'_{k,s}|\| + N'\||\hat{\mathcal{Y}}_k|\cdot|\hat{r}'_{k,s}|\|, & j = s \\ 0, & \text{o.w.} \end{cases}
\end{aligned}
$$

where $N' = \max(N, 2s+1)$.

A computable bound

- In each iteration, update error estimate d_i $(i \equiv sk + j)$ by:

Communication only increased by *at most* factor of 2

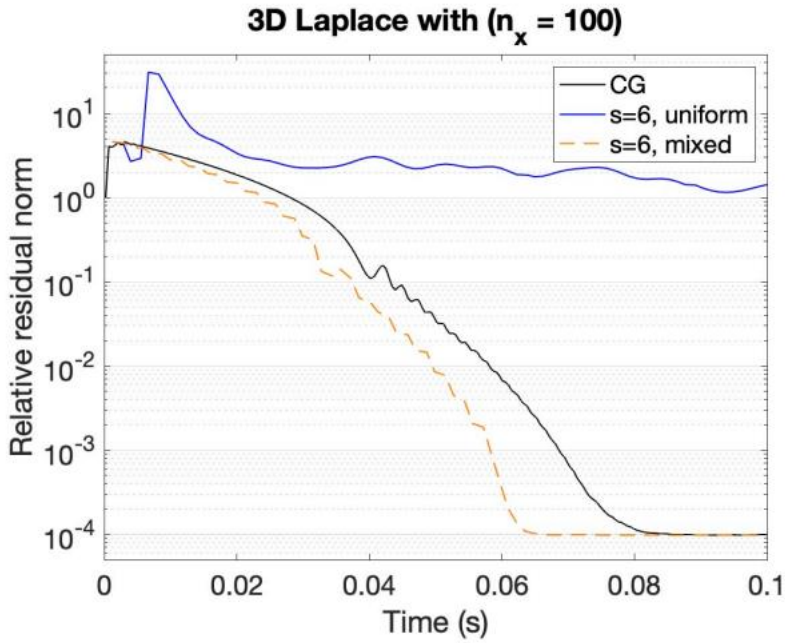
$$
\begin{aligned}
d_i \equiv\ & d_{i-1} \\
& + \varepsilon\left[(4+N')\left(\|A\|\,\left\||\hat{\mathcal{Y}}_k|\cdot|\hat{x}'_{k,j}|\right\| + \left\||\hat{\mathcal{Y}}_k|\cdot|\mathcal{B}_k|\cdot|\hat{x}'_{k,j}|\right\|\right) + \left\||\hat{\mathcal{Y}}_k|\cdot|\hat{r}'_{k,j}|\right\|\right] \\
& + \varepsilon \begin{cases} \|A\|\,\|\hat{x}_{sk+s}\| + (2+2N')\|A\|\,\left\||\hat{\mathcal{Y}}_k|\cdot|\hat{x}'_{k,s}|\right\| + N'\left\||\hat{\mathcal{Y}}_k|\cdot|\hat{r}'_{k,s}|\right\|, & j = s \\ 0, & \text{o.w.} \end{cases}
\end{aligned}
$$

where $N' = \max(N, 2s+1)$.

Laplace problem with RR (single)

$\|b\|_2 = 1$, equal entries

[Yamazaki, C., Kelley, 2022]



$\approx 1.2 \times$ speedup

6 NVIDIA 100 GPUs, single working precision

Matrices from SuiteSparse (single)

[Yamazaki, C., Kelley, 2022]

Name	Type	n	nnz/n
G3_circuit	Circuit Design	1,585,473	4.8
af_shell7	Semiconductor Design	504,855	34.8
parabolic_fem	CFD	525,825	7.0

	CG		s = 2	s = 3	s = 4	s = 5
G3_circuit	1.62 (3196)	uniform	1.72 (4359)	22.59 (66856)	--	--
		mixed	1.39 (3398)	1.40 (3329)	1.71 (3515)	2.80 (8155)
af_shell7	0.25 (504)	uniform	0.20 (504)	0.19 (503)	0.27 (816)	0.23 (522)
		mixed	0.19 (501)	0.19 (503)	0.20 (504)	0.22 (506)
parabolic_fem	0.28 (554)	uniform	0.20 (552)	0.21 (555)	0.22 (562)	0.45 (1060)
		mixed	0.22 (500)	0.21 (555)	0.22 (550)	0.25 (550)

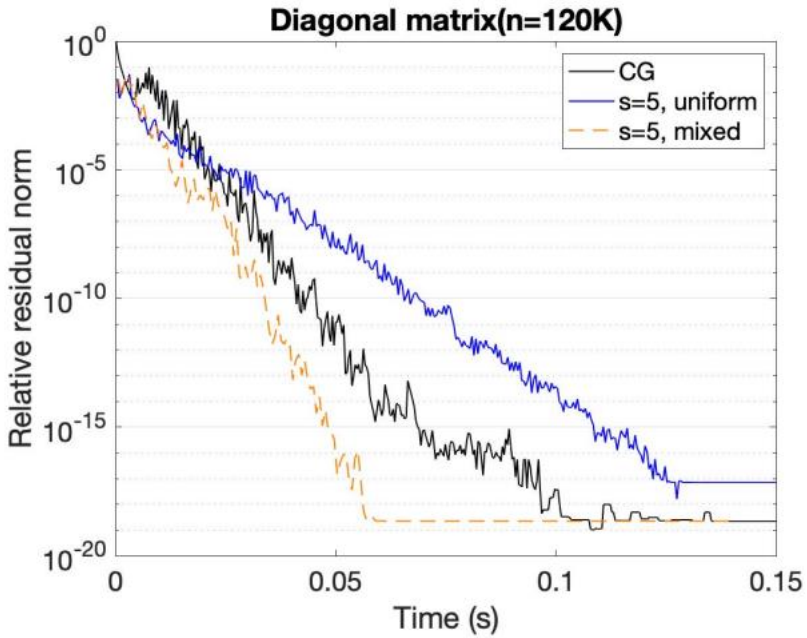
3 NVIDIA 100 GPUs, single working precision

Diagonal Problem with RR (double)

$\lambda_1 = 10^{-3},\ \lambda_n = 10^2,\ \lambda_i = \lambda_1 + (i-1)/(n-1)(\lambda_n - \lambda_1)\rho^{n-i}$ [Yamazaki, C., Kelley, 2022]

$n = 120{,}000,\ \ \rho = 0.65$ (eigenvalues accumulated to the left)

$\|b\|_2 = 1$, equal entries



$\approx 1.8 \times$ speedup

6 NVIDIA 100 GPUs, double working precision

Diagonal Problems (double)

[Yamazaki, C., Kelley, 2022]

$\lambda_n = 10^2,\ \lambda_i = \lambda_1 + (i-1)/(n-1)(\lambda_n - \lambda_1)\rho^{n-i}$

$n = 120{,}000,\ \ \rho = 0.65$ (eigenvalues accumulated to the left)

$\|b\|_2 = 1$, equal entries

	CG		s = 2	s = 3	s = 4	s = 5
$\lambda_1 = 10^{-2}$	.041 (113)	uniform	.024 (125)	.021 (155)	.023 (218)	.030 (334)
		mixed	.023 (111)	.022 (136)	.026 (174)	.028 (194)
$\lambda_1 = 10^{-3}$	.087 (186)	uniform	.058 (257)	.057 (341)	--	--
		mixed	.062 (241)	.057 (281)	.059 (319)	.064 (329)
$\lambda_1 = 10^{-4}$	.121 (336)	uniform	--	--	--	--
		mixed	.083 (410)	.073 (459)	.090 (628)	.091 (632)

3 NVIDIA 100 GPUs, double working precision

Takeaway

- Mixed precision + residual replacement can make s-step CG more reliable
  - Most beneficial for ill-conditioned matrices
- Depending on the setting, overhead can be minimal

- Is using better polynomial bases enough?
- Can we develop a way to adaptively "turn on" mixed precision (and/or RR) in s-step CG?
- Mixed precision strategies for other synchronization-reducing variants (e.g., pipelined CG?)

Thank you!

carson@karlin.mff.cuni.cz
www.karlin.mff.cuni.cz/~carson