Performance and Stability Tradeoffs in Large-Scale Krylov Subspace Methods

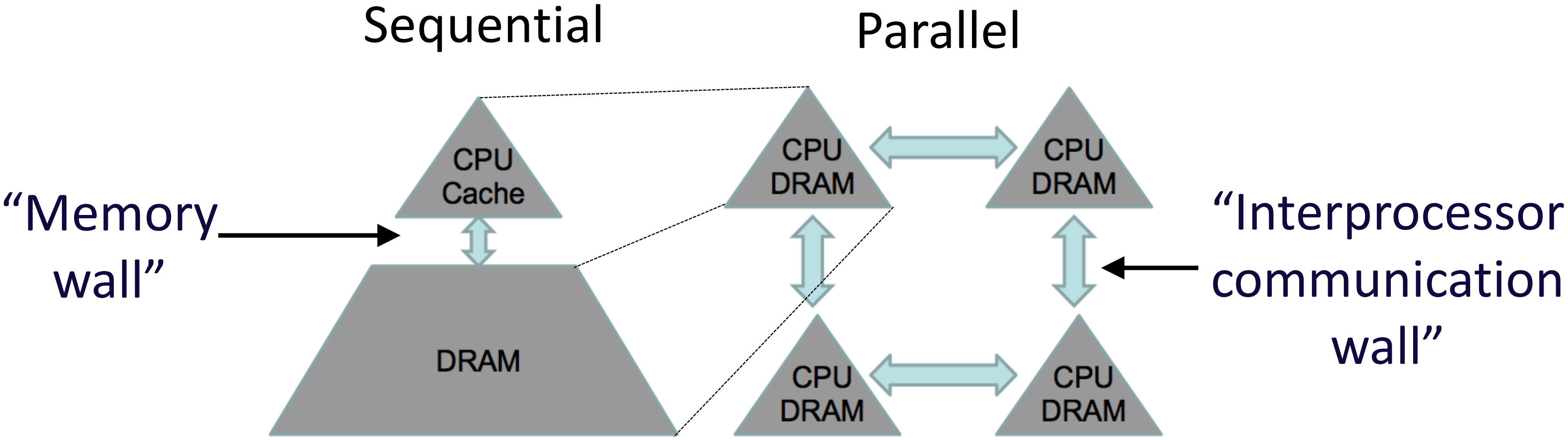
Erin C. Carson

Courant Institute, NYU

Applied Mathematics and Scientific Computing Seminar,
Temple University, November 16, 2016

The cost of an algorithm

- Algorithms have two costs: **computation** and **communication**
  - **Communication : moving data** between levels of memory hierarchy (sequential), between processors (parallel)



- On today's computers, computation is cheap, but communication is expensive, in terms of both **time and energy**
- Barrier to scalability for many scientific codes

Future exascale systems

	Petascale Systems (2009)
System Peak	$2 \cdot 10^{15}$ flops/s
Node Memory Bandwidth	25 GB/s
Total Node Interconnect Bandwidth	3.5 GB/s
Memory Latency	100 ns
Interconnect Latency	1 μs

*Sources: from P. Beckman (ANL), J. Shalf (LBL), and D. Unat (LBL)

- Gaps between communication/computation cost only growing larger in future systems
- **Avoiding communication will be essential for applications at exascale!**

Krylov subspace methods

- A **Krylov Subspace Method** is a projection process onto the Krylov subspace

$$\mathcal{K}_m(A, r_1) = \mathrm{span}\{r_1, Ar_1, A^2 r_1, \dots, A^{m-1} r_1\}$$

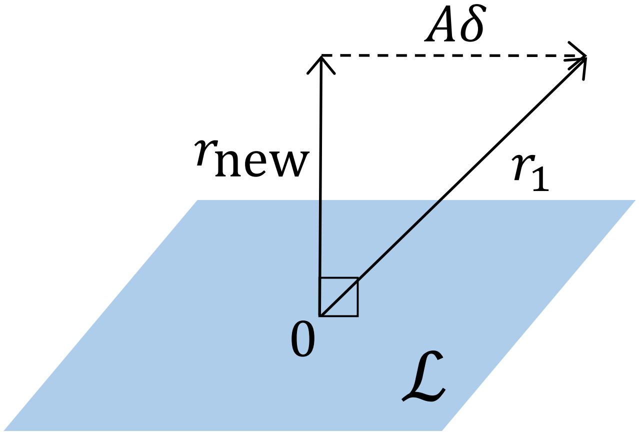
- **Linear systems, eigenvalue problems**, singular value problems, least squares, etc.

- Best for: A large & very sparse, stored implicitly, or only approximation needed

- In each iteration,
  - Add a dimension to the Krylov subspace $\mathcal{K}_m$

    $$\mathcal{K}_1(A, r_1) \subset \mathcal{K}_2(A, r_1) \subset \cdots \subset \mathcal{K}_m(A, r_1)$$

  - Orthogonalize (with respect to some $\mathcal{L}_m$)



- Examples: Lanczos/Conjugate Gradient (CG), Arnoldi/Generalized Minimum Residual (GMRES), Biconjugate Gradient (BICG), BICGSTAB, GKL, LSQR, etc.

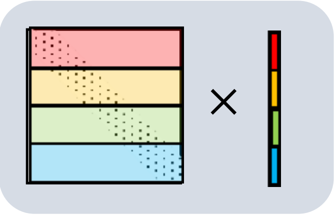
Communication bottleneck

Projection process in terms of communication:

“Add a dimension to $\mathcal{K}_m$”

$\rightarrow$ Sparse matrix-vector multiplication (SpMV)

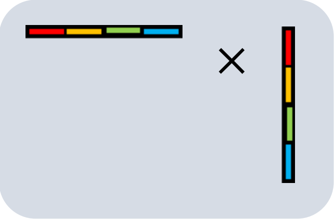
- Parallel: comm. vector entries w/ neighbors
- Sequential: read A/vectors from slow memory

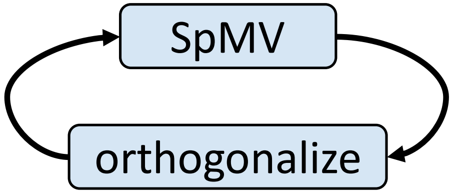


“Orthogonalize (with respect to some $\mathcal{L}_m$)”

$\rightarrow$ Inner products

- Parallel: global reduction (MPI All-Reduce)
- Sequential: multiple reads/writes to slow memory





Dependencies between communication-bound kernels in each iteration limit performance!

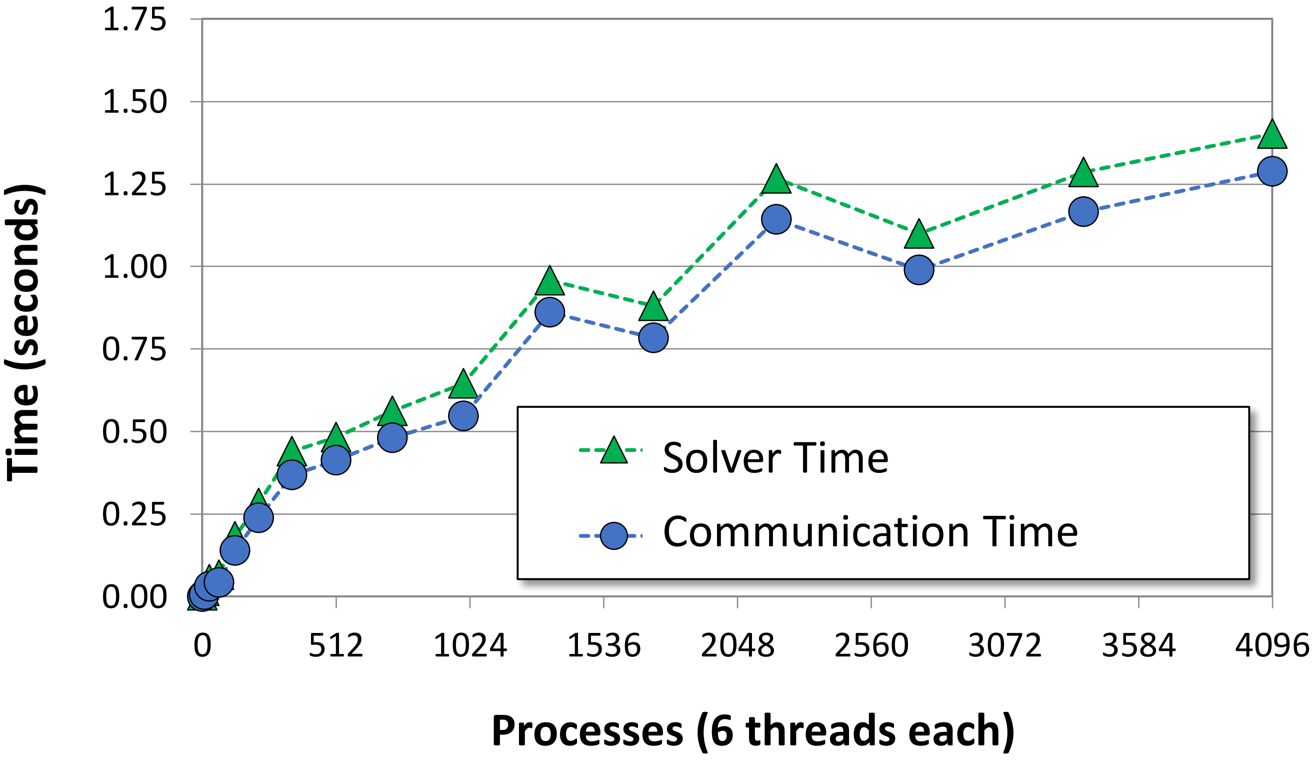
Example: classical conjugate gradient (CG)

```
Given: initial approximation $x_1$ for solving $Ax = b$
Let $p_1 = r_1 = b - Ax_1$
**for** $m = 1, 2, \dots,$ until convergence **do**
    $\alpha_m = \frac{r_m^T r_m}{p_m^T A p_m}$
    $x_{m+1} = x_m + \alpha_m p_m$
    $r_{m+1} = r_m - \alpha_m A p_m$
    $\beta_m = \frac{r_{m+1}^T r_{m+1}}{r_m^T r_m}$
    $p_{m+1} = r_{m+1} + \beta_m p_m$
**end for**
```

SpMVs and inner products require communication in each iteration!

miniGMG multigrid benchmark (Williams et al., 2012) on NERSC's Hopper (Cray XE6)
Variable coefficient Helmholtz operator

Timing for coarse grid solve (BICGSTAB Krylov solver)
Weak scaling: 4^3 points per process (0 slope ideal)



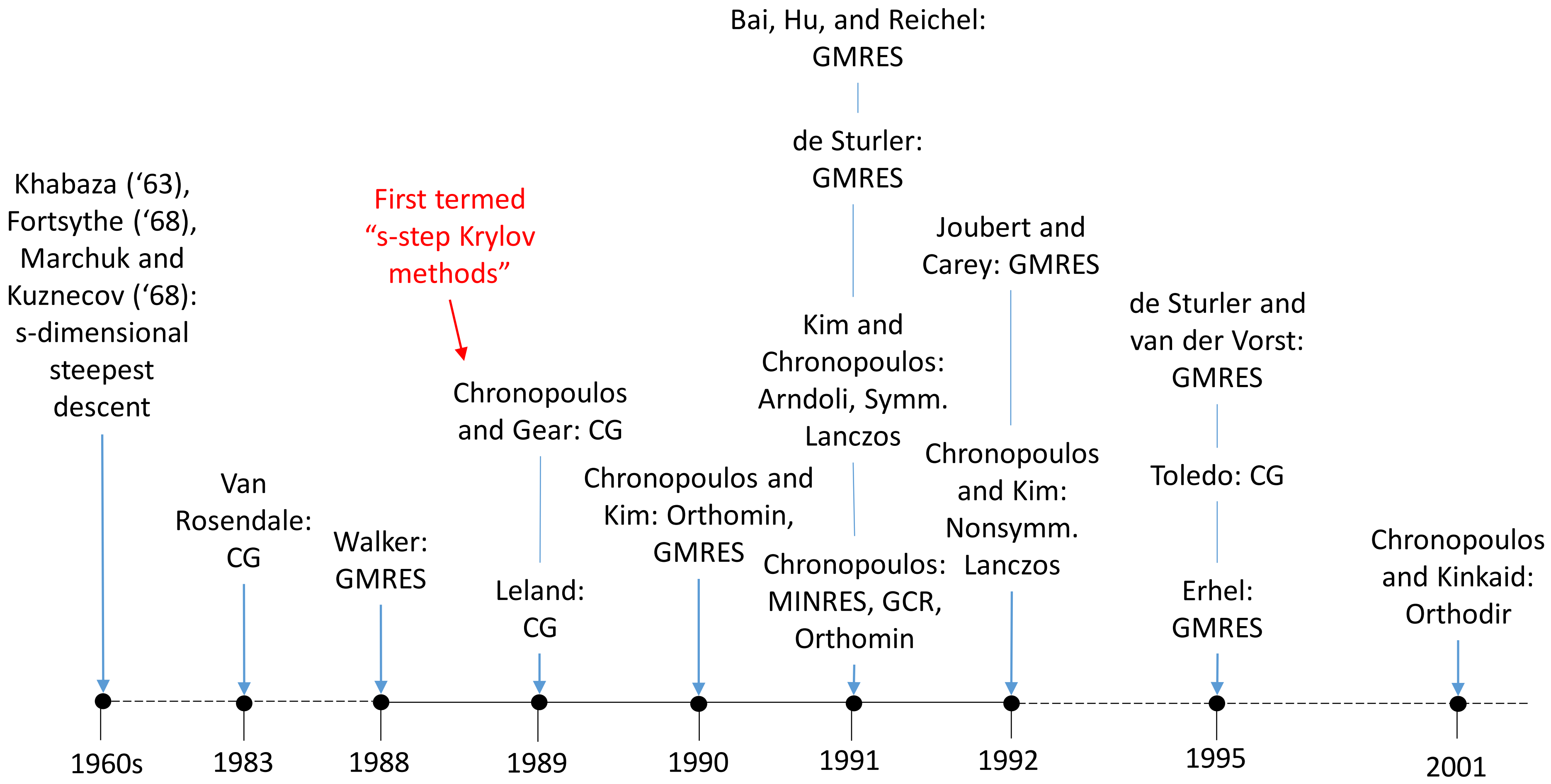
Solver performance and scalability limited by communication!

s-step Krylov subspace methods

- Krylov subspace methods can be reorganized to **reduce communication cost by $\boldsymbol{O(s)}$**
    - "Communication cost": latency in parallel, latency and bandwidth in sequential
    - Compute iteration updates in blocks of size s
    - Communicate once every s iterations instead of every iteration

- Called "s-step" or "communication-avoiding" Krylov subspace methods
- Lots of related work…

History of *s*-step Krylov subspace methods



Khabaza ('63), Fortsythe ('68), Marchuk and Kuznecov ('68): s-dimensional steepest descent

1960s

Van Rosendale: CG

1983

Walker: GMRES

1988

First termed "s-step Krylov methods"

Chronopoulos and Gear: CG

Leland: CG

1989

Chronopoulos and Kim: Orthomin, GMRES

1990

Bai, Hu, and Reichel: GMRES

de Sturler: GMRES

Kim and Chronopoulos: Arndoli, Symm. Lanczos

Chronopoulos: MINRES, GCR, Orthomin

1991

Joubert and Carey: GMRES

Chronopoulos and Kim: Nonsymm. Lanczos

1992

de Sturler and van der Vorst: GMRES

Toledo: CG

Erhel: GMRES

1995

Chronopoulos and Kinkaid: Orthodir

2001

Brief derivation of s-step CG

Main idea: Unroll iteration loop by a factor of s; split iteration loop into outer (k) and inner loop (j). By induction, for $j \in \{1, \dots, s+1\}$

$$x_{sk+j} - x_{sk+1}, \quad r_{sk+j}, \quad p_{sk+j} \quad \in \quad \mathcal{K}_{s+1}(A, p_{sk+1}) + \mathcal{K}_{s}(A, r_{sk+1})$$

Outer loop: Communication step

Expand solution space s dimensions at once

- Compute "basis" matrix $\mathcal{Y}_k$ whose cols. span $\mathcal{K}_{s+1}(A, p_{sk+1}) + \mathcal{K}_{s}(A, r_{sk+1})$
- If A^s is well partitioned, **requires reading A/communicating vectors only once** using matrix powers kernel (Demmel et al.,'07)

Orthogonalize all at once:

- Encode inner products between basis vectors with Gram matrix $\mathcal{G}_k = \mathcal{Y}_k^T \mathcal{Y}_k$ (or compute Tall-Skinny QR)
- Communication cost of **one global reduction**

- Compute basis $\mathcal{P}_k$ for $\mathcal{K}_{s+1}(A, p_{sk+1})$
  - $\mathcal{P}_k$ is matrix of dimension $n \times (s+1)$
  - Let $\underline{\mathcal{P}}_k$ denote the first s columns of $\mathcal{P}_k$
  - $A\underline{\mathcal{P}}_k = \mathcal{P}_k \mathcal{B}_{P,k}$

- Compute basis $\mathcal{R}_k$ for $\mathcal{K}_s(A, r_{sk+1})$
  - $\mathcal{R}_k$ is a matrix of dimension $n \times s$
  - Let $\underline{\mathcal{R}}_k$ denote the first $s-1$ columns of $\mathcal{R}_k$
  - $A\underline{\mathcal{R}}_k = \mathcal{R}_k \mathcal{B}_{R,k}$

$$\mathcal{Y}_k = [\mathcal{P}_k, \mathcal{R}_k] \qquad \underline{\mathcal{Y}}_k = [\underline{\mathcal{P}}_k, 0, \underline{\mathcal{R}}_k, 0] \quad \mathcal{B}_k = \begin{bmatrix} [\mathcal{B}_{P,k}, 0] & 0 \\ 0 & [\mathcal{B}_{R,k}, 0] \end{bmatrix}$$

$$\boxed{A\underline{\mathcal{Y}}_k = \mathcal{Y}_k \mathcal{B}_k}$$

Brief derivation of s-step CG

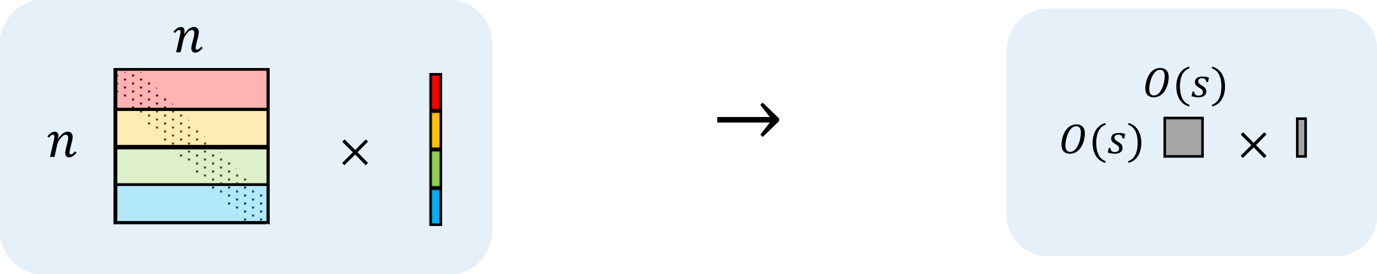
Inner loop: Computation steps, no communication!

Perform s iterations of updates

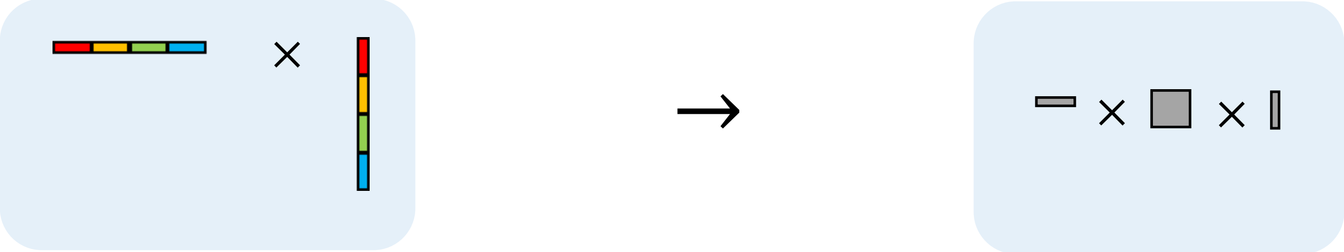
- Using $\mathcal{Y}_k$ and $\mathcal{G}_k = \mathcal{Y}_k^T \mathcal{Y}_k$, this requires **no communication!**
- Represent n-vectors by their $O(s)$ coordinates in $\mathcal{Y}_k$:

$$x_{sk+j+1} - x_{sk+1} = \mathcal{Y}_k x'_{k,j+1}, \quad r_{sk+j+1} = \mathcal{Y}_k r'_{k,j+1}, \quad p_{sk+j+1} = \mathcal{Y}_k p'_{k,j+1}$$

$$Ap_{sk+j} \quad = \quad A\mathcal{Y}_k p'_{k,j} \quad = \quad \mathcal{Y}_k(\mathcal{B}_k p'_{k,j})$$



$$r_{sk+j+1}^T r_{sk+j+1} \quad = \quad r'^T_{k,j+1} \mathcal{Y}_k^T \mathcal{Y}_k r'_{k,j+1} = r'^T_{k,j+1} \mathcal{G}_k r'_{k,j+1}$$



s-step CG

```
Given: initial approximation $x_1$ for solving $Ax = b$
Let $p_1 = r_1 = b - Ax_1$
for k = 0, 1, … , until convergence do
        Compute $\mathcal{Y}_k$ such that $A\underline{\mathcal{Y}}_k = \mathcal{Y}_k\mathcal{B}_k$, compute $\mathcal{G}_k = \mathcal{Y}_k^T\mathcal{Y}_k$
        Let $x'_{k,1} = 0_{2s+1},\ r'_{k,1} = e_{s+2},\ p'_{k,1} = e_1$
        for $j = 1, \dots, s$ do
                $\alpha_{sk+j} = \frac{r'^T_{k,j}\mathcal{G}_k r'_{k,j}}{p'^T_{k,j}\mathcal{G}_k\mathcal{B}_k p'_{k,j}}$
                $x'_{k,j+1} = x'_{k,j} + \alpha_{sk+j}p'_{k,j}$
                $r'_{k,j+1} = r'_{k,j} - \alpha_{sk+j}\mathcal{B}_k p'_{k,j}$
                $\beta_{sk+j} = \frac{r'^T_{k,j+1}\mathcal{G}_k r'_{k,j+1}}{r'^T_{k,j}\mathcal{G}_k r'_{k,j}}$
                $p'_{k,j+1} = r'_{k,j+1} + \beta_{sk+j}p'_{k,j}$
        end for
        $x_{sk+s+1} = \mathcal{Y}_k x'_{k,s+1} + x_{sk+1},\ r_{sk+s+1} = \mathcal{Y}_k r'_{k,s+1},\ p_{sk+s+1} = \mathcal{Y}_k p'_{k,s+1}$
end for
```

via Matrix Powers Kernel

Global reduction to compute $\mathcal{G}_k$

Local computations within inner loop require no communication!

Complexity comparison

Example of parallel (per processor) complexity for s iterations of CG vs. s-step CG for a 2D 9-point stencil:

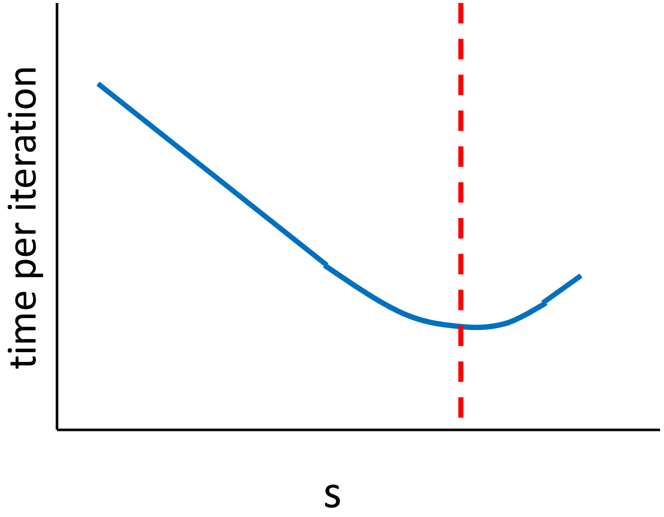
(Assuming each of p processors owns n/p rows of the matrix and $s \leq \sqrt{n/p}$)

	Flops		Words Moved		Messages	
	SpMV	Orth.	SpMV	Orth.	SpMV	Orth.
Classical CG	$\frac{sn}{p}$	$\frac{sn}{p}$	$s\sqrt{n/p}$	$s \log_2 p$	s	$s \log_2 p$
s-step CG	$\frac{sn}{p}$	$\frac{s^2 n}{p}$	$s\sqrt{n/p}$	$s^2 \log_2 p$	1	$\log_2 p$

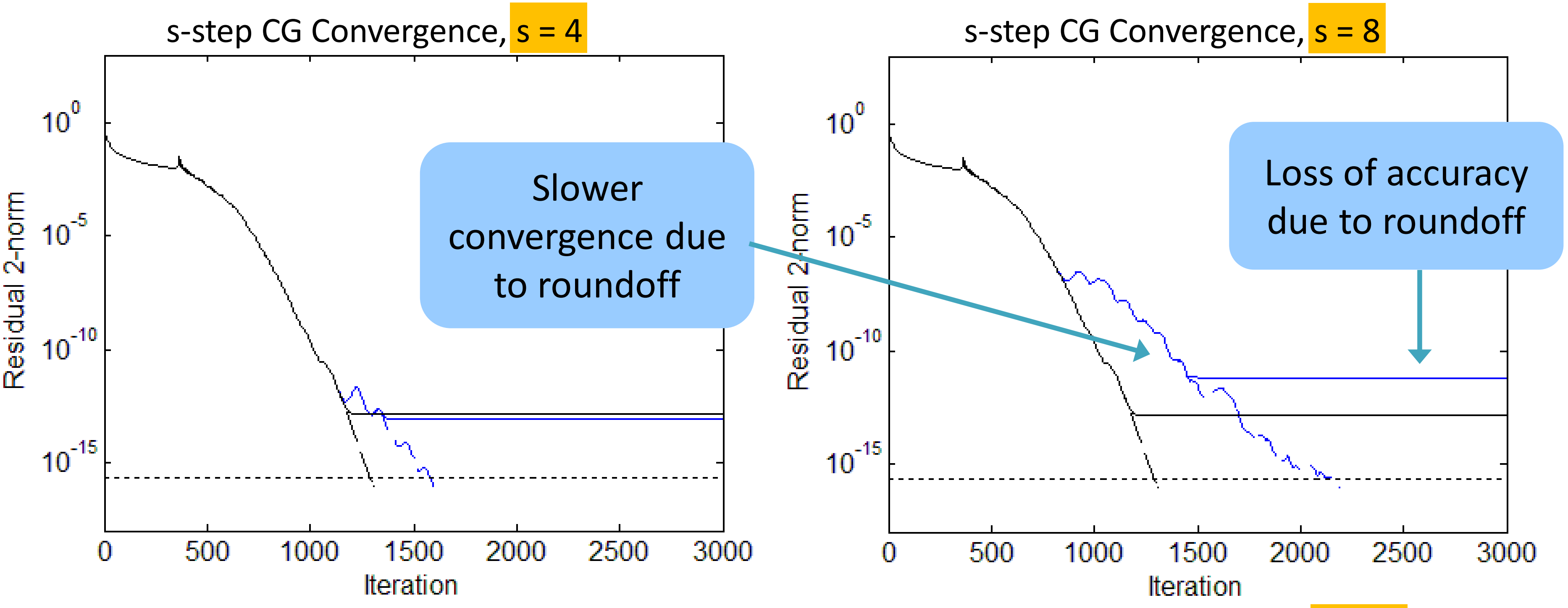
All values in the table meant in the Big-O sense (i.e., lower order terms and constants not included)

Tradeoffs

- Parameter s is limited by machine parameters and matrix sparsity structure
- We can auto-tune to find the best s based on these properties
  - That is, find s that gives the least time per iteration



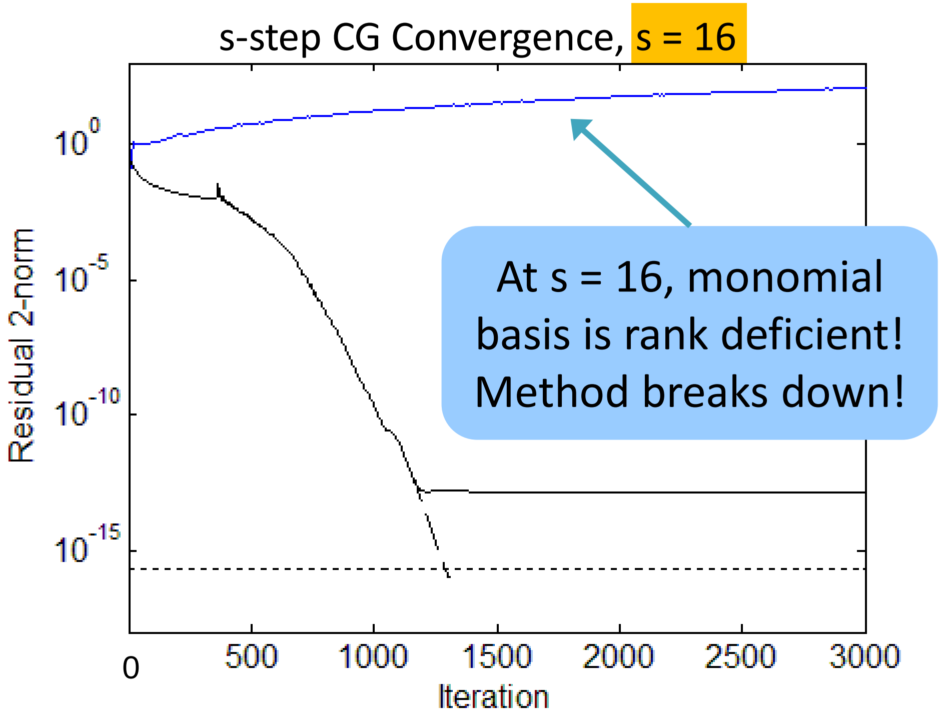
- But s is also limited by numerical properties



Slower convergence due to roundoff

Loss of accuracy due to roundoff

━━	CG true
- - -	CG updated
━━	s-step CG (monomial) true
- - -	s-step CG (monomial) updated



At s = 16, monomial basis is rank deficient! Method breaks down!

Model Problem: 2D Poisson (5-pt stencil),
$n = 512^2, \; N \approx 10^6, \kappa(A) \approx 10^4$
$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$

Behavior in finite precision

- s-step variants are mathematically equivalent to classical methods
- But can behave much differently in finite precision!

- Roundoff errors have two discernable effects:
  1. **Decrease in attainable accuracy** → Tradeoff: increasing blocking factor s past a certain point: **true residual** $\boldsymbol{b - Ax_i}$ stagnates
  2. **Delay of convergence** → Tradeoff: increasing blocking factor s past a certain point: no speedup expected

Runtime = (time/iteration) x (# iterations)

Optimizing for speed and accuracy

- Selecting the best s to use (minimize runtime subject to accuracy constraint) is a hard problem
  - Can tune to minimize time per iteration (based on hardware, matrix structure)
  - But numerical properties (stability, convergence rate) are important too!
    - The “best” s for minimizing time per iteration might not be the best s for minimizing overall runtime, and might give an inaccurate solution

- Goal: Based on finite precision analysis, develop ways to automate parameter choice to improve reliability and usability of s-step Krylov subspace methods
    - Improving s-step basis conditioning
    - Residual replacement
    - Variable s-step methods

Choosing a polynomial basis

- Recall: in each outer loop of s-step CG, we compute bases for some Krylov subspaces, $\mathcal{K}_m(A, v) = \mathrm{span}\{v, Av, \dots, A^{m-1}v\}$

- Simple loop unrolling leads to the choice of monomials $\{v, Av, \dots, A^s v\}$
  - Monomial basis condition number can grow exponentially with s - expected (near) linear dependence of basis vectors for modest s values
  - Recognized early on that this negatively affects convergence (Leland, 1989), (Chronopoulous & Swanson, 1995)

- **Improve basis condition number to improve convergence**: Use different polynomials to compute a basis for the same subspace.

- Two choices based on spectral information that usually lead to well-conditioned bases:
  - **Newton polynomials**
  - **Chebyshev polynomials**

Better conditioned bases

- The Newton basis:

$$\{v, (A-\theta_1)v, (A-\theta_2)(A-\theta_1)v, \dots, (A-\theta_s)\cdots(A-\theta_1)v\}$$

where $\{\theta_1, \dots, \theta_s\}$ are approximate eigenvalues of A, ordered according to Leja ordering

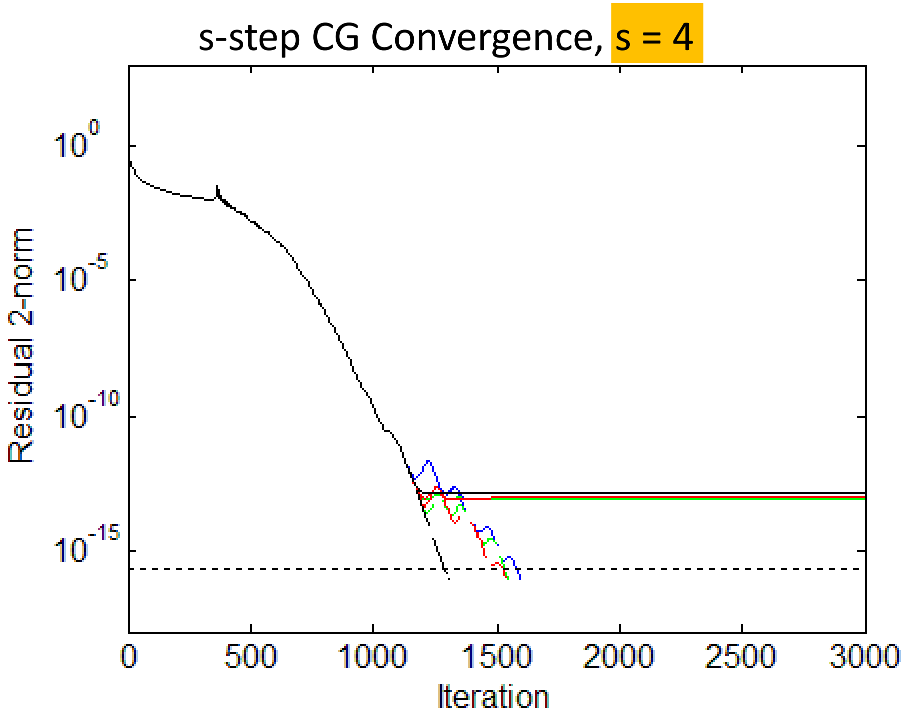
  - In practice: recover Ritz (Petrov) values from the first few iterations, iteratively refine eigenvalue estimates to improve basis
  - Used by many to improve s-step variants: e.g., Bai, Hu, and Reichel (1991), Erhel (1995), Hoemmen (2010)

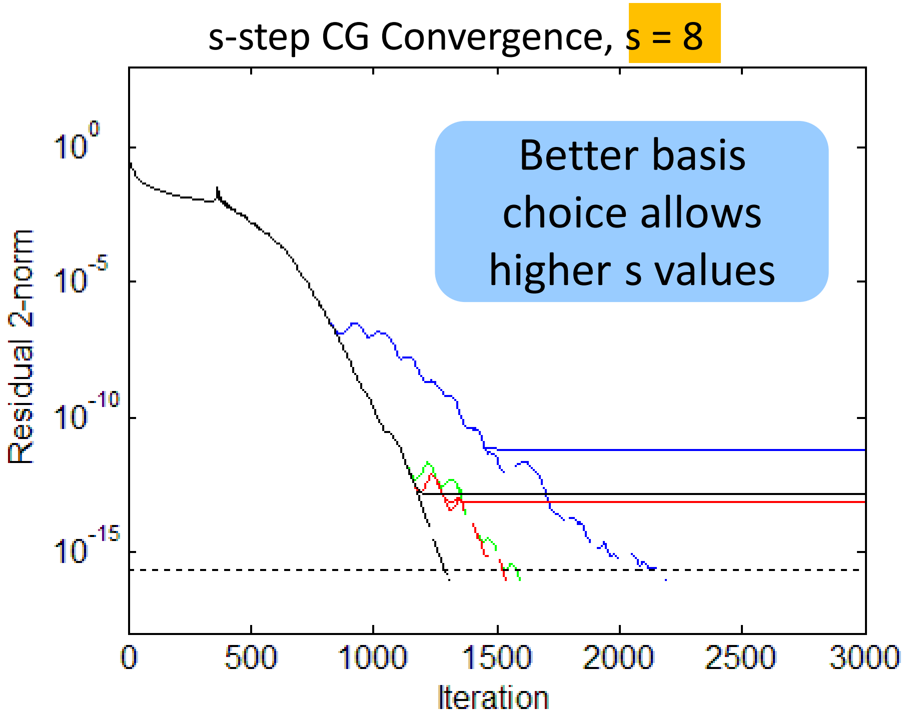
- Chebyshev basis: given ellipse enclosing spectrum of A with foci at $d \pm c$, we can generate the scaled and shifted Chebyshev polynomials as:

$$\tilde{\tau}_j(z) = \left(\tau_j\left(\frac{d-z}{c}\right)\right) \Big/ \left(\tau_j\left(\frac{d}{c}\right)\right)$$

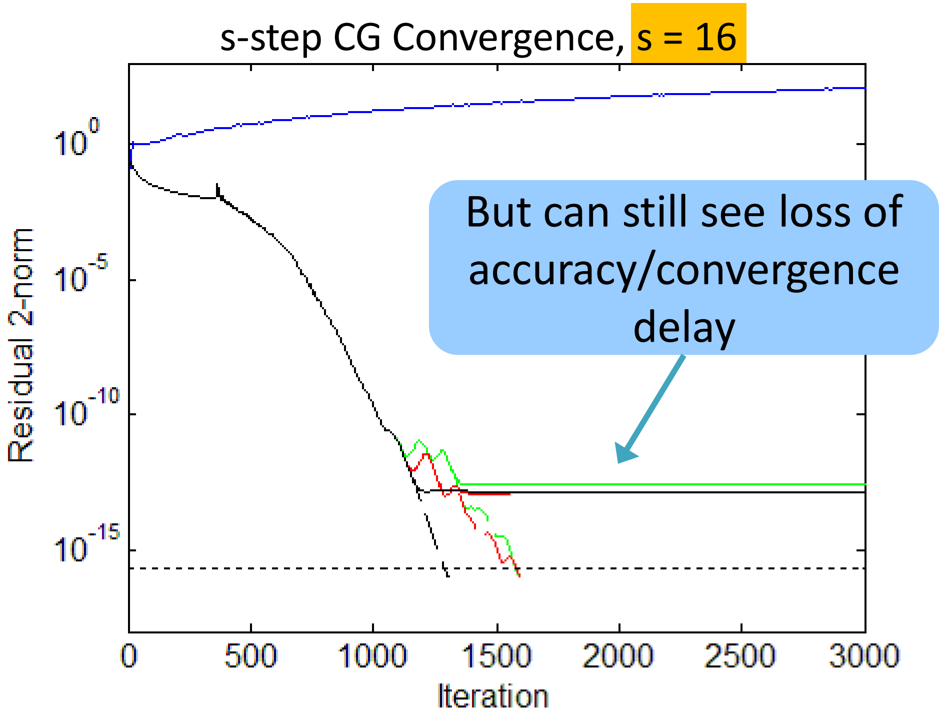
where $\{\tau_j\}_{j\geq 0}$ are the Chebyshev polynomials of the first kind

  - In practice: estimate d and c parameters from Ritz values recovered from the first few iterations
  - Used by many to improve s-step variants: e.g., de Sturler (1991), Joubert and Carey (1992), de Sturler and van der Vorst (1995)





Better basis choice allows higher s values



But can still see loss of accuracy/convergence delay

- ━ CG true
- ╌ CG updated
- ━ s-step CG (monomial) true
- ╌ s-step CG (monomial) updated
- ━ **s-step CG (Newton) true**
- ╌ **s-step CG (Newton) updated**
- ━ **s-step CG (Chebyshev) true**
- ╌ **s-step CG (Chebyshev) updated**

Model Problem: 2D Poisson (5-pt stencil),

$n\ = 512^2,\ N \approx 10^6, \kappa(A) \approx 10^4$

$b = A(1\sqrt{n} \cdot \text{ones}(n,1))$

Maximum attainable accuracy of CG

- In classical CG, iterates are updated by

$$\hat{x}_{m+1} = \hat{x}_m + \hat{\alpha}_m \hat{p}_m + \xi_{m+1} \quad \text{and} \quad \hat{r}_{m+1} = \hat{r}_m - \hat{\alpha}_m A \hat{p}_m + \eta_{m+1}$$

- Accumulation of rounding errors cause the **true residual, $\boldsymbol{b - A\hat{x}_{m+1}}$**, and the **updated residual, $\boldsymbol{\hat{r}_{m+1}}$**, to deviate

- The size of the true residual:

$$\|b - A\hat{x}_{m+1}\| \leq \|\hat{r}_{m+1}\| + \|b - A\hat{x}_{m+1} - \hat{r}_{m+1}\|$$

  - As $\|\hat{r}_{m+1}\| \to 0$, $\|b - A\hat{x}_{m+1}\|$ depends on $\|b - A\hat{x}_{m+1} - \hat{r}_{m+1}\|$

- Many results on 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).

- Can perform a similar analysis to upper bound the maximum attainable accuracy in finite precision s-step CG

Sources of roundoff error in s-step CG

Computing the s-step Krylov basis:

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

Error in computing s-step basis

Updating coordinate vectors in the inner loop:

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

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

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

Error in updating coefficient vectors

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+j+1} = \hat{\mathcal{Y}}_k \hat{x}'_{k,j+1} + \hat{x}_{sk+1} + \phi_{sk+j+1}$$

$$\hat{r}_{sk+j+1} = \hat{\mathcal{Y}}_k \hat{r}'_{k,j+1} + \psi_{sk+j+1}$$

Error in basis change

Maximum attainable accuracy of s-step CG

- We can write the deviation of the true and updated residuals in terms of these errors:

$$
\begin{aligned}
\delta_{sk+j+1} &\equiv b - A\hat{x}_{sk+j+1} - \hat{r}_{sk+j+1} \\
&= \delta_1 \\
&\quad - \sum_{\ell=0}^{k-1}\left[A\phi_{s\ell+s+1} + \psi_{s\ell+s+1} + \sum_{i=1}^{s}\left[A\hat{\mathcal{Y}}_\ell \xi_{\ell,i+1} + \hat{\mathcal{Y}}_\ell \eta_{\ell,i+1} - \Delta_\ell \hat{q}'_{\ell,i}\right]\right] \\
&\quad - A\phi_{sk+j+1} - \psi_{sk+j+1} - \sum_{i=1}^{j}\left[A\hat{\mathcal{Y}}_k \xi_{k,i+1} + \hat{\mathcal{Y}}_k \eta_{k,i+1} - \Delta_k \hat{q}'_{k,i}\right]
\end{aligned}
$$

- Using standard rounding error results, this allows us to obtain an upper bound on $\left\|\delta_{sk+j+1}\right\|$.

Attainable accuracy of CG versus s-step CG

For CG:

$$\|\delta_{m+1}\| \leq \|\delta_1\| + \varepsilon \sum_{i=1}^{m} (1+N)\|A\|\|\hat{x}_{i+1}\| + \|\hat{r}_{i+1}\|$$

For s-step CG:

$$\left\|\delta_{sk+j+1}\right\| \leq \|\delta_1\| + \varepsilon \boldsymbol{c}\bar{\boldsymbol{\Gamma}}_{\boldsymbol{k}} \sum_{i=1}^{sk+j} (1+N)\|A\|\|\hat{x}_{i+1}\| + \|\hat{r}_{i+1}\|$$

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

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

Residual replacement strategy

- Improve accuracy by replacing **updated residual $\hat{r}_{m+1}$** by the **true residual $b - A\hat{x}_{m+1}$** in certain iterations
  - Related work for classical CG: van der Vorst and Ye (1999)

- Choose when to replace $\hat{r}_{m+1}$ with $b - A\hat{x}_{m+1}$ to meet two constraints:
  1. $\|b - A\hat{x}_{m+1} - \hat{r}_{m+1}\|$ 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 strategy allows **both speed and accuracy!**

Residual replacement for s-step CG

- Use computable bound for $\|b - Ax_{sk+j+1} - r_{sk+j+1}\|$ to update d_{sk+j+1}, an estimate of error in computing r_{sk+j+1}, in each iteration

- Set threshold $\hat{\varepsilon} \approx \sqrt{\varepsilon}$, replace whenever $d_{sk+j+1}/\|r_{sk+j+1}\|$ reaches threshold

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

```
if  $d_{sk+j} \leq \hat{\varepsilon}\|r_{sk+j}\|$  and  $d_{sk+j+1} > \hat{\varepsilon}\|r_{sk+j+1}\|$  and  $d_{sk+j+1} > 1.1 d_{init}$
        $z = z + \mathcal{Y}_k\, x'_{k,j+1} + x_{sk+1}$          group update of approximate solution
        $x_{sk+j+1} = 0$
        $r_{sk+j+1} = b - Az$                             set residual to true residual
        $d_{init} = d_{sk+j+1} = \varepsilon\left((1 + 2N')\|A\|\|z\| + \|r_{sk+j+1}\|\right)$
        $p_{sk+j+1} = \mathcal{Y}_k p'_{k,j+1}$
        break from inner loop and begin new outer loop
end
```

A computable bound

- In each iteration, update error estimate d_{sk+j} by:

$O(s^3)$ flops per s iterations; ≤1 reduction per s iterations to compute $\left(|\hat{\mathcal{Y}}_k|^T|\hat{\mathcal{Y}}_k|\right)$

Estimated only once

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

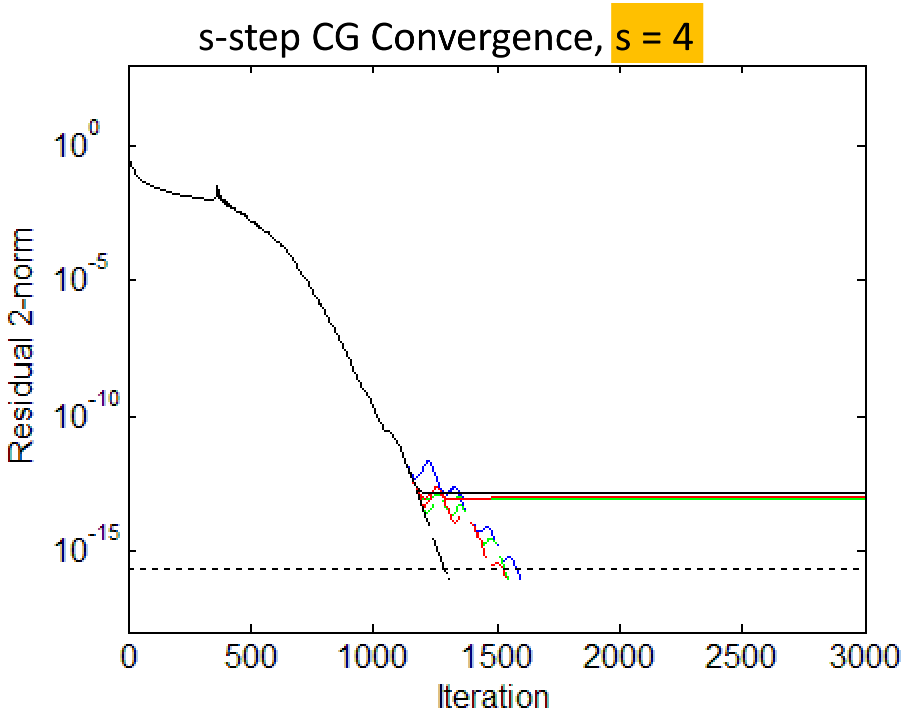
Extra computation all lower order terms, communication only increased by *at most* factor of 2

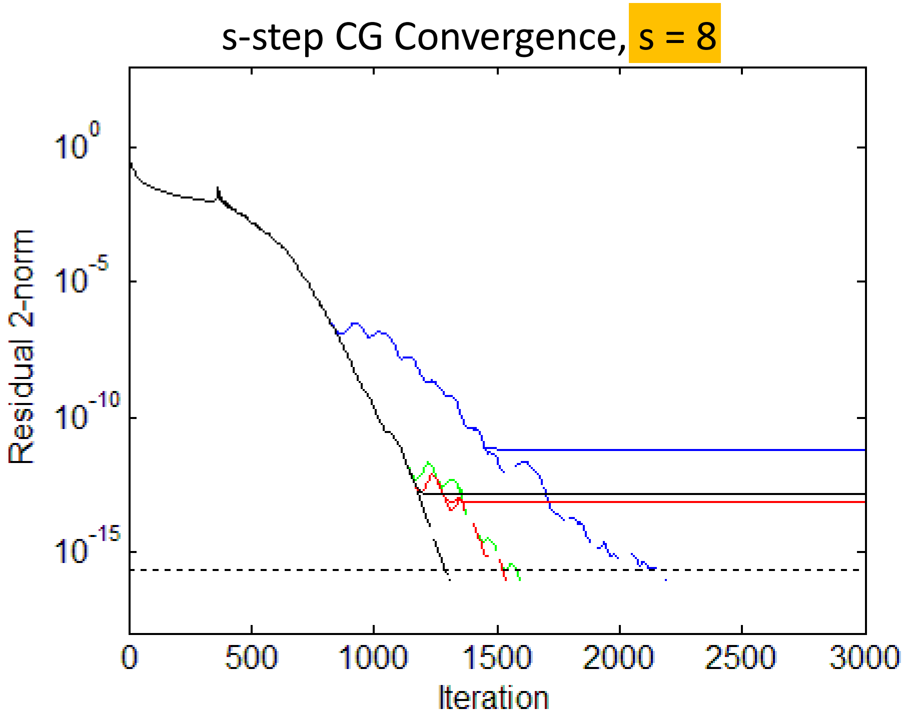
$$d_{sk+j+1} \equiv d_{sk+j}$$

$$+\varepsilon\Big[(4+N')\Big(\|A\|\,\big\|\,|\hat{\mathcal{Y}}_k|\cdot|\hat{x}'_{k,j+1}|\,\big\| + \big\|\,|\hat{\mathcal{Y}}_k|\cdot|\mathcal{B}_k|\cdot|\hat{x}'_{k,j+1}|\,\big\|\Big) + \big\|\,|\hat{\mathcal{Y}}_k|\cdot|\hat{r}'_{k,j+1}|\,\big\|\Big]$$

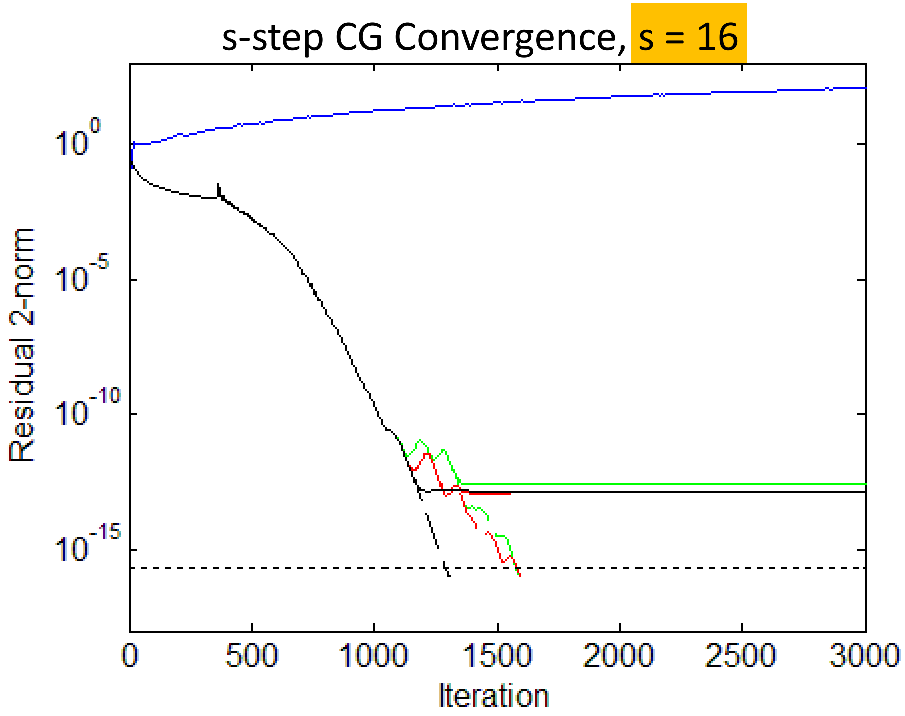
$$+\varepsilon\begin{cases}\|A\|\|\hat{x}_{sk+s+1}\|+(2+2N')\|A\|\big\|\,|\hat{\mathcal{Y}}_k|\cdot|\hat{x}'_{k,s+1}|\,\big\|+N'\big\|\,|\hat{\mathcal{Y}}_k|\cdot|\hat{r}'_{k,s+1}|\,\big\|, & j=s\\ 0, & \text{o.w.}\end{cases}$$

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





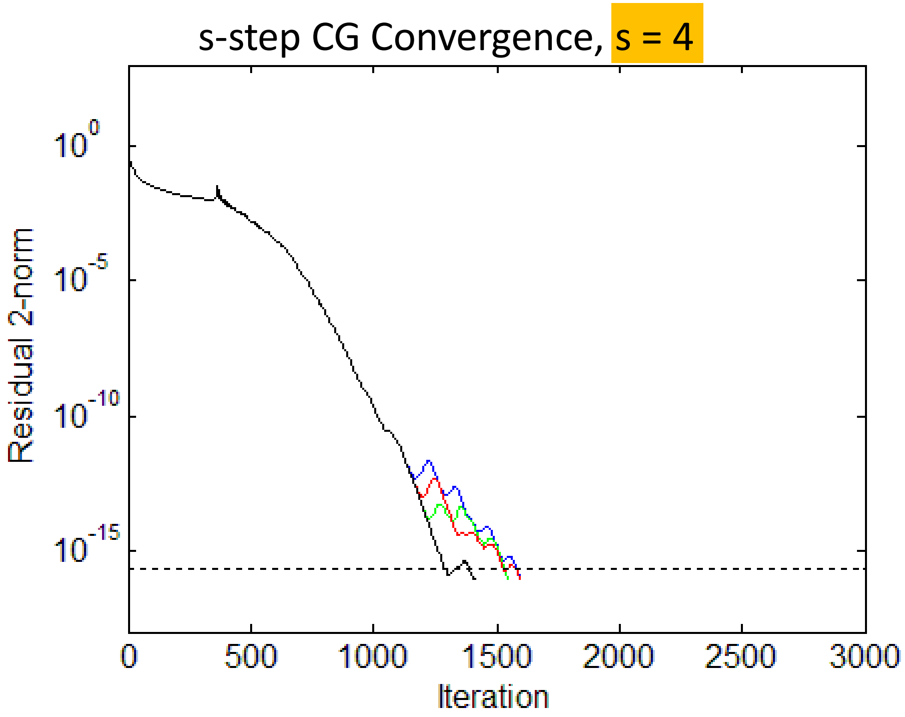
— CG true
- - - CG updated
— s-step CG (monomial) true
- - - s-step CG (monomial) updated
— s-step CG (Newton) true
- - - s-step CG (Newton) updated
— s-step CG (Chebyshev) true
- - - s-step CG (Chebyshev) updated

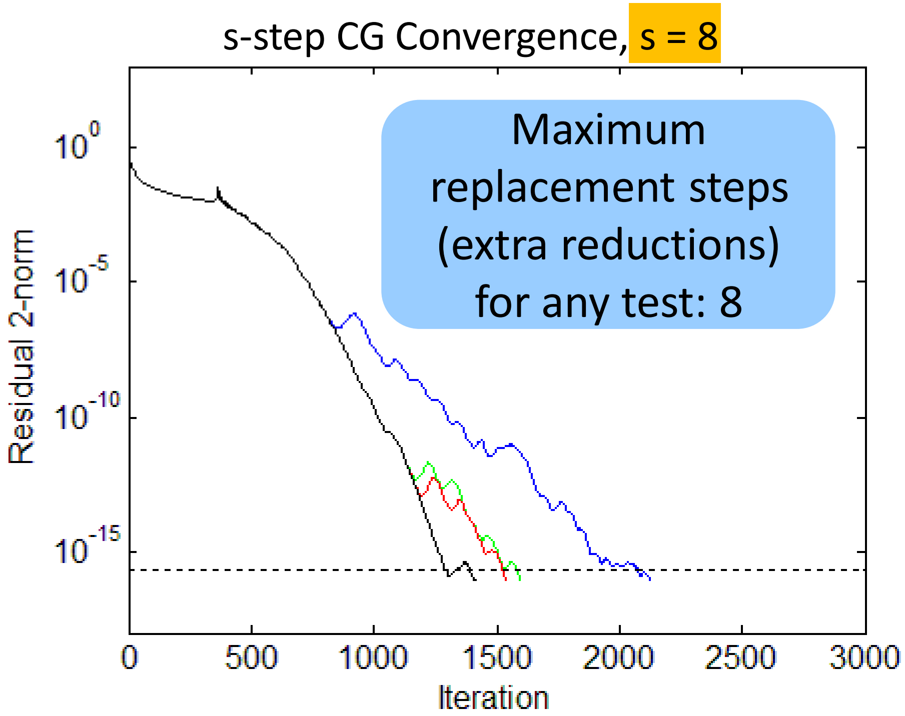


Model Problem: 2D Poisson (5-pt stencil),

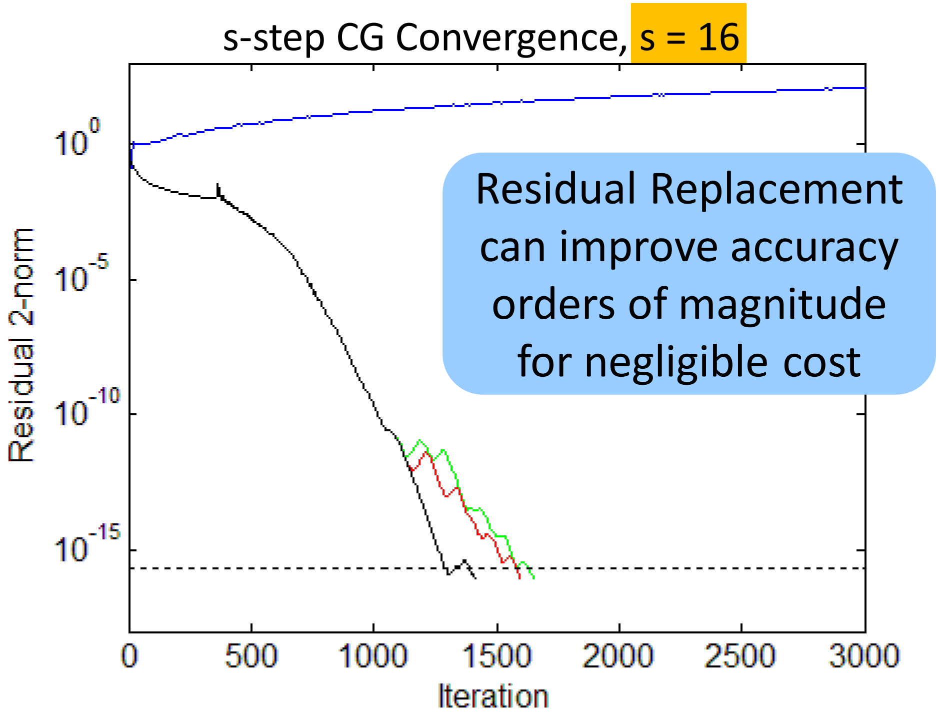
$$n = 512^2, \; N \approx 10^6, \kappa(A) \approx 10^4$$

$$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$$


s-step CG Convergence, s = 4


s-step CG Convergence, s = 8

Maximum replacement steps (extra reductions) for any test: 8


s-step CG Convergence, s = 16

Residual Replacement can improve accuracy orders of magnitude for negligible cost

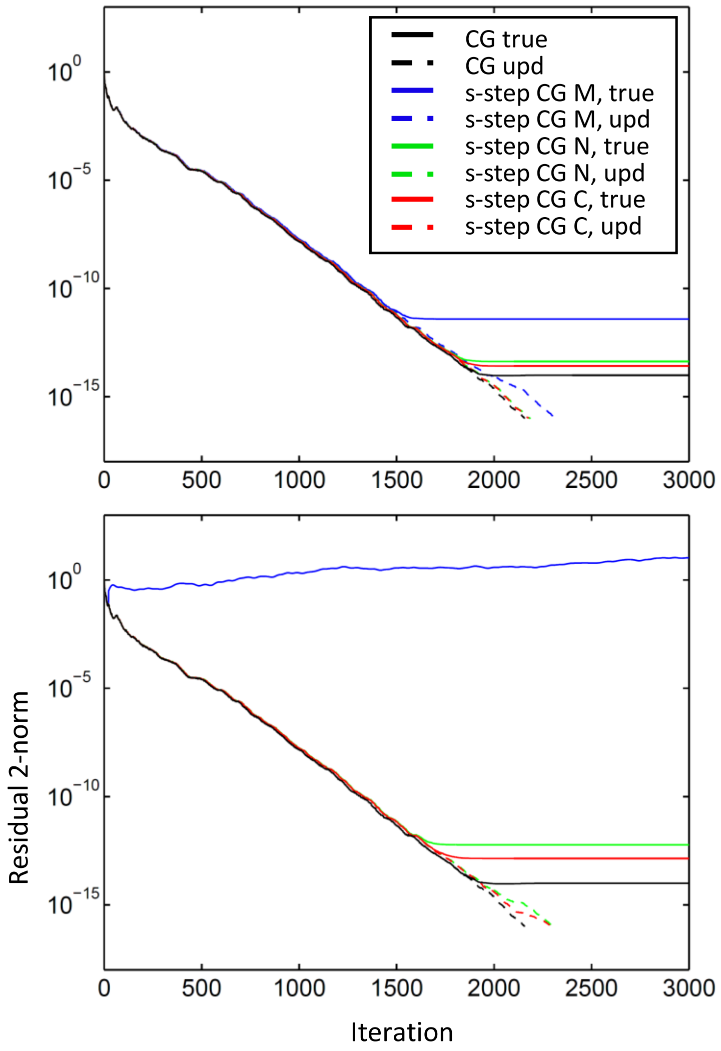
— CG+**RR** true
- - - CG+**RR** updated
— s-step CG+**RR** (monomial) true
- - - s-step CG+**RR** (monomial) updated
— s-step CG+**RR** (Newton) true
- - - s-step CG+**RR** (Newton) updated
— s-step CG+**RR**(Chebyshev) true
- - - s-step CG+**RR**(Chebyshev) updated

Model Problem: 2D Poisson (5-pt stencil),
$n = 512^2,\ N \approx 10^6, \kappa(A) \approx 10^4$
$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$

consph8, FEM/Spheres (from UFSMC)

$n = 8.3 \cdot 10^4$, $N = 6.0 \cdot 10^6$, $\kappa(A) = 9.7 \cdot 10^3$, $\|A\| = 9.7$

Before



Variable s-step CG derivation

- Consider the growth of the relative residual gap caused by errors in outer loop k
- We can approximate an upper bound on this quantity by

$$\frac{\|\delta_{sk+s+1} - \delta_{sk+1}\|}{\|A\|\|x\|} \lesssim c\kappa(A)\Gamma_k \varepsilon \frac{\|\hat{r}_{sk+1}\|}{\|A\|\|x\|}$$

where c is a low-degree polynomial in s

- If our application requires relative accuracy ε^*, we must have

$$\Gamma_k \equiv \|\hat{\mathcal{Y}}_k^+\| \| |\hat{\mathcal{Y}}_k| \| \lesssim \frac{\varepsilon^* \|b\|}{c\varepsilon \|\hat{r}_{sk+1}\|}$$

- In other words, as the method converges (i.e., as $\|\hat{r}_{sk+1}\|$ decreases), we can tolerate more ill-conditioned s-step bases without affecting attainable accuracy
- This naturally leads to a variable s-step approach, where s starts off small and increases as the method converges
  - Analogy to relaxation strategy in “inexact Krylov subspace methods”

Variable s-step CG method

- Input (or tune off-line to find) best s based on speed per iteration; set this as s_{max}
- Run variable s-step CG
    - In each outer loop, stop constructing basis $\hat{\mathcal{Y}}_k$ after $s_k \leq s_{max}$ SpMVs such that

$$\kappa\left(\hat{\mathcal{Y}}_k\right) \leq \frac{\varepsilon^* \|b\|}{\varepsilon \|\hat{r}_{curr}\|}$$

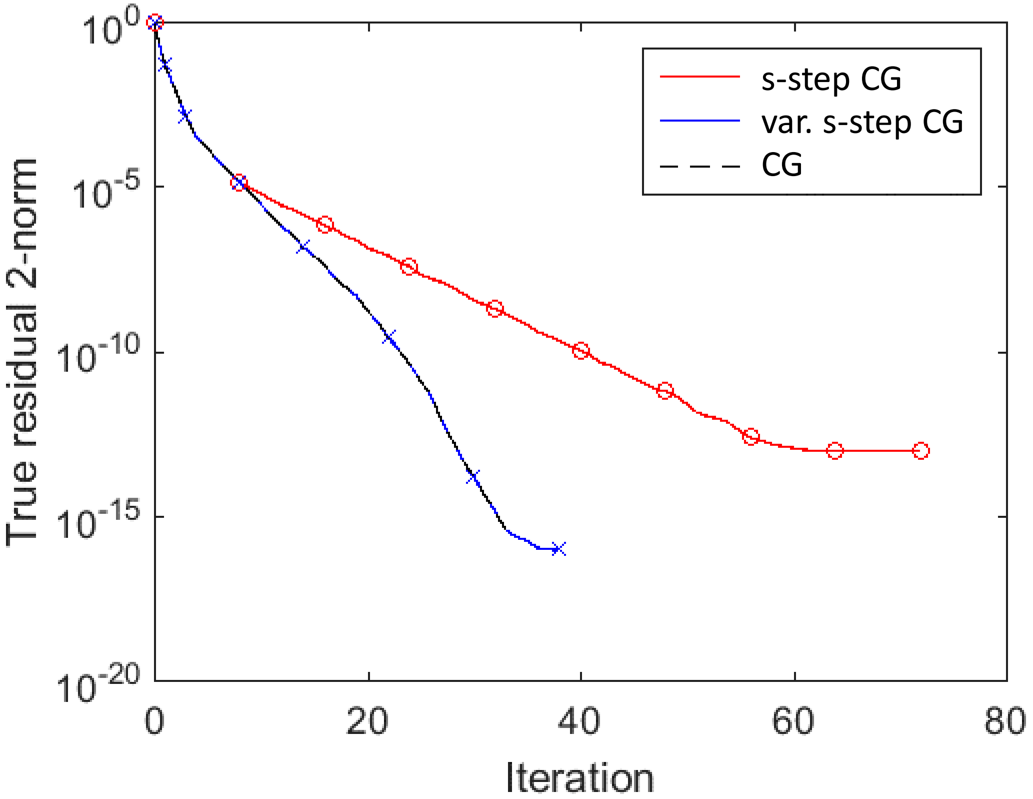
    - Perform s_k inner iteration updates

mesh3e1 (UFSMC)

$n = 289$

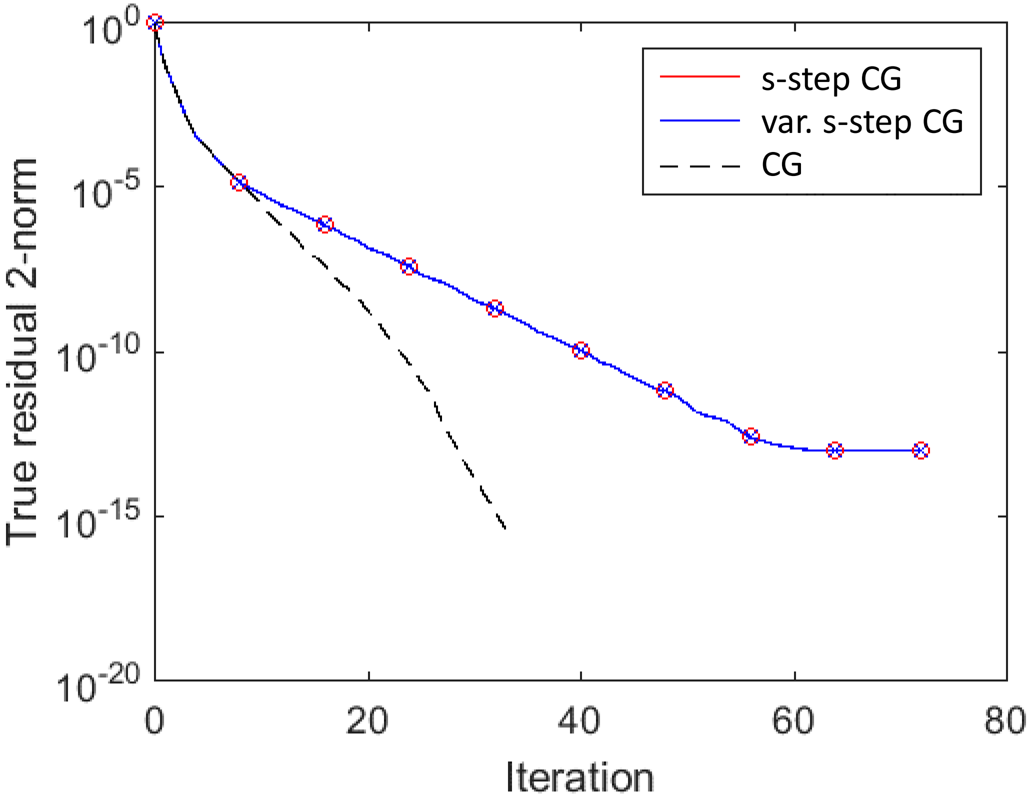
$\kappa(A) \approx 10$

$b_i = 1/\sqrt{n}$ $s = 8$ (monomial basis) $\varepsilon^* = 10^{-14}$



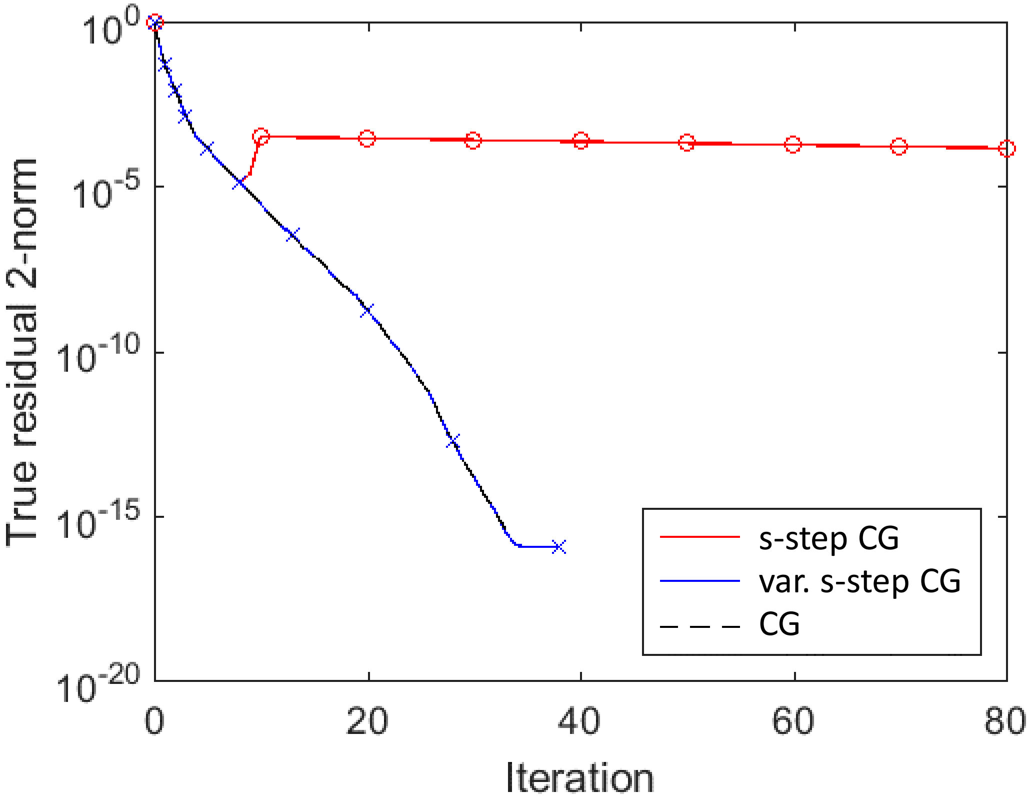
CG		s-step CG		variable s-step CG	
# outer loops	accuracy	# outer loops	accuracy	# outer loops	accuracy
33	4×10^{-16}	9	1×10^{-13}	7	1×10^{-16}

mesh3e1 (UFSMC)
$n = 289$
$\kappa(A) \approx 10$
$b_i = 1/\sqrt{n}$ $s = 8$ (monomial basis) $\varepsilon^* = 10^{-6}$



CG		s-step CG		variable s-step CG	
# outer loops	accuracy	# outer loops	accuracy	# outer loops	accuracy
33	4×10^{-16}	9	1×10^{-13}	9	1×10^{-13}

mesh3e1 (UFSMC)
$n = 289$
$\kappa(A) \approx 10$
$b_i = 1/\sqrt{n}$ $\quad$ $s = 10$ (monomial basis) $\quad$ $\varepsilon^* = 10^{-14}$



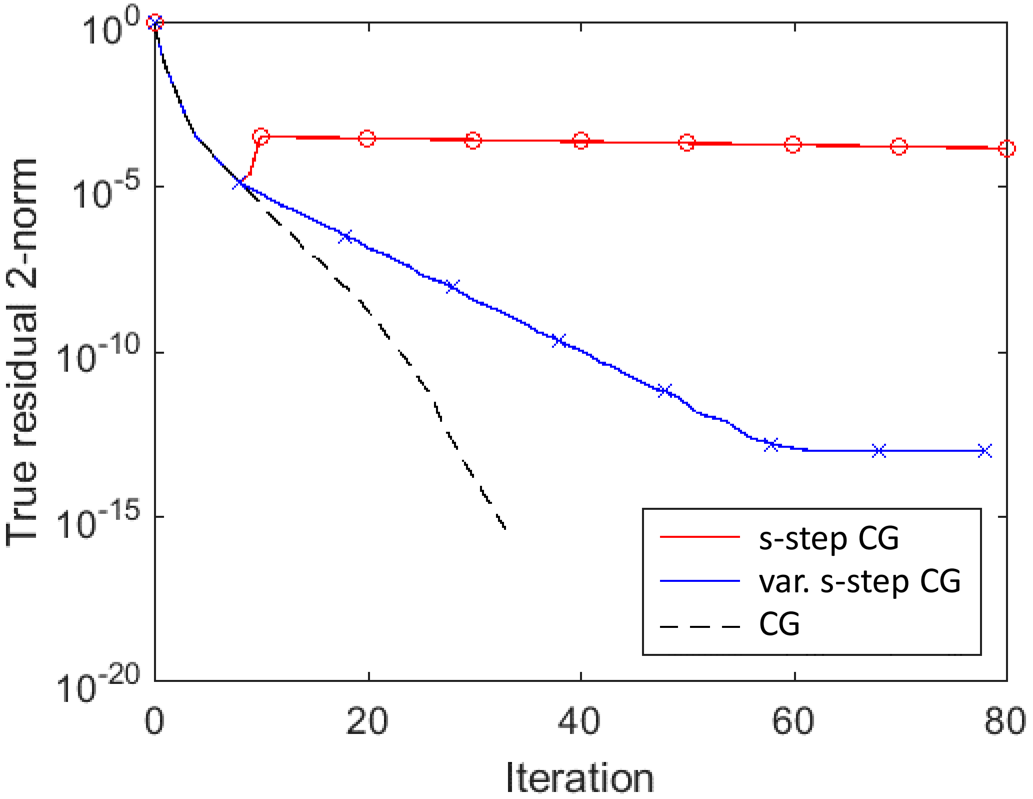
CG		s-step CG		variable s-step CG	
# outer loops	accuracy	# outer loops	accuracy	# outer loops	accuracy
33	4×10^{-16}	–	–	9	1×10^{-16}

mesh3e1 (UFSMC)

$n = 289$

$\kappa(A) \approx 10$

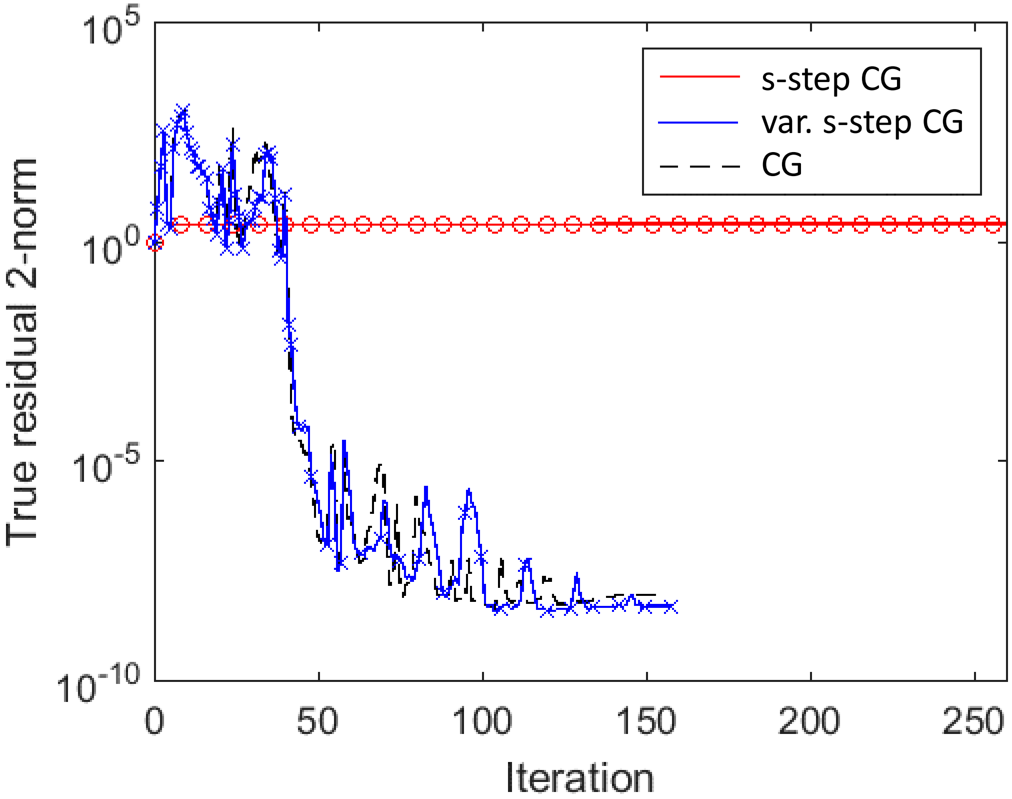
$b_i = 1/\sqrt{n}$ $\quad$ $s = 10$ (monomial basis) $\quad$ $\varepsilon^* = 10^{-6}$



CG		s-step CG		variable s-step CG	
# outer loops	accuracy	# outer loops	accuracy	# outer loops	accuracy
33	4×10^{-16}	–	–	7	1×10^{-13}

ex5 (UFSMC)
$n = 27$
$\kappa(A) \approx 7 \times 10^7$
$b_i = 1/\sqrt{n}$

$s = 10$ (monomial basis) $\varepsilon^* = 10^{-14}$



CG		s-step CG		variable s-step CG	
# outer loops	accuracy	# outer loops	accuracy	# outer loops	accuracy
157	9×10^{-9}	–	–	60	5×10^{-9}

Paige's results for classical Lanczos

- Using bounds on local rounding errors in Lanczos, Paige showed that
  1. The computed Ritz values 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 Ritz values have converged.

Do the same statements hold for s-step Lanczos?

Lanczos analysis

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

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

Classic Lanczos rounding error result of Paige (1976):

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

$$\|\delta\hat{v}_i\|_2 \leq \varepsilon_1\sigma$$

$$\hat{\beta}_{i+1}\left|\hat{v}_i^T\hat{v}_{i+1}\right| \leq 2\varepsilon_0\sigma$$

$$\left|\hat{v}_{i+1}^T\hat{v}_{i+1} - 1\right| \leq \varepsilon_0/2$$

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

where $\sigma \equiv \|A\|_2$, $\theta\sigma \equiv \||A|\|_2$, $\varepsilon_0 = O(\varepsilon n)$, $\varepsilon_1 = O(\varepsilon N\theta)$

For s-step Lanczos:

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

$$\bar{\Gamma}_k \leq \max_{\ell\leq k} \|\mathcal{Y}_\ell^+\| \cdot \||\mathcal{Y}_\ell|\|$$

The amplification term

- Roundoff errors in s-step variant follow same pattern as classical variant, but amplified by factor of $\bar{\Gamma}_k$ or $\bar{\Gamma}_k^2$
    - **Theoretically confirms empirical observations** on importance of basis conditioning (dating back to late ‘80s)

- Using the definition

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

  gives simple, but loose bounds

- What we really need: $\| |\mathcal{Y}| |y'| \| \le \Gamma \|\mathcal{Y}y'\|$ to hold for the computed basis $\mathcal{Y}$ and coordinate vector y' in every bound.

- **Alternate definition of Γ gives tighter bounds**; requires light bookkeeping

- Example: for bounds on $\hat{\beta}_{i+1}\left|\hat{v}_i^T \hat{v}_{i+1}\right|$ and $\left|\hat{v}_{i+1}^T \hat{v}_{i+1} - 1\right|$, we can use the definition

$$\Gamma_{k,j} \equiv \max_{x \in \{\hat{w}'_{k,j}, \hat{u}'_{k,j}, \hat{v}'_{k,j}, \hat{v}'_{k,j-1}\}} \frac{\left\| |\hat{\mathcal{Y}}_k| |x| \right\|}{\|\hat{\mathcal{Y}}_k x\|}$$

Results for s-step Lanczos

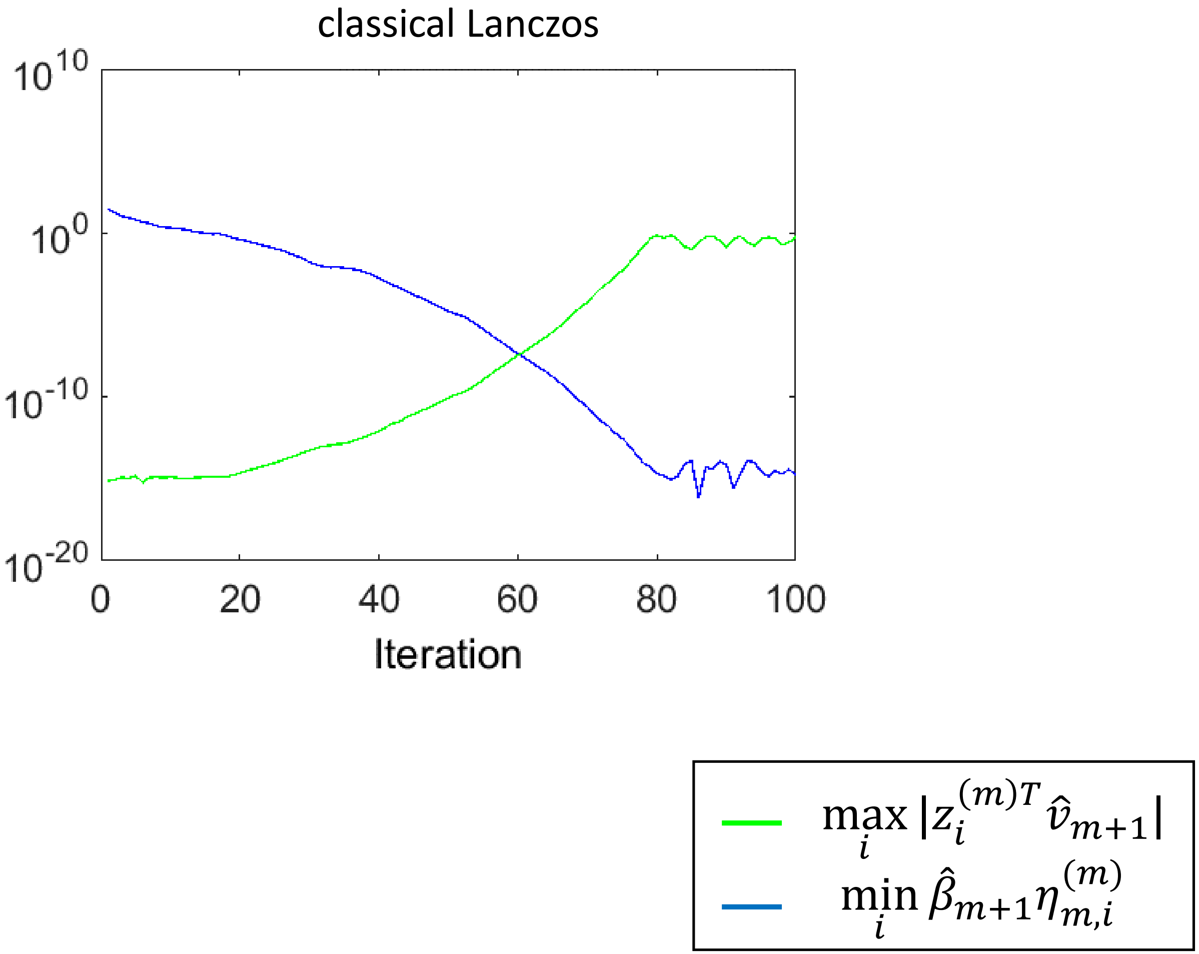
- Back to our question: Do Paige's results, e.g.,

  loss of orthogonality → eigenvalue convergence

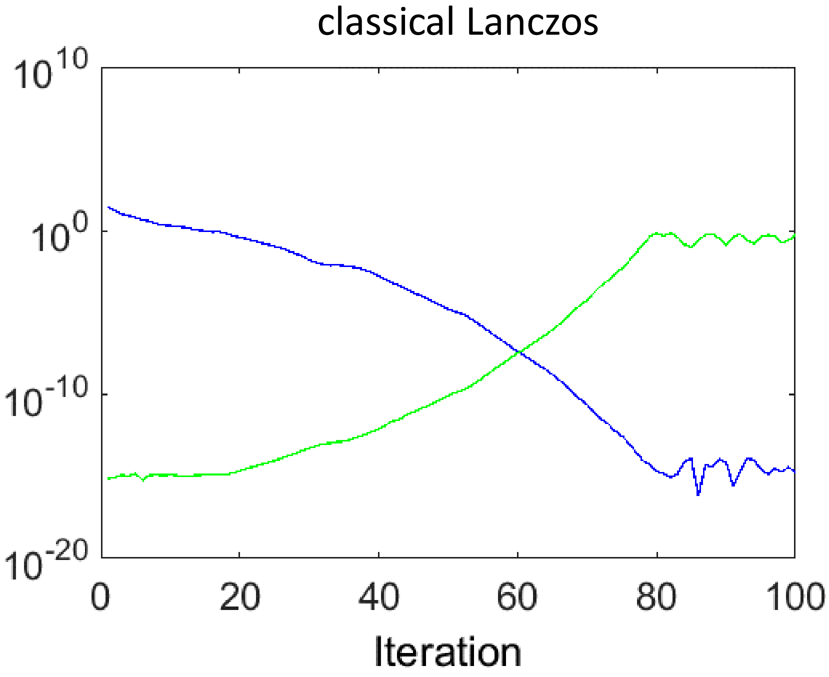
  hold for s-step Lanczos?

- The answer is **YES** …if
  - $\hat{\mathcal{Y}}_\ell$ is numerically full rank for $0 \leq \ell \leq k$ and
  - $\varepsilon_0 \equiv 2\varepsilon(n+11s+15)\,\bar{\Gamma}_k^2 \leq \frac{1}{12}$
    - i.e., $\bar{\Gamma}_k^2 \leq \left(24\varepsilon(n+11s+15)\right)^{-1}$
  - Otherwise, e.g., can lose orthogonality due to computation with rank-deficient basis

Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector


classical Lanczos

Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector


classical Lanczos

🟩 $\max_i |z_i^{(m)T} \hat{v}_{m+1}|$

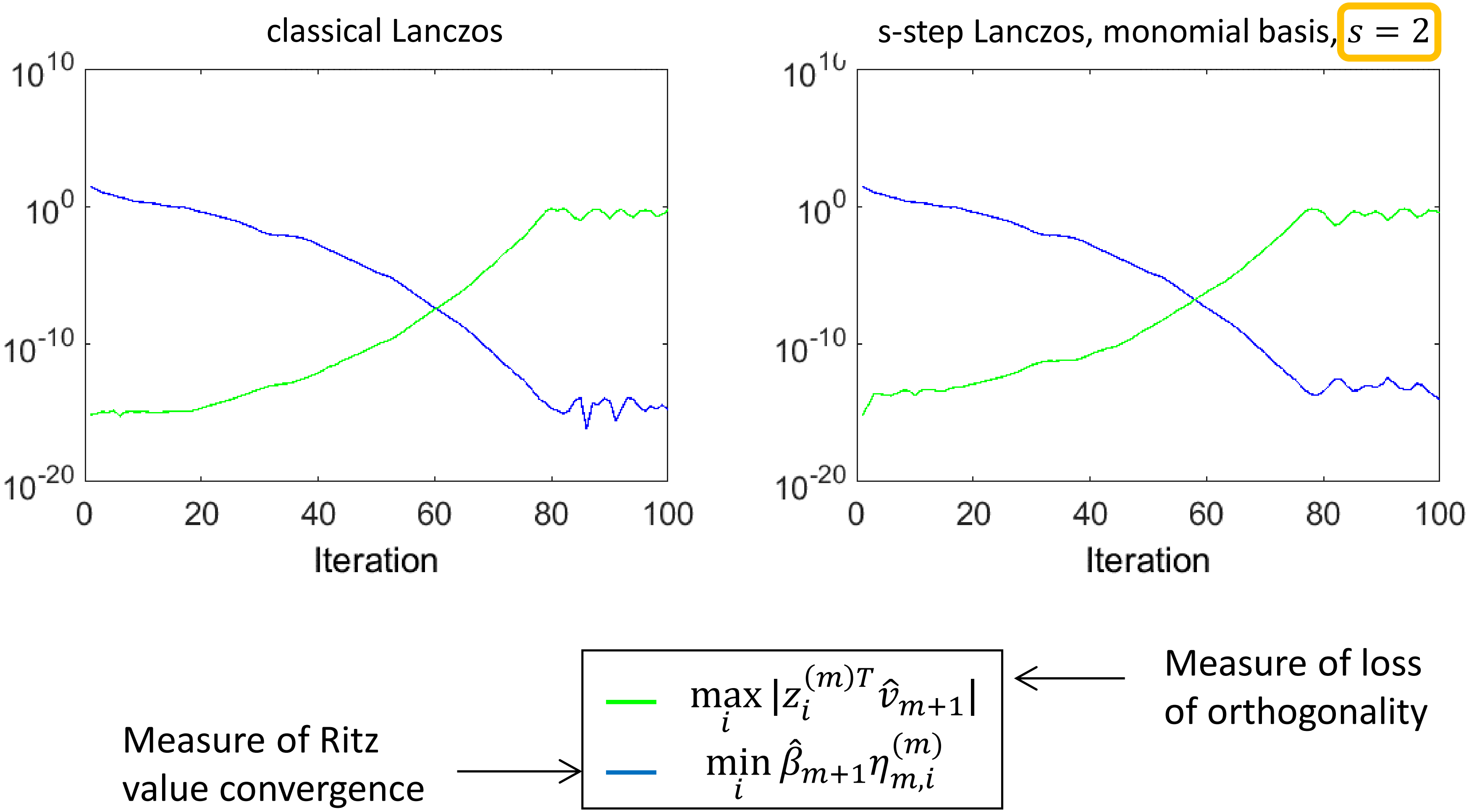
🟦 $\min_i \hat{\beta}_{m+1} \eta_{m,i}^{(m)}$

Measure of loss of orthogonality

Measure of Ritz value convergence

Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector

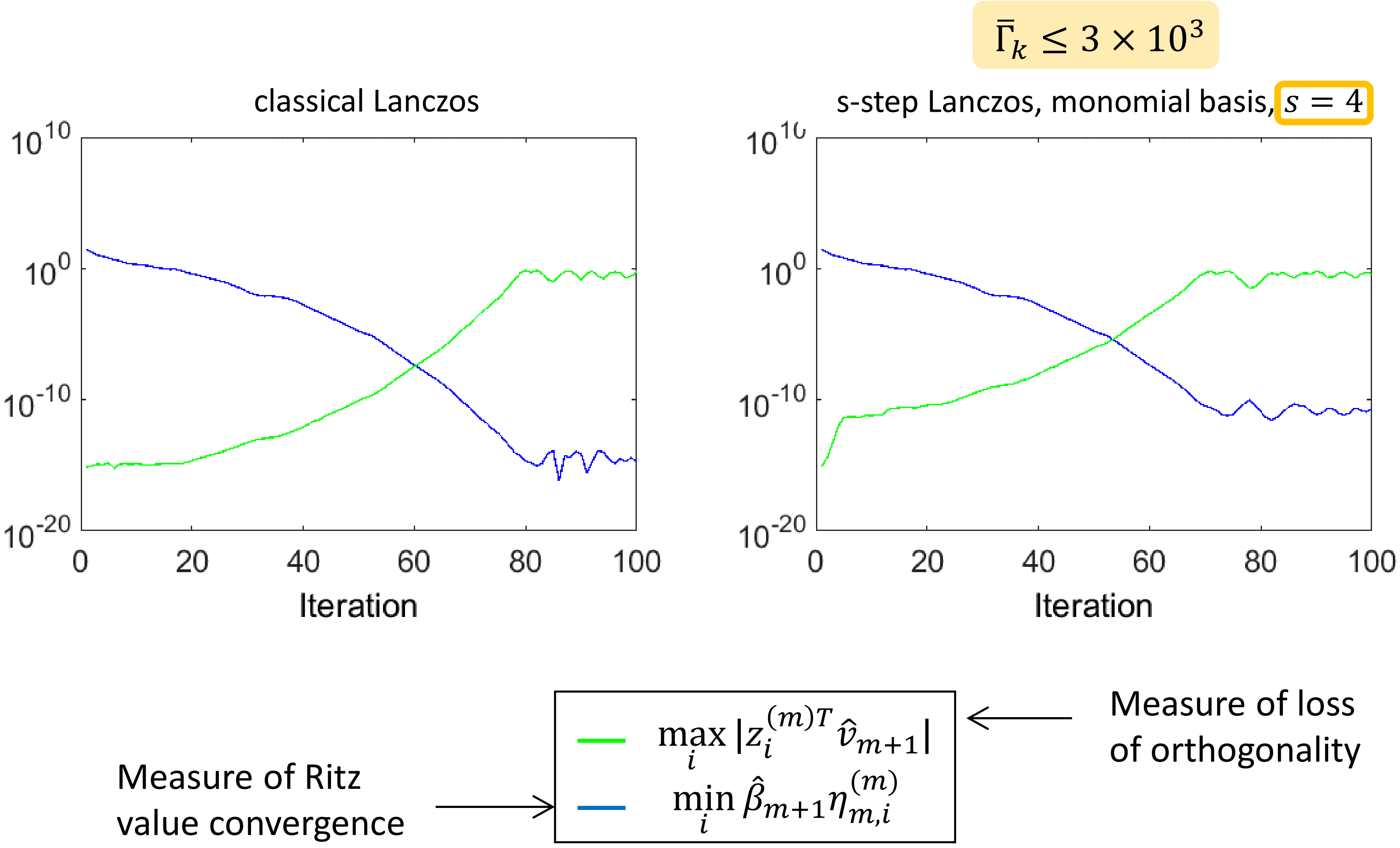
$\bar{\Gamma}_k \leq 7 \times 10^2$



Measure of loss of orthogonality

Measure of Ritz value convergence

Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector



$\bar{\Gamma}_k \leq 3 \times 10^3$

classical Lanczos

s-step Lanczos, monomial basis, $\boxed{s = 4}$

🟩 $\max_i |z_i^{(m)T} \hat{v}_{m+1}|$

🟦 $\min_i \hat{\beta}_{m+1} \eta_{m,i}^{(m)}$

Measure of loss of orthogonality

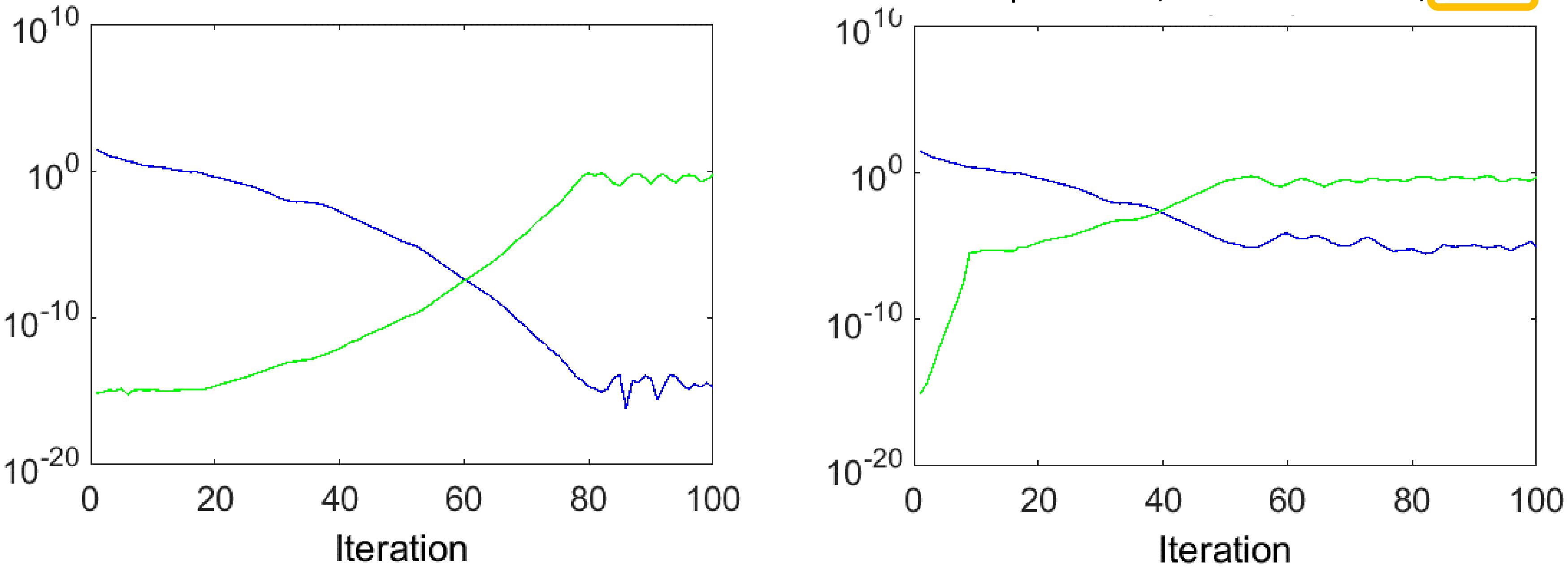
Measure of Ritz value convergence

Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector

$\bar{\Gamma}_k \leq 2 \times 10^6$

classical Lanczos

s-step Lanczos, monomial basis, $\boxed{s = 8}$

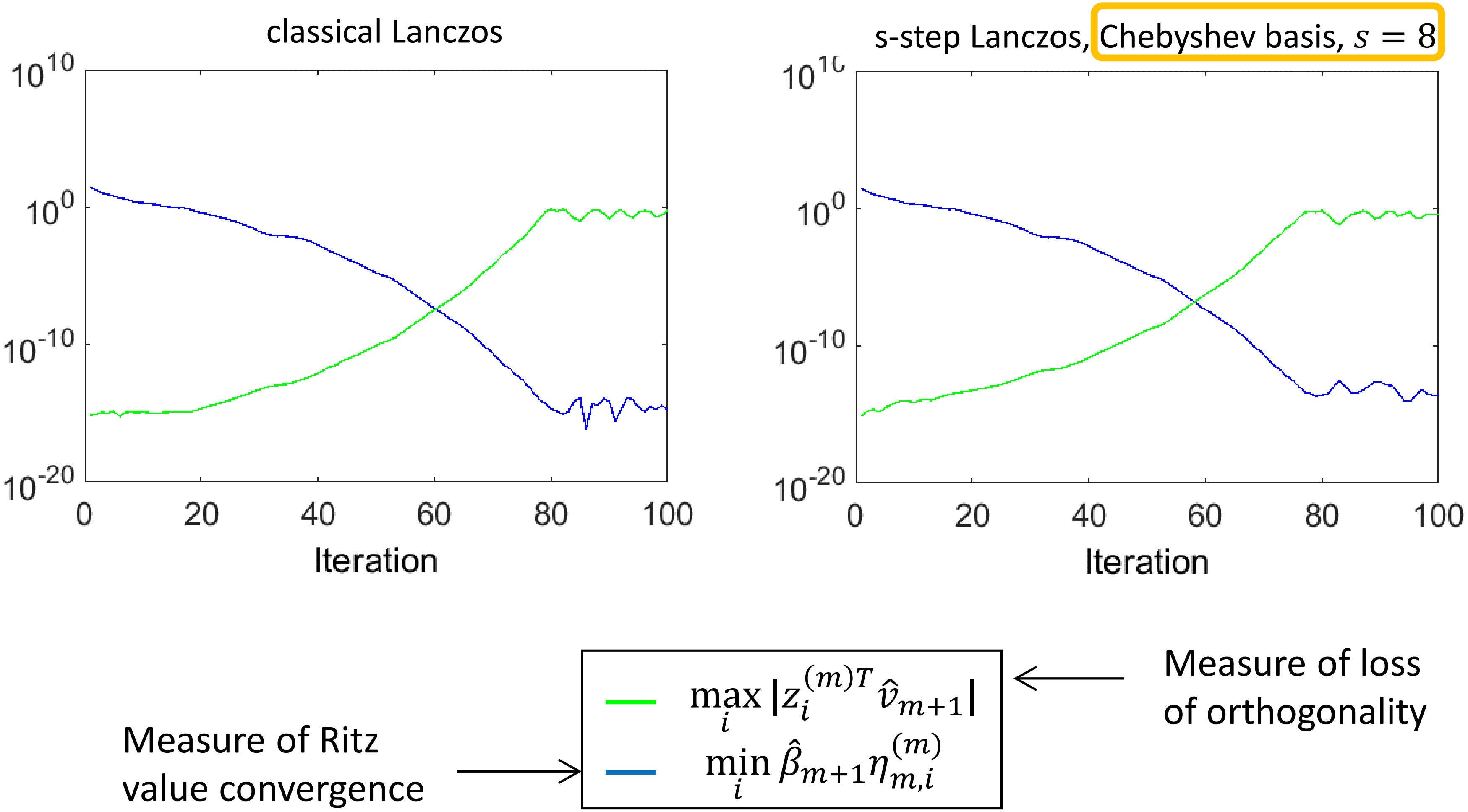


🟩 $\max_i |z_i^{(m)T} \hat{v}_{m+1}|$ ← Measure of loss of orthogonality

Measure of Ritz value convergence → 🟦 $\min_i \hat{\beta}_{m+1} \eta_{m,i}^{(m)}$

Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector

$\bar{\Gamma}_k \leq 2 \times 10^3$



classical Lanczos

s-step Lanczos, Chebyshev basis, $s = 8$

Measure of loss of orthogonality

Measure of Ritz value convergence

Preconditioning for s-step variants

- Preconditioners improve spectrum of system to improve convergence rate
  - E.g., instead of $Ax = b$, solve $M^{-1}Ax = M^{-1}b$, where $M^{-1} \approx A^{-1}$
  - Essential in practice

- In s-step variants, general preconditioning is a challenge
  - Except for very simple cases, ability to exploit temporal locality across iterations is diminished by preconditioning
  - If possible to avoid communication at all, usually necessitates significant modifications to the algorithm

- Tradeoff: speed up convergence, but increase time per iteration due to communication!
  - For each specific app, must evaluate tradeoff between **preconditioner quality** and **sparsity** of the system

Recent efforts in s-step preconditioners

- Much recent/ongoing work in developing communication-avoiding preconditioned methods
- Many approaches shown to be compatible
    - **Diagonal**
    - **Sparse Approx. Inverse (SAI)** – same sparsity as A; recent work for CA-BICGSTAB by Mehri (2014)
    - **Polynomial preconditioning** (Saad, 1985)
    - **HSS preconditioning** (Hoemmen, 2010); for banded matrices (Knight, C., Demmel, 2014) - same general technique for any system that can be written as sparse + low-rank
    - **CA-ILU(0), CA-ILU(k)** – Moufawad, Grigori (2013), Cayrols, Grigori (2015)
    - **Deflation** for CA-CG (C., Knight, Demmel, 2014), based on Deflated CG of (Saad et al., 2000); for CA-GMRES (Yamazaki et al., 2014)
    - **Domain decomposition** – avoid introducing additional communication by "underlapping" subdomains (Yamazaki, Rajamanickam, Boman, Hoemmen, Heroux, Tomov, 2014)

Summary

- New communication-avoiding approaches to algorithm design are necessary
  - But modifications may affect numerical properties
- s-step Krylov subspace methods can asymptotically reduce communication cost; potential applications in many scientific domains
  - But complicated tradeoffs depending on matrix structure, numerical properties, and machine parameters
- Solving exascale-level problems efficiently will require a *holistic* approach
  - Best method, best parameters, best preconditioners, etc. all very problem- and machine-dependent
  - Requires a better understanding of how algorithmic changes affect finite precision behavior

Thank you!

contact: erinc@cims.nyu.edu

http://www.cims.nyu.edu/~erinc/