Communication-Avoiding Krylov Subspace Methods in Theory and Practice

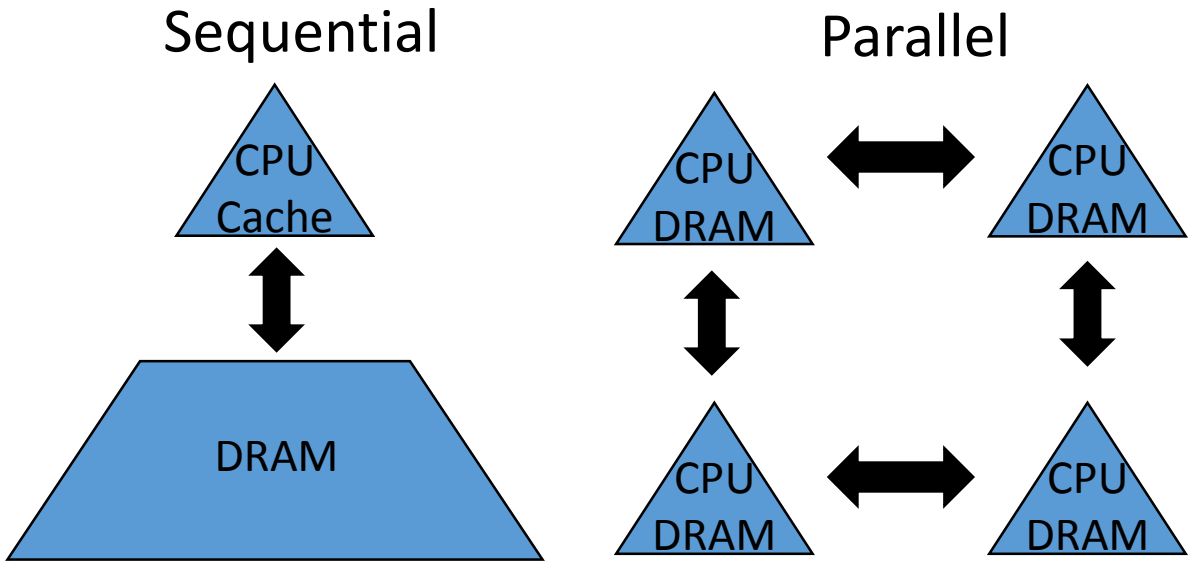
Erin Carson, NYU

DMML Workshop

October 23, 2015

Why Avoid "Communication"?

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



- On today's computers, communication is expensive, computation is cheap, in terms of both time and energy!

Future Exascale Systems

	Petascale Systems (2009)	Predicted Exascale Systems*	Factor Improvement
System Peak	$2 \cdot 10^{15}$ flops	10^{18} flops	~1000
Node Memory Bandwidth	25 GB/s	0.4-4 TB/s	~10-100
Total Node Interconnect Bandwidth	3.5 GB/s	100-400 GB/s	~100
Memory Latency	100 ns	50 ns	~1
Interconnect Latency	1 μs	0.5 μs	~1

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

Future Exascale Systems

	Petascale Systems (2009)	Predicted Exascale Systems*	Factor Improvement
System Peak	$2 \cdot 10^{15}$ flops	10^{18} flops	~1000
Node Memory Bandwidth	25 GB/s	0.4-4 TB/s	~10-100
Total Node Interconnect Bandwidth	3.5 GB/s	100-400 GB/s	~100
Memory Latency	100 ns	50 ns	~1
Interconnect Latency	1 μs	0.5 μs	~1

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

- Gaps between communication/computation cost only growing larger in future systems

Future Exascale Systems

	Petascale Systems (2009)	Predicted Exascale Systems*	Factor Improvement
System Peak	$2 \cdot 10^{15}$ flops	10^{18} flops	~1000
Node Memory Bandwidth	25 GB/s	0.4-4 TB/s	~10-100
Total Node Interconnect Bandwidth	3.5 GB/s	100-400 GB/s	~100
Memory Latency	100 ns	50 ns	~1
Interconnect Latency	1 μs	0.5 μs	~1

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

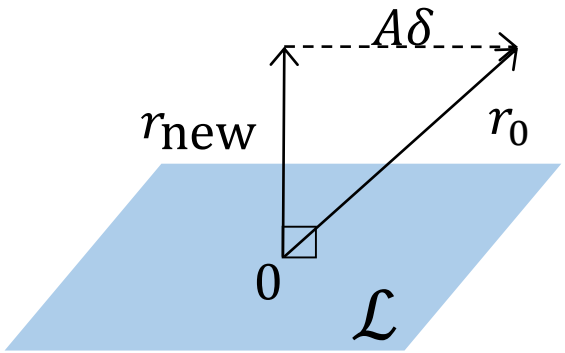
- Gaps between communication/computation cost only growing larger in future systems
- **Avoiding communication will be essential for applications at exascale!**

Krylov Subspace Methods

- General class of iterative solvers: used for linear systems, eigenvalue problems, singular value problems, least squares, etc.
- Examples: Lanczos/Conjugate Gradient (CG), Arnoldi/Generalized Minimum Residual (GMRES), Biconjugate Gradient (BICG), BICGSTAB, GKL, LSQR, etc.
- Projection process onto the expanding **Krylov subspace**

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \dots, A^{m-1} r_0\}$$

- In each iteration,
  - Add a dimension to the Krylov subspace $\mathcal{K}_m$
  - Orthogonalize (with respect to some $\mathcal{L}_m$)



Krylov Solvers: Limited by Communication

In terms of communication:

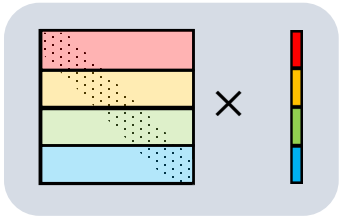
Krylov Solvers: Limited by Communication

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



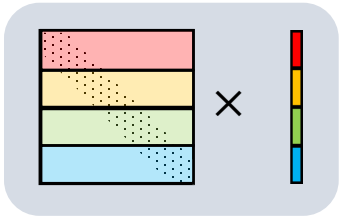
Krylov Solvers: Limited by Communication

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 (All-Reduce)

Sequential: multiple reads/writes to slow memory

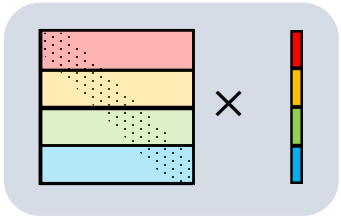
Krylov Solvers: Limited by Communication

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

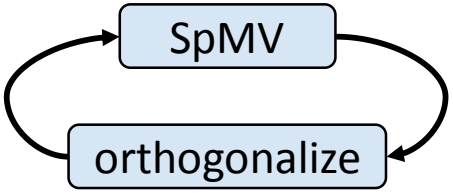


"Orthogonalize (with respect to some $\mathcal{L}_m$)"

$\rightarrow$ Inner products

Parallel: global reduction (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_0 for solving $Ax = b$

Let $p_0 = r_0 = b - Ax_0$

for $m = 0, 1, 2, \ldots,$ until convergence **do**

$$\alpha_m = \frac{r_m^T r_m}{p_m^T A p_m}$$

$$x_{m+1} = x_m + \alpha_m p_m$$

$$r_{m+1} = r_m - \alpha_m A p_m$$

$$\beta_{m+1} = \frac{r_{m+1}^T r_{m+1}}{r_m^T r_m}$$

$$p_{m+1} = r_{m+1} + \beta_{m+1} p_m$$

end for

Example: Classical Conjugate Gradient (CG)

Given: initial approximation x_0 for solving $Ax = b$

Let $p_0 = r_0 = b - Ax_0$

for $m = 0, 1, 2, \ldots,$ until convergence **do**

$$\alpha_m = \frac{r_m^T r_m}{p_m^T A p_m}$$

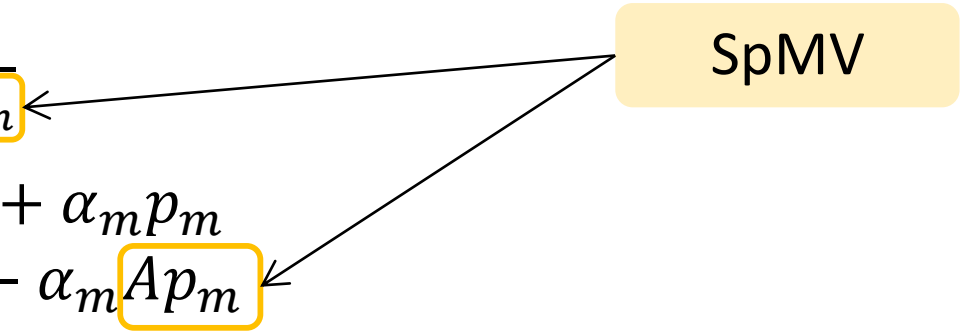
$$x_{m+1} = x_m + \alpha_m p_m$$

$$r_{m+1} = r_m - \alpha_m A p_m$$

$$\beta_{m+1} = \frac{r_{m+1}^T r_{m+1}}{r_m^T r_m}$$

$$p_{m+1} = r_{m+1} + \beta_{m+1} p_m$$

end for



Example: Classical Conjugate Gradient (CG)

Given: initial approximation x_0 for solving $Ax = b$

Let $p_0 = r_0 = b - Ax_0$

for $m = 0, 1, 2, \ldots,$ until convergence **do**

$$\alpha_m = \frac{r_m^T r_m}{p_m^T A p_m}$$

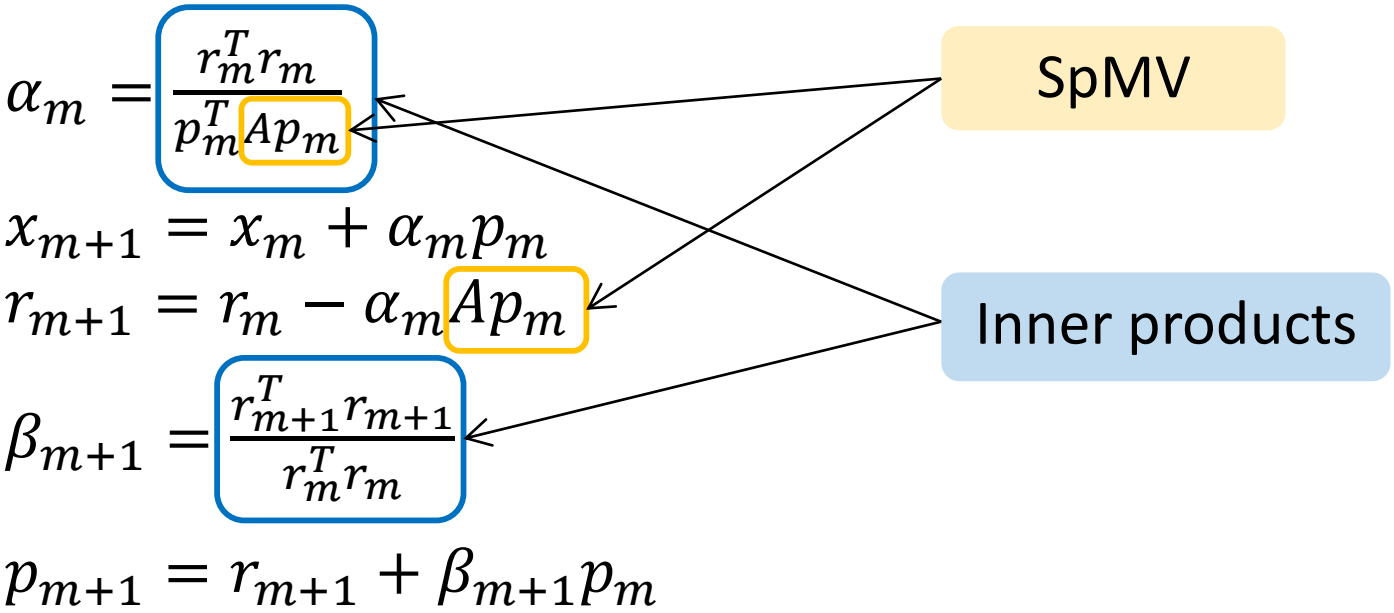
$$x_{m+1} = x_m + \alpha_m p_m$$

$$r_{m+1} = r_m - \alpha_m A p_m$$

$$\beta_{m+1} = \frac{r_{m+1}^T r_{m+1}}{r_m^T r_m}$$

$$p_{m+1} = r_{m+1} + \beta_{m+1} p_m$$

end for



Communication-Avoiding KSMs

- Idea: Compute blocks of s iterations at once
  - Communicate every s iterations instead of every iteration
  - **Reduces communication cost by $O(s)$!**
    - (latency in parallel, latency and bandwidth in sequential)

Communication-Avoiding KSMs

- Idea: Compute blocks of s iterations at once
  - Communicate every s iterations instead of every iteration
  - **Reduces communication cost by $O(s)$!**
    - (latency in parallel, latency and bandwidth in sequential)

- An idea rediscovered many times…
- First related work: s-dimensional steepest descent - Khabaza (‘63), Forsythe (‘68), Marchuk and Kuznecov (‘68):
- Flurry of work on s-step Krylov methods in ‘80s/early ‘90s: see, e.g., Van Rosendale, 1983; Chronopoulos and Gear, 1989
  - Goals: increasing parallelism, avoiding I/O, increasing “convergence rate”

Communication-Avoiding KSMs

- Idea: Compute blocks of s iterations at once
  - Communicate every s iterations instead of every iteration
  - **Reduces communication cost by $O(s)$!**
    - (latency in parallel, latency and bandwidth in sequential)
- An idea rediscovered many times…
- First related work: s-dimensional steepest descent - Khabaza ('63), Forsythe ('68), Marchuk and Kuznecov ('68):
- Flurry of work on s-step Krylov methods in '80s/early '90s: see, e.g., Van Rosendale, 1983; Chronopoulos and Gear, 1989
  - Goals: increasing parallelism, avoiding I/O, increasing "convergence rate"
- Resurgence of interest in recent years due to growing problem sizes; growing relative cost of communication

Communication-Avoiding KSMs: CA-CG

- Main idea: Unroll iteration loop by a factor of s; split iteration loop into an outer loop and an inner loop
- Key observation: starting at some iteration m,

$$x_{m+j} - x_m,\ r_{m+j},\ p_{m+j} \in \mathcal{K}_{s+1}(A, p_m) + \mathcal{K}_s(A, r_m) \quad \text{for} \quad j \in \{0, \ldots, s\}$$

Communication-Avoiding KSMs: CA-CG

- Main idea: Unroll iteration loop by a factor of s; split iteration loop into an outer loop and an inner loop
- Key observation: starting at some iteration m,

$$x_{m+j} - x_m,\ r_{m+j},\ p_{m+j} \in \mathcal{K}_{s+1}(A, p_m) + \mathcal{K}_s(A, r_m) \quad \text{for} \quad j \in \{0, \ldots, s\}$$

Outer loop k: Communication step

Expand solution space s dimensions at once

- Compute "basis matrix" Y_k with columns spanning

$$\mathcal{K}_{s+1}(A, p_m) + \mathcal{K}_s(A, r_m)$$

- Requires **reading A/communicating vectors only once**
  - Using "matrix powers kernel"

Orthogonalize all at once

- Compute/store block of inner products between basis vectors in Gram matrix:

$$G_k = Y_k^T Y_k$$

- Communication cost of **one global reduction**

Communication-Avoiding KSMs: CA-CG

Inner loop: Computation steps, no communication!

Perform s iterations of updates

- Using Y_k and G_k, this requires **no communication!**
- Represent n-vectors by their $O(s)$ coordinates in Y_k:

$$x_{sk+j} - x_{sk} = Y_k x'_j, \quad r_{sk+j} = Y_k r'_j, \quad p_{sk+j} = Y_k p'_j$$

Communication-Avoiding KSMs: CA-CG

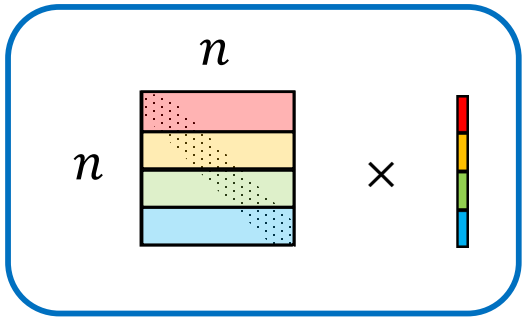
Inner loop: Computation steps, no communication!

Perform s iterations of updates

- Using Y_k and G_k, this requires **no communication!**
- Represent n-vectors by their $O(s)$ coordinates in Y_k:

$$x_{sk+j} - x_{sk} = Y_k x'_j, \quad r_{sk+j} = Y_k r'_j, \quad p_{sk+j} = Y_k p'_j$$

Ap_{sk+j}



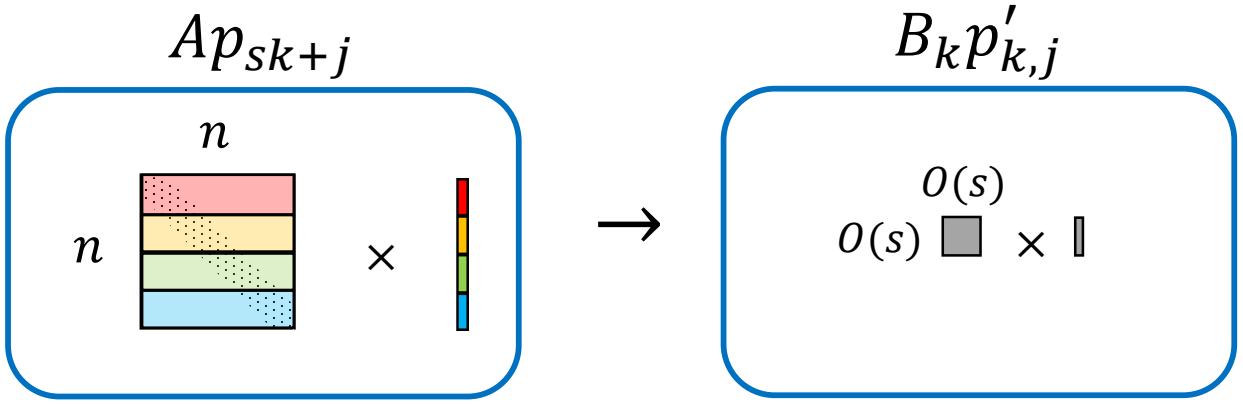
Communication-Avoiding KSMs: CA-CG

Inner loop: Computation steps, no communication!

Perform s iterations of updates

- Using Y_k and G_k, this requires **no communication!**
- Represent n-vectors by their $O(s)$ coordinates in Y_k:

$$x_{sk+j} - x_{sk} = Y_k x'_j, \quad r_{sk+j} = Y_k r'_j, \quad p_{sk+j} = Y_k p'_j$$



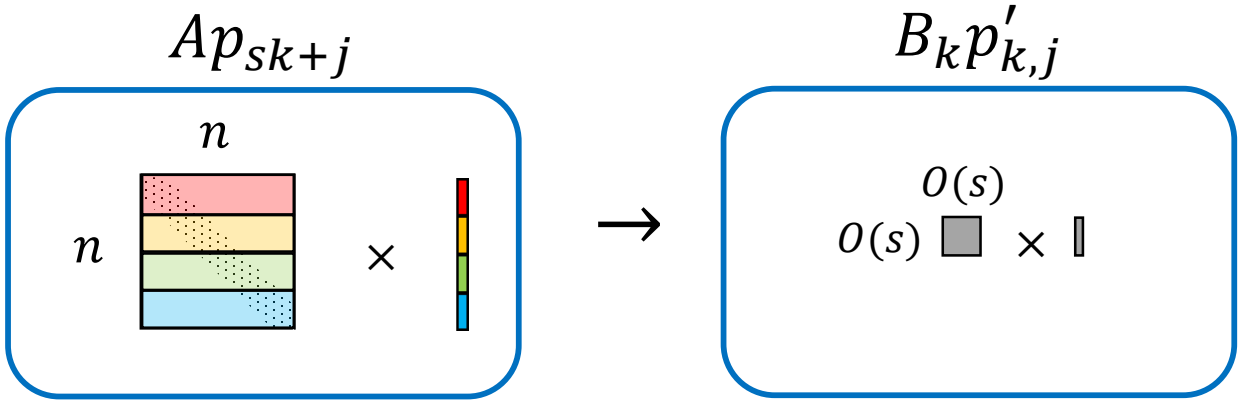
Communication-Avoiding KSMs: CA-CG

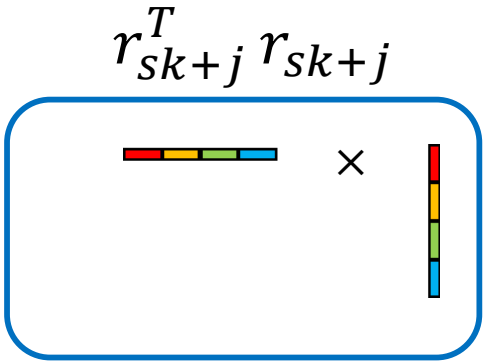
Inner loop: Computation steps, no communication!

Perform s iterations of updates

- Using Y_k and G_k, this requires **no communication!**
- Represent n-vectors by their $O(s)$ coordinates in Y_k:

$$x_{sk+j} - x_{sk} = Y_k x'_j, \quad r_{sk+j} = Y_k r'_j, \quad p_{sk+j} = Y_k p'_j$$





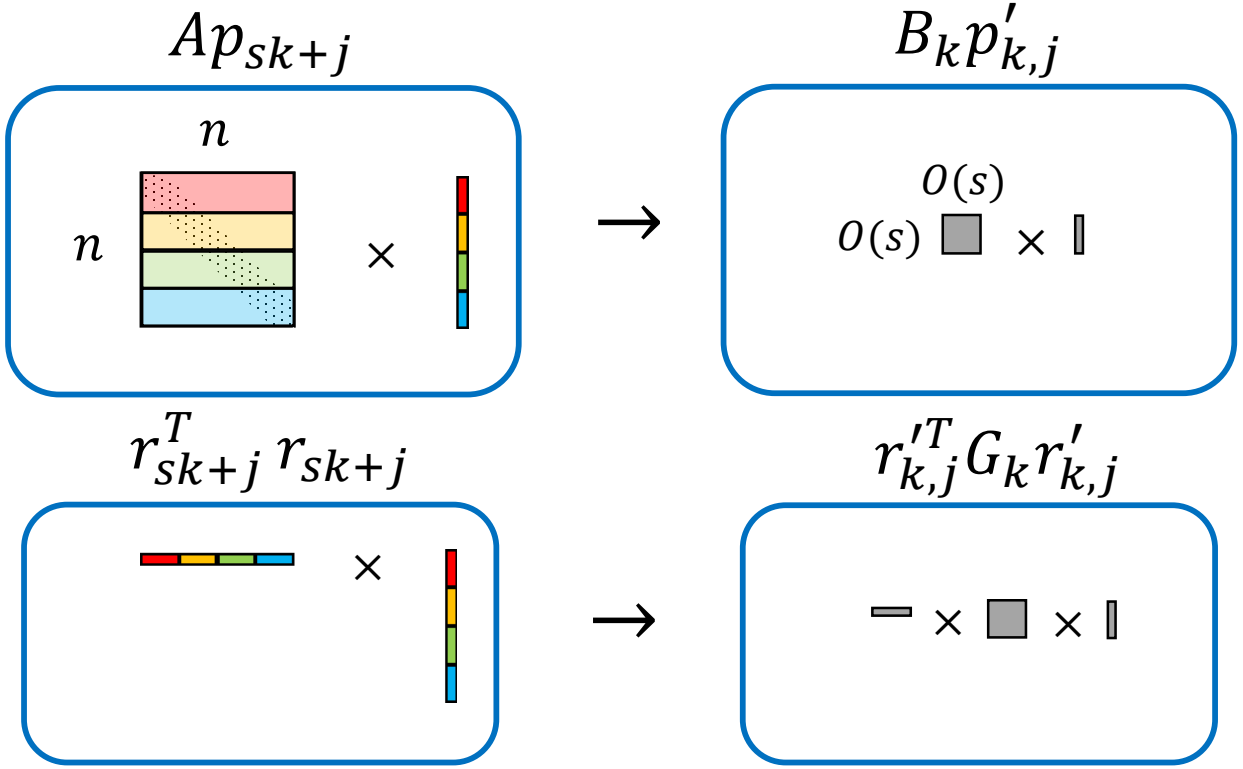
Communication-Avoiding KSMs: CA-CG

Inner loop: Computation steps, no communication!

Perform s iterations of updates

- Using Y_k and G_k, this requires **no communication!**
- Represent n-vectors by their $O(s)$ coordinates in Y_k:

$$x_{sk+j} - x_{sk} = Y_k x'_j, \quad r_{sk+j} = Y_k r'_j, \quad p_{sk+j} = Y_k p'_j$$



Example: CA-Conjugate Gradient

Given: initial approximation x_0 for solving $Ax = b$

Let $p_0 = r_0 = b - Ax_0$

for k $= 0, 1, \ldots,$ until convergence **do**

Compute Y_k, compute $G_k = Y_k^T Y_k$

Let $x'_0 = 0_{2s+1},\ r'_0 = e_{s+2},\ p'_0 = e_1$

for $j = 0, \ldots, s-1$ **do**

$$\alpha_{sk+j} = \frac{\left(r'_j\right)^T G_k r'_j}{\left(p'_j\right)^T G_k B_k p'_j}$$

$$x'_{j+1} = x'_j + \alpha_{sk+j} p'_j$$

$$r'_{j+1} = r'_j - \alpha_{sk+j} B_k p'_j$$

$$\beta_{sk+j+1} = \frac{\left(r'_{j+1}\right)^T G_k r'_{j+1}}{\left(r'_j\right)^T G_k r'_j}$$

$$p'_{j+1} = r'_{j+1} + \beta_{sk+j+1} p'_j$$

end for

Compute $x_{sk+s} = Y_k x'_s + x_{sk},\ r_{sk+s} = Y_k r'_s,\ p_{sk+s} = Y_k p'_s$

end for

Example: CA-Conjugate Gradient

Given: initial approximation x_0 for solving $Ax = b$
Let $p_0 = r_0 = b - Ax_0$
for k $= 0, 1, \ldots,$ until convergence **do**
  Compute Y_k, compute $G_k = Y_k^T Y_k$
  Let $x'_0 = 0_{2s+1},\ r'_0 = e_{s+2},\ p'_0 = e_1$
  for $j = 0, \ldots, s-1$ **do**

$$\alpha_{sk+j} = \frac{\left(r'_j\right)^T G_k r'_j}{\left(p'_j\right)^T G_k B_k p'_j}$$

$$x'_{j+1} = x'_j + \alpha_{sk+j} p'_j$$

$$r'_{j+1} = r'_j - \alpha_{sk+j} B_k p'_j$$

$$\beta_{sk+j+1} = \frac{\left(r'_{j+1}\right)^T G_k r'_{j+1}}{\left(r'_j\right)^T G_k r'_j}$$

$$p'_{j+1} = r'_{j+1} + \beta_{sk+j+1} p'_j$$

  end for
  Compute $x_{sk+s} = Y_k x'_s + x_{sk},\ r_{sk+s} = Y_k r'_s,\ p_{sk+s} = Y_k p'_s$
end for

via CA Matrix Powers Kernel

Global reduction to compute G_k

Example: CA-Conjugate Gradient

Given: initial approximation x_0 for solving $Ax = b$
Let $p_0 = r_0 = b - Ax_0$
for k $= 0, 1, \ldots,$ until convergence **do**
  Compute Y_k, compute $G_k = Y_k^T Y_k$
  Let $x'_0 = 0_{2s+1},\ r'_0 = e_{s+2},\ p'_0 = e_1$
  for $j = 0, \ldots, s-1$ **do**

$$\alpha_{sk+j} = \frac{\left(r'_j\right)^T G_k r'_j}{\left(p'_j\right)^T G_k B_k p'_j}$$
$$x'_{j+1} = x'_j + \alpha_{sk+j} p'_j$$
$$r'_{j+1} = r'_j - \alpha_{sk+j} B_k p'_j$$
$$\beta_{sk+j+1} = \frac{\left(r'_{j+1}\right)^T G_k r'_{j+1}}{\left(r'_j\right)^T G_k r'_j}$$
$$p'_{j+1} = r'_{j+1} + \beta_{sk+j+1} p'_j$$

  end for
  Compute $x_{sk+s} = Y_k x'_s + x_{sk},\ r_{sk+s} = Y_k r'_s,\ p_{sk+s} = Y_k p'_s$
end for

via CA Matrix Powers Kernel

Global reduction to compute G_k

Local computations within inner loop require no communication!

Complexity Comparison

Example of parallel (per processor) complexity for s iterations of CG vs. CA-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	$s \log_2 p$
CA-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)

Complexity Comparison

Example of parallel (per processor) complexity for s iterations of CG vs. CA-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	$s \log_2 p$
CA-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)

Complexity Comparison

Example of parallel (per processor) complexity for s iterations of CG vs. CA-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	$s \log_2 p$
CA-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)

From Theory to Practice

- 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 fastest **speed per iteration**

From Theory to Practice

- 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 fastest **speed per iteration**

- In practice, we don't just care about speed per iteration, but also the number of iterations

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

From Theory to Practice

- 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 fastest **speed per iteration**

- In practice, we don't just care about speed per iteration, but also the number of iterations

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

- We also need to consider how convergence rate and accuracy are affected by choice of s!

From Theory to Practice

- CA-KSMs are mathematically equivalent to classical KSMs

From Theory to Practice

- CA-KSMs are mathematically equivalent to classical KSMs
- But can behave much differently in finite precision!

From Theory to Practice

- CA-KSMs are mathematically equivalent to classical KSMs
- But can behave much differently in finite precision!
- Roundoff error bounds generally grow with increasing s

From Theory to Practice

- CA-KSMs are mathematically equivalent to classical KSMs
- But can behave much differently in finite precision!
- Roundoff error bounds generally grow with increasing s
- Two effects of roundoff error:

From Theory to Practice

- CA-KSMs are mathematically equivalent to classical KSMs
- But can behave much differently in finite precision!
- Roundoff error bounds generally grow with increasing s
- Two effects of roundoff error:
  1. **Decrease in accuracy** → Tradeoff: increasing blocking factor s past a certain point: **true residual $\boldsymbol{b - Ax}$** stagnates

From Theory to Practice

- CA-KSMs are mathematically equivalent to classical KSMs
- But can behave much differently in finite precision!
- Roundoff error bounds generally grow with increasing s
- Two effects of roundoff error:
  1. **Decrease in accuracy** → Tradeoff: increasing blocking factor s past a certain point: **true residual $\boldsymbol{b - Ax}$** stagnates
  2. **Delay of convergence** → Tradeoff: increasing blocking factor s past a certain point: no speedup expected

From Theory to Practice

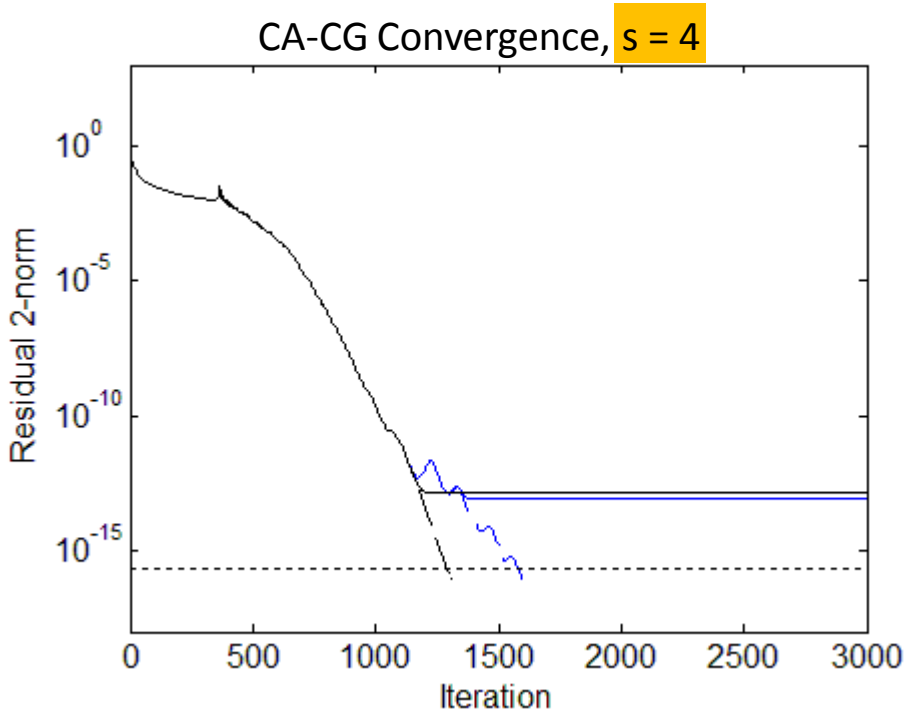
- CA-KSMs are mathematically equivalent to classical KSMs
- But can behave much differently in finite precision!
- Roundoff error bounds generally grow with increasing s
- Two effects of roundoff error:
  1. **Decrease in accuracy** → Tradeoff: increasing blocking factor s past a certain point: **true residual $\boldsymbol{b - Ax}$** stagnates
  2. **Delay of convergence** → Tradeoff: increasing blocking factor s past a certain point: no speedup expected

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

From Theory to Practice

- CA-KSMs are mathematically equivalent to classical KSMs
- But can behave much differently in finite precision!
- Roundoff error bounds generally grow with increasing s
- Two effects of roundoff error:
  1. **Decrease in accuracy** → Tradeoff: increasing blocking factor s past a certain point: **true residual $\boldsymbol{b - Ax}$** stagnates
  2. **Delay of convergence** → Tradeoff: increasing blocking factor s past a certain point: no speedup expected

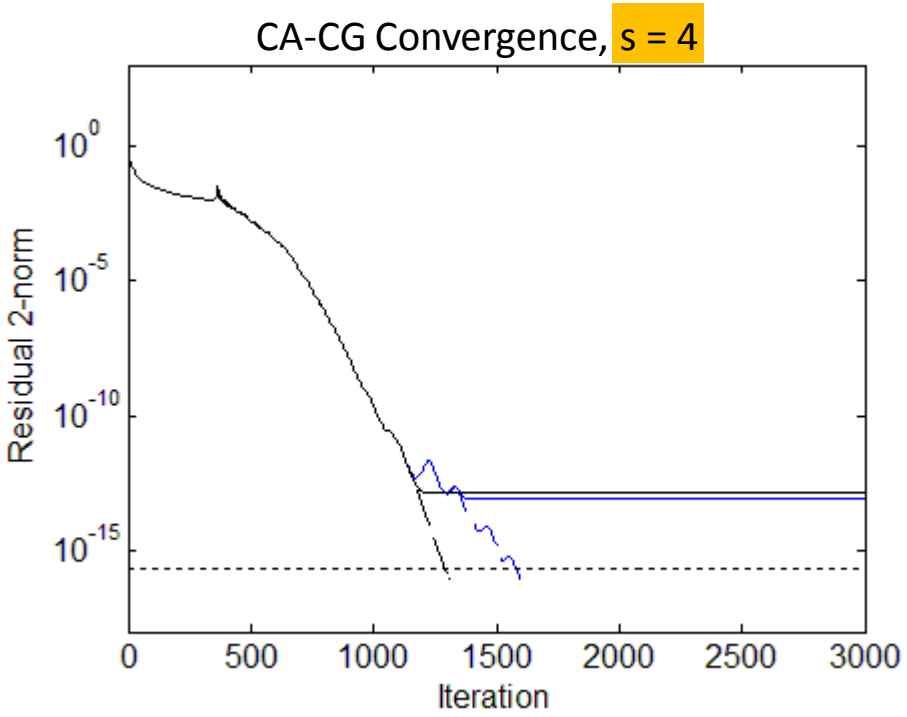
Runtime = (time/iteration) x (# iterations)

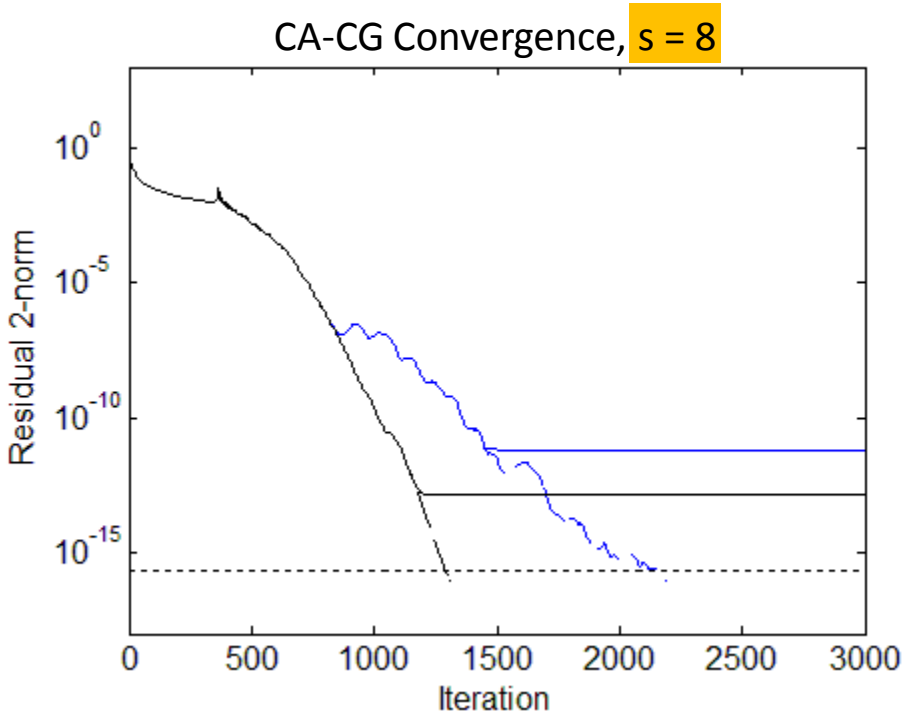


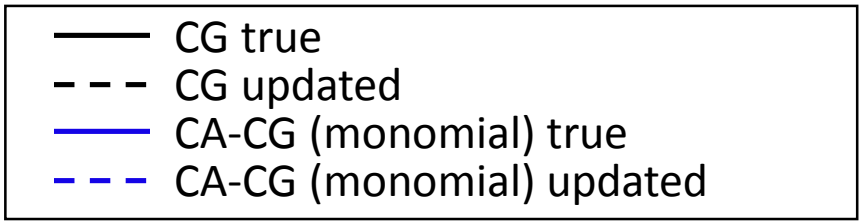
Model Problem: 2D Poisson (5-pt stencil),

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

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



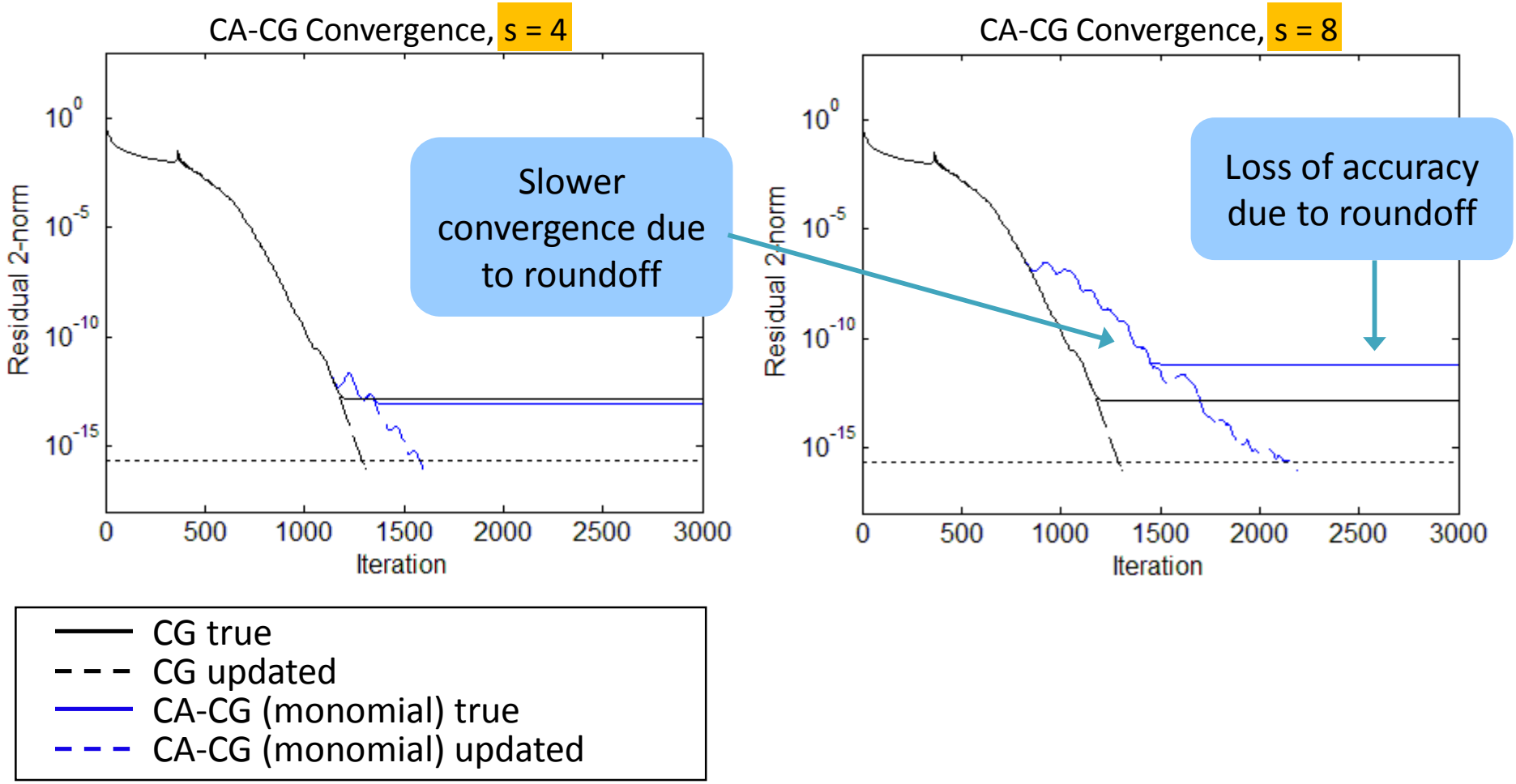




Model Problem: 2D Poisson (5-pt stencil),

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

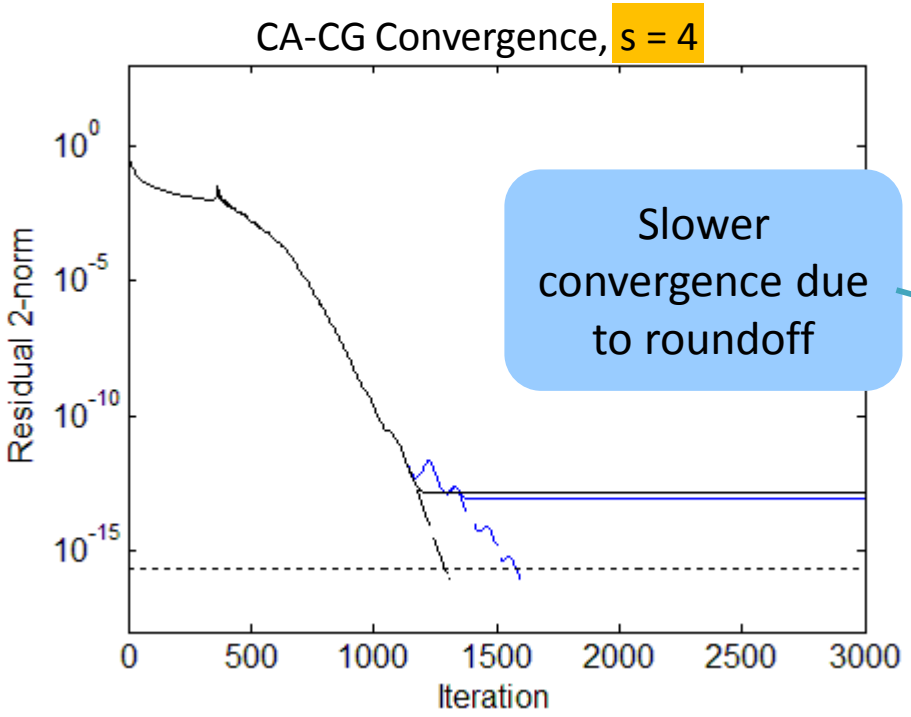
$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$

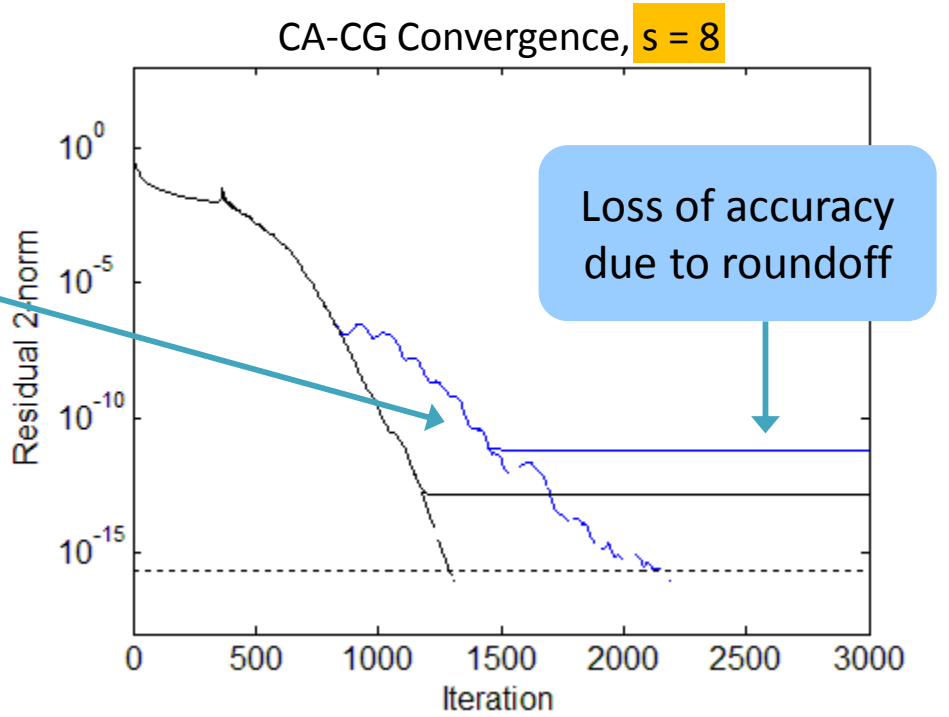


Model Problem: 2D Poisson (5-pt stencil),

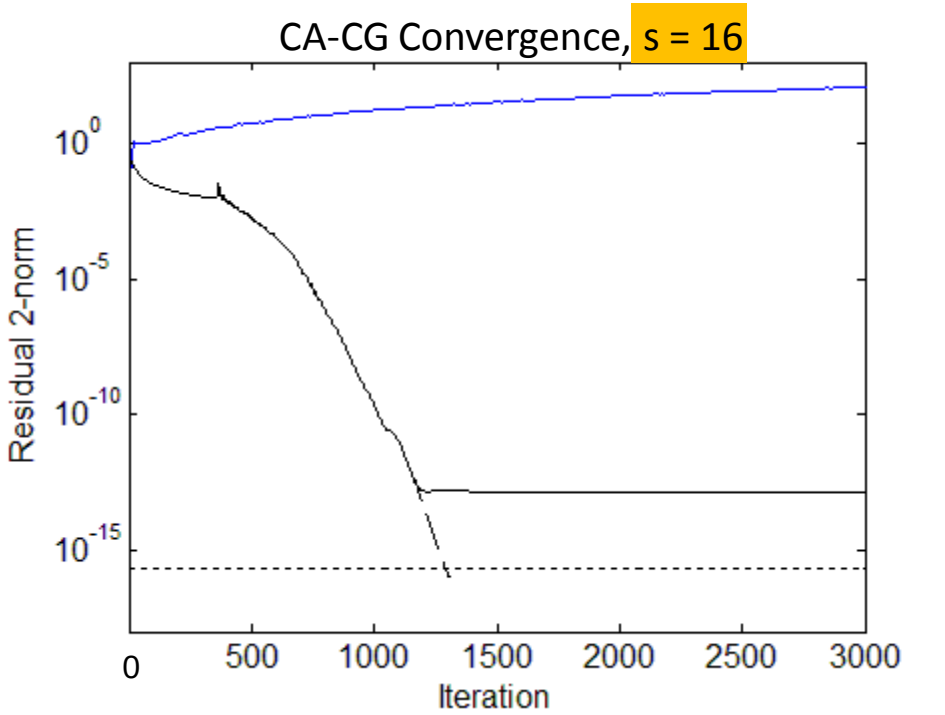
$n = 512^2$, nnz $\approx 10^6$, $\kappa(A) \approx 10^4$

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

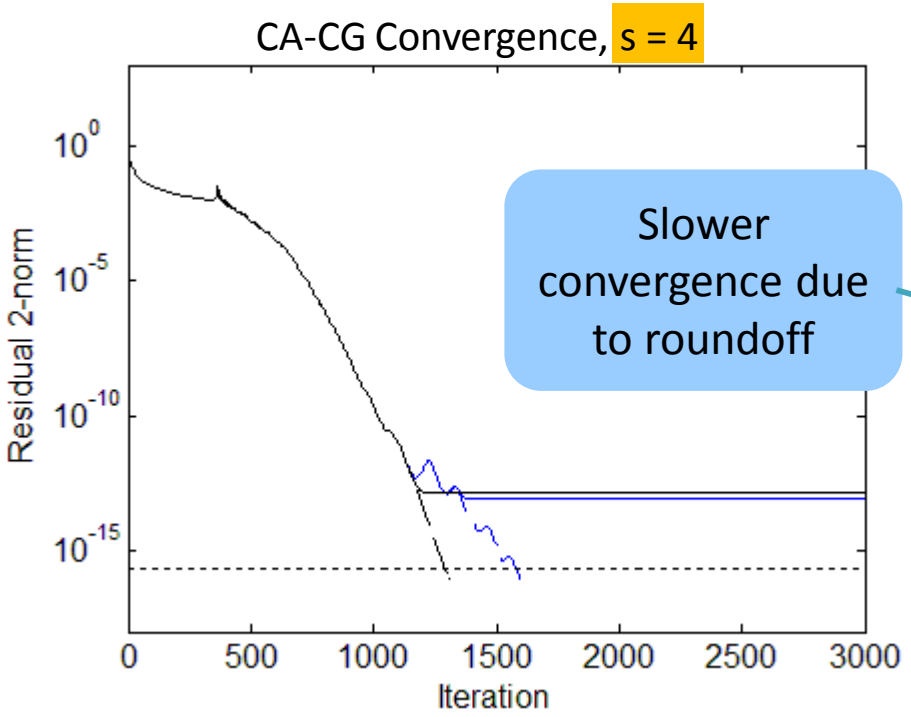


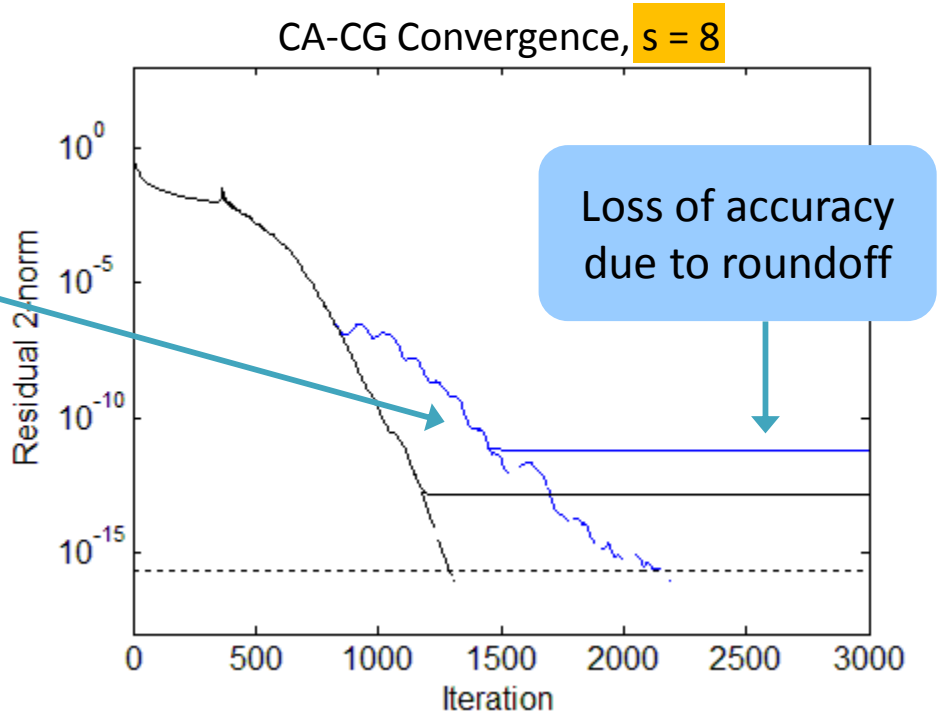


CG true
CG updated
CA-CG (monomial) true
CA-CG (monomial) updated

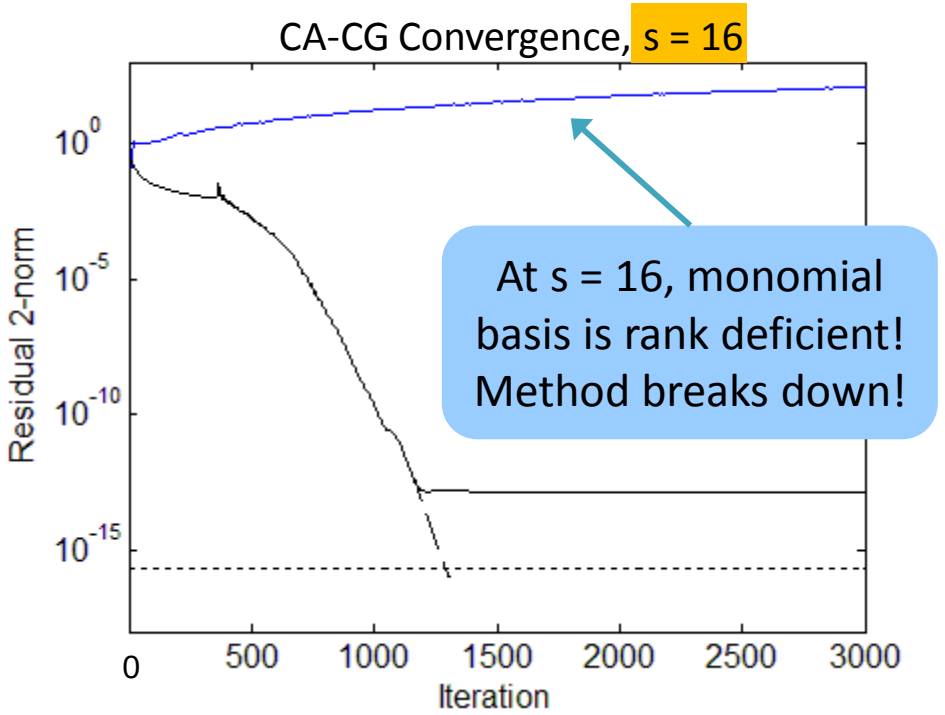


Model Problem: 2D Poisson (5-pt stencil),
$n = 512^2$, nnz $\approx 10^6$, $\kappa(A) \approx 10^4$
$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$



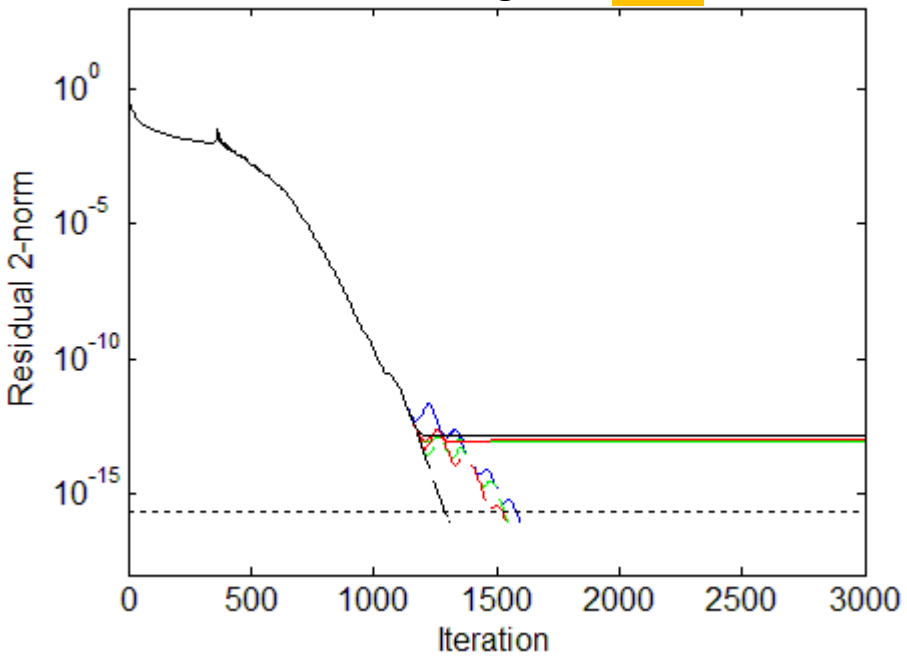


CG true
CG updated
CA-CG (monomial) true
CA-CG (monomial) updated

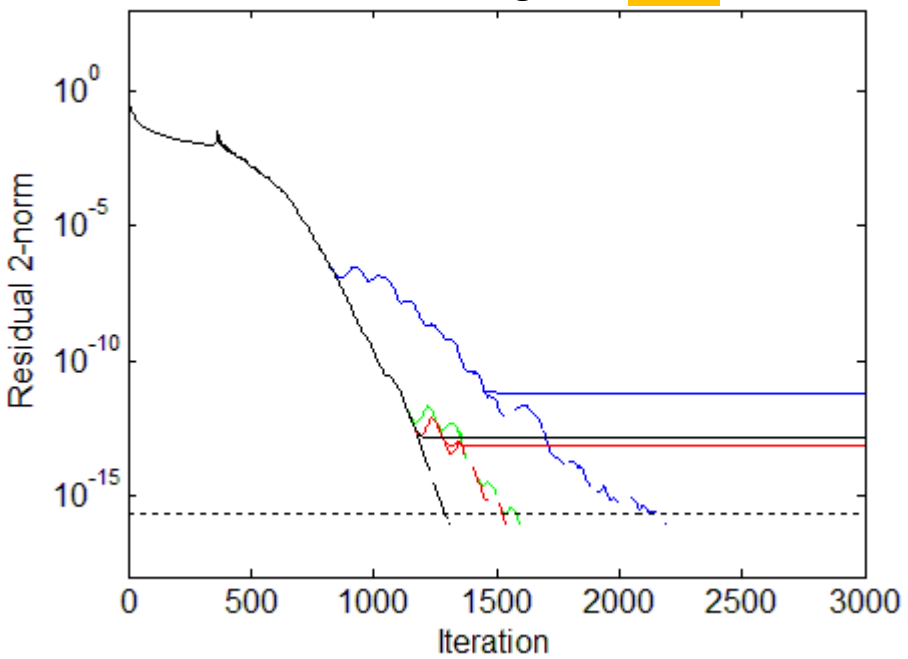


Model Problem: 2D Poisson (5-pt stencil),
$n = 512^2$, nnz $\approx 10^6$, $\kappa(A) \approx 10^4$
$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$

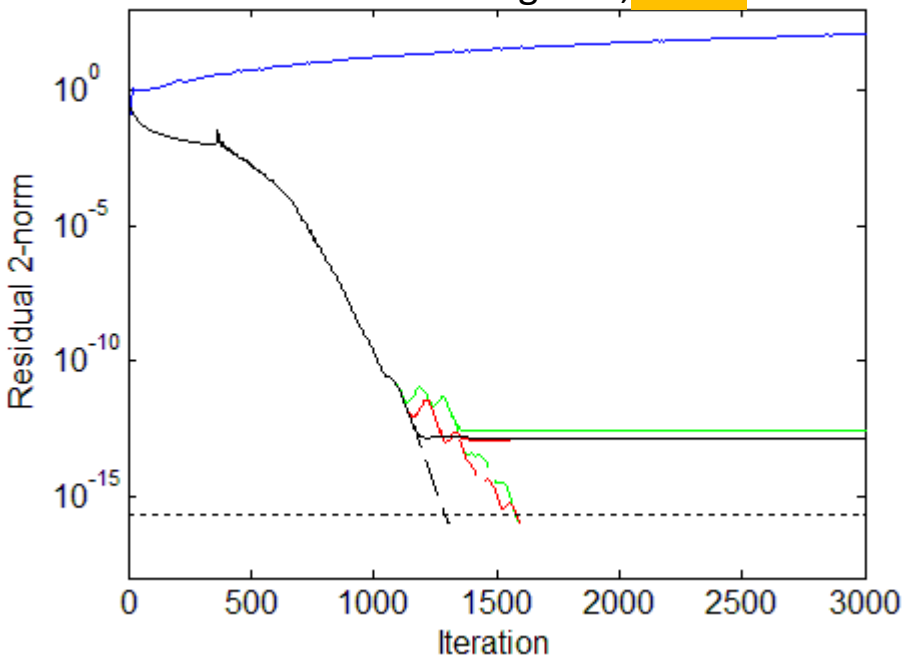
CA-CG Convergence, s = 4



CA-CG Convergence, s = 8



CA-CG Convergence, s = 16

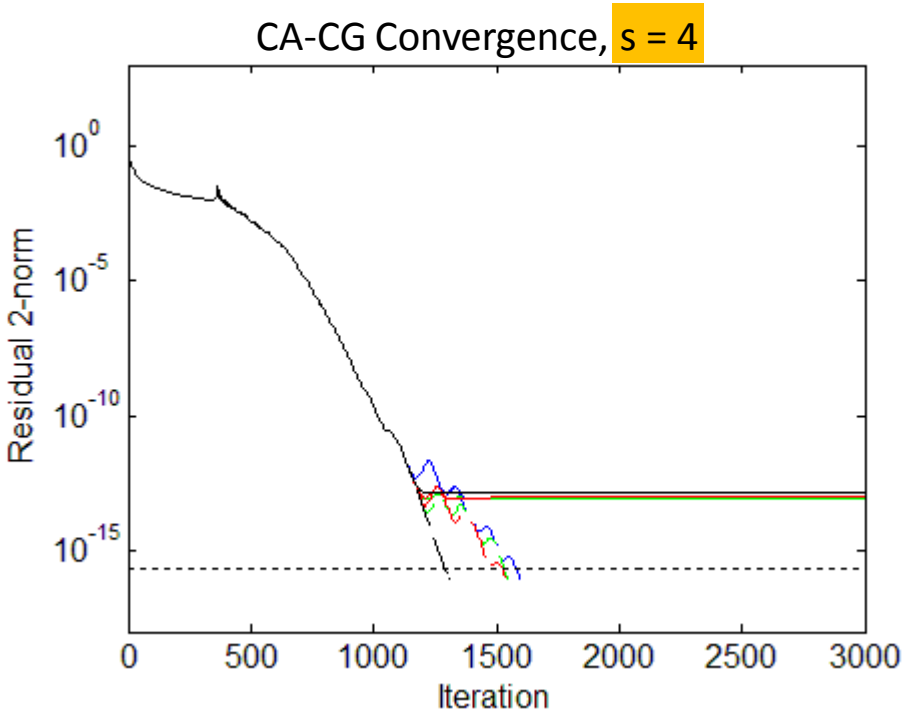


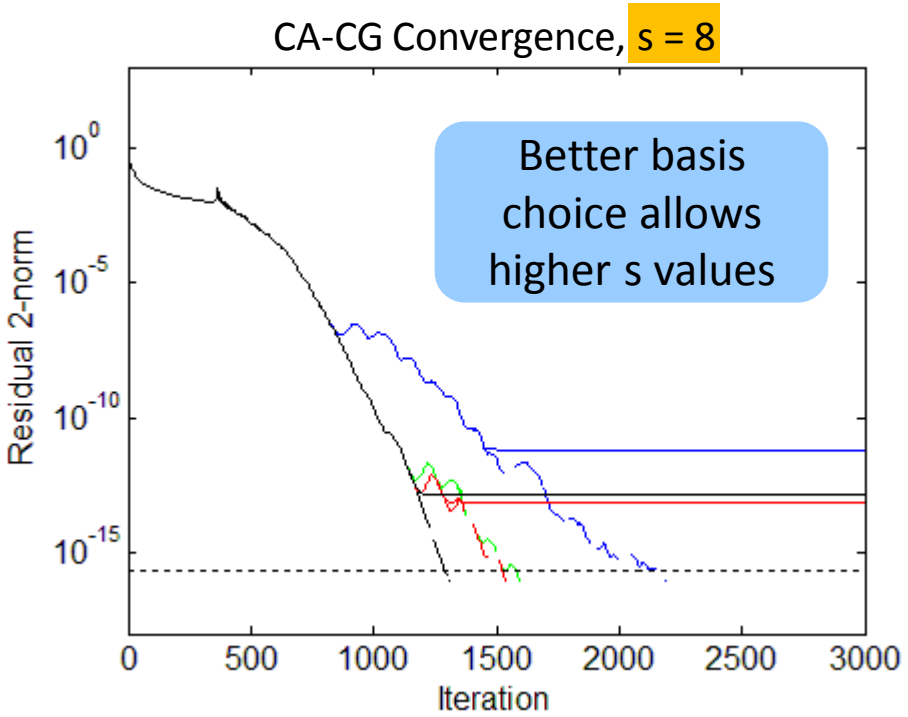
- CG true
- CG updated
- CA-CG (monomial) true
- CA-CG (monomial) updated
- **CA-CG (Newton) true**
- **CA-CG (Newton) updated**
- **CA-CG (Chebyshev) true**
- **CA-CG (Chebyshev) updated**

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

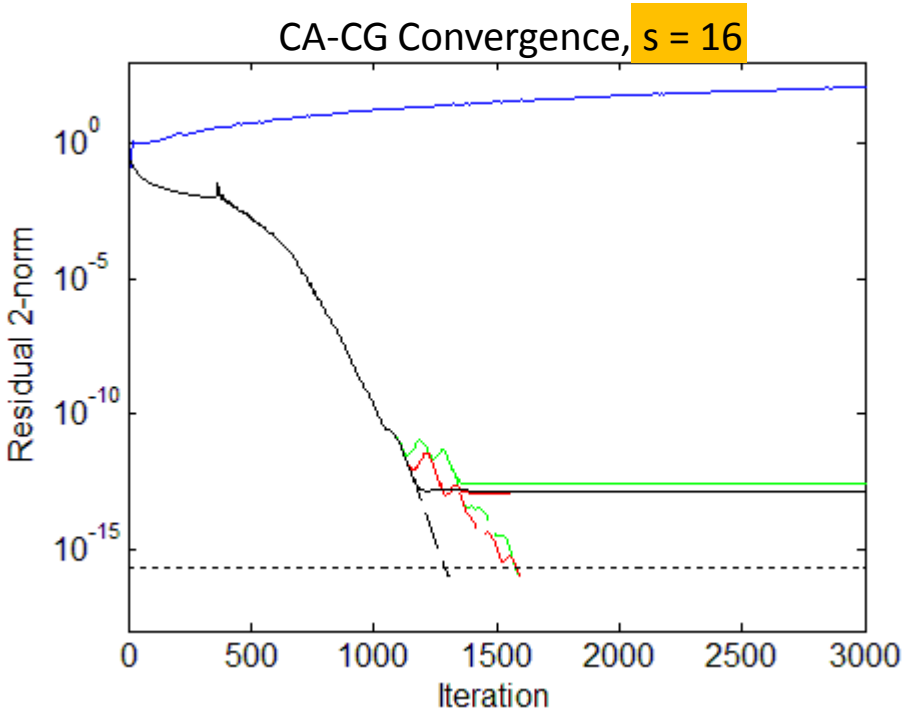
$n = 512^2$, nnz $\approx 10^6$, $\kappa(A) \approx 10^4$

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



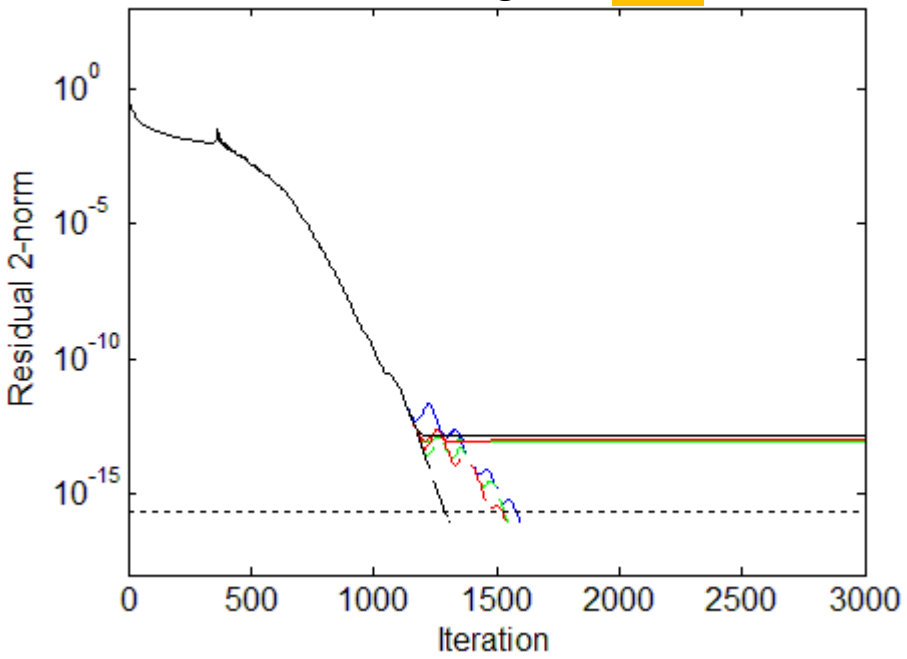


CG true
CG updated
CA-CG (monomial) true
CA-CG (monomial) updated
CA-CG (Newton) true
CA-CG (Newton) updated
CA-CG (Chebyshev) true
CA-CG (Chebyshev) updated

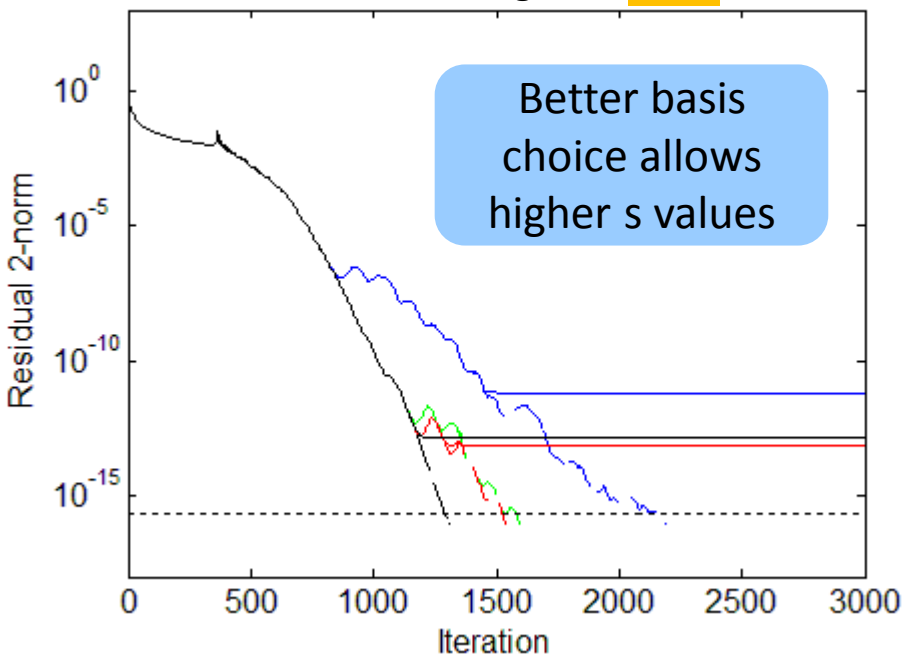


Model Problem: 2D Poisson (5-pt stencil),
$n = 512^2$, nnz $\approx 10^6$, $\kappa(A) \approx 10^4$
$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$

CA-CG Convergence, s = 4

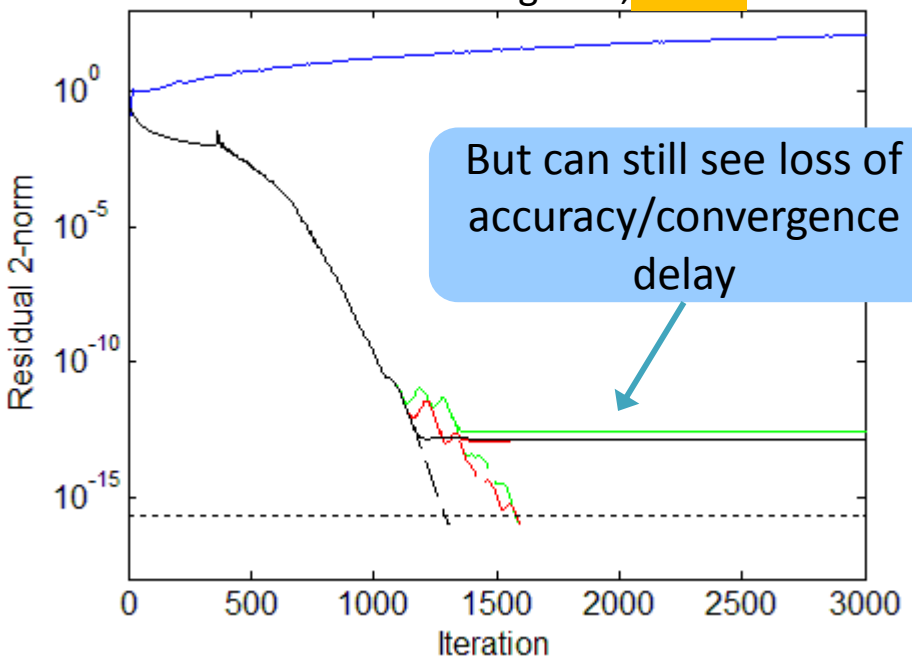


CA-CG Convergence, s = 8



- CG true
- CG updated
- CA-CG (monomial) true
- CA-CG (monomial) updated
- **CA-CG (Newton) true**
- **CA-CG (Newton) updated**
- **CA-CG (Chebyshev) true**
- **CA-CG (Chebyshev) updated**

CA-CG Convergence, s = 16



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

Maximum attainable accuracy of CG

- In classical CG, iterates are updated by

$$x_{m+1} = x_m + \alpha_m p_m \quad \text{and} \quad r_{m+1} = r_m - \alpha_m A p_m$$

- Formulas for x_{m+1} and r_{m+1} do not depend on each other - rounding errors cause the true residual, $b - Ax_{m+1}$, and the updated residual, r_{m+1}, to deviate

Maximum attainable accuracy of CG

- In classical CG, iterates are updated by

$$x_{m+1} = x_m + \alpha_m p_m \quad \text{and} \quad r_{m+1} = r_m - \alpha_m A p_m$$

- Formulas for x_{m+1} and r_{m+1} do not depend on each other - rounding errors cause the true residual, $b - Ax_{m+1}$, and the updated residual, r_{m+1}, to deviate
- The size of the true residual is bounded by

$$\|b - Ax_{m+1}\| \leq \|r_{m+1}\| + \|b - Ax_{m+1} - r_{m+1}\|$$

  - When $\|r_{m+1}\| \gg \|b - Ax_{m+1} - r_{m+1}\|$, $\|r_{m+1}\|$ and $\|b - Ax_{m+1}\|$ have similar magnitude
  - When $\|r_{m+1}\| \to 0$, $\|b - Ax_{m+1}\|$ depends on $\|b - Ax_{m+1} - r_{m+1}\|$

Maximum attainable accuracy of CG

- In classical CG, iterates are updated by

$$x_{m+1} = x_m + \alpha_m p_m \quad \text{and} \quad r_{m+1} = r_m - \alpha_m A p_m$$

- Formulas for x_{m+1} and r_{m+1} do not depend on each other - rounding errors cause the true residual, $b - Ax_{m+1}$, and the updated residual, r_{m+1}, to deviate
- The size of the true residual is bounded by

$$\|b - Ax_{m+1}\| \leq \|r_{m+1}\| + \|b - Ax_{m+1} - r_{m+1}\|$$

  - When $\|r_{m+1}\| \gg \|b - Ax_{m+1} - r_{m+1}\|$, $\|r_{m+1}\|$ and $\|b - Ax_{m+1}\|$ have similar magnitude
  - When $\|r_{m+1}\| \to 0$, $\|b - Ax_{m+1}\|$ depends on $\|b - Ax_{m+1} - r_{m+1}\|$

- 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).
- We have applied a similar analysis to upper bound the maximum attainable accuracy in finite precision CA-KSMs

Residual Replacement Strategy for CG

- van der Vorst and Ye (1999): Improve accuracy by replacing **updated residual** $\boldsymbol{r_{m+1}}$ by the **true residual** $\boldsymbol{b - Ax_{m+1}}$ in certain iterations, combined with group update.

Residual Replacement Strategy for CG

- van der Vorst and Ye (1999): Improve accuracy by replacing **updated residual r_{m+1}** by the **true residual $b - Ax_{m+1}$** in certain iterations, combined with group update.
- Choose when to replace r_{m+1} with $b - Ax_{m+1}$ to meet two constraints:

Residual Replacement Strategy for CG

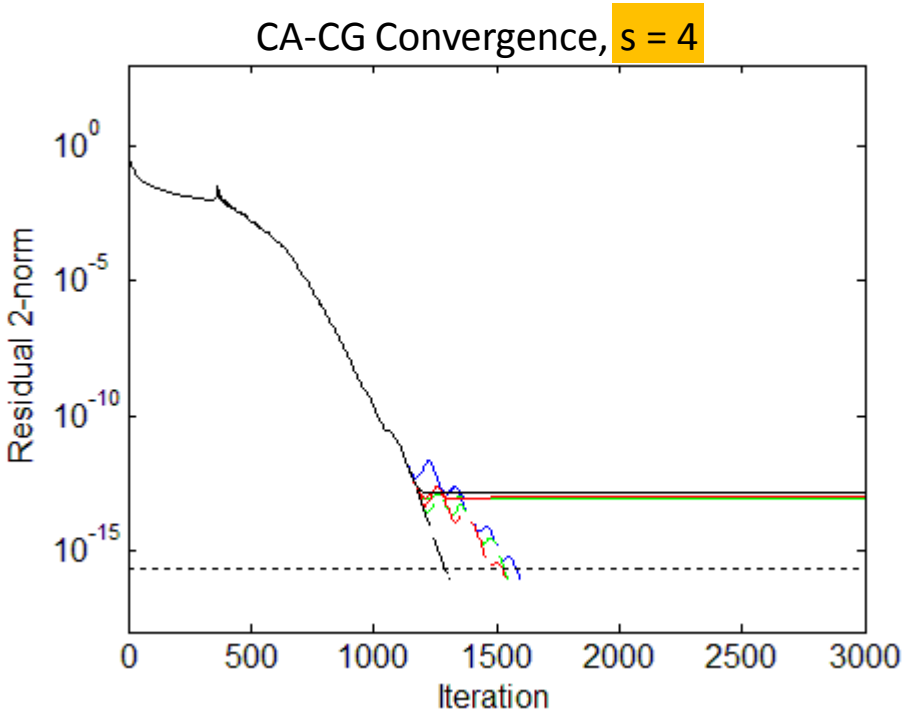
- van der Vorst and Ye (1999): Improve accuracy by replacing **updated residual** $\boldsymbol{r_{m+1}}$ by the **true residual** $\boldsymbol{b - Ax_{m+1}}$ in certain iterations, combined with group update.
- Choose when to replace r_{m+1} with $b - Ax_{m+1}$ to meet two constraints:
  1. **Replace often enough** so that at termination, $\|b - Ax_{m+1} - r_{m+1}\|$ is small relative to $\varepsilon N \|A\| \|x_{m+1}\|$

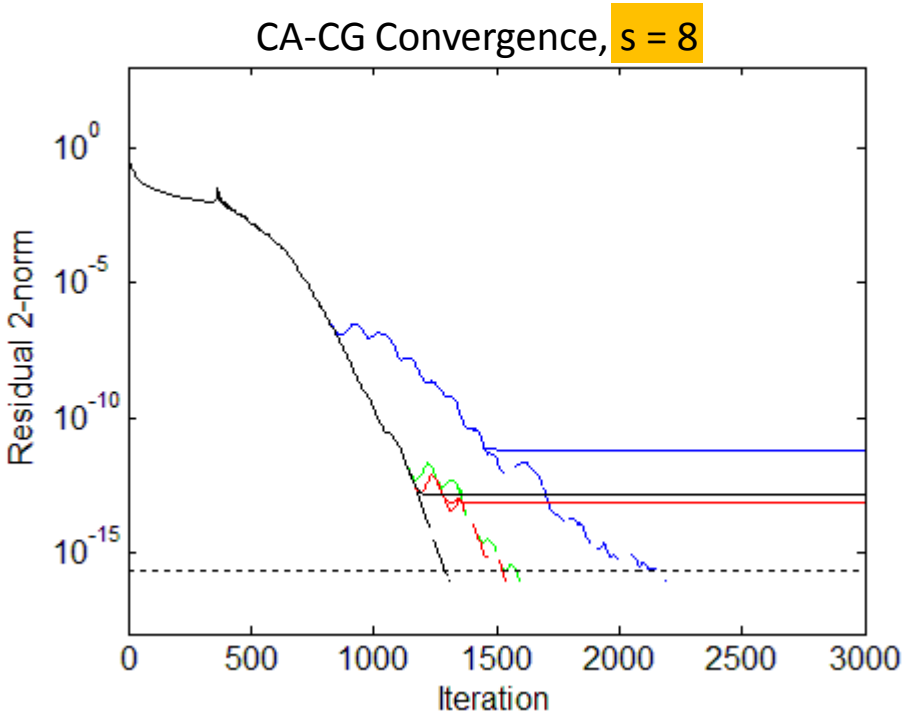
Residual Replacement Strategy for CG

- van der Vorst and Ye (1999): Improve accuracy by replacing **updated residual** $\boldsymbol{r_{m+1}}$ by the **true residual** $\boldsymbol{b - Ax_{m+1}}$ in certain iterations, combined with group update.
- Choose when to replace r_{m+1} with $b - Ax_{m+1}$ to meet two constraints:
  1. **Replace often enough** so that at termination, $\|b - Ax_{m+1} - r_{m+1}\|$ is small relative to $\varepsilon N \|A\| \|x_{m+1}\|$
  2. **Don't replace so often** that original convergence mechanism of updated residuals is destroyed (avoid large perturbations to finite precision CG recurrence)

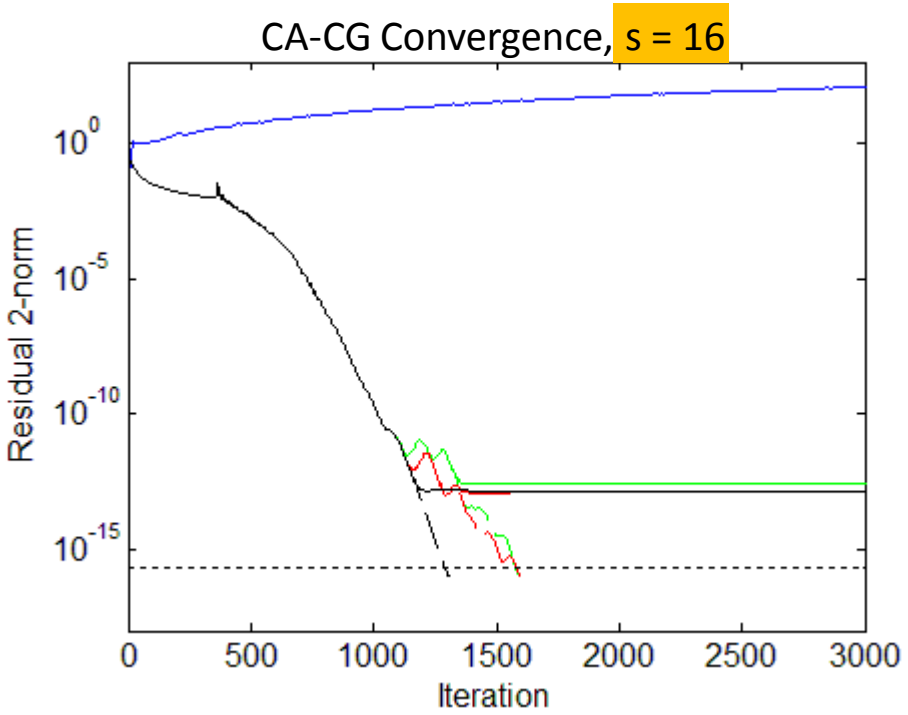
Residual Replacement Strategy for CG

- van der Vorst and Ye (1999): Improve accuracy by replacing **updated residual** r_{m+1} by the **true residual** $b - Ax_{m+1}$ in certain iterations, combined with group update.
- Choose when to replace r_{m+1} with $b - Ax_{m+1}$ to meet two constraints:
  1. **Replace often enough** so that at termination, $\|b - Ax_{m+1} - r_{m+1}\|$ is small relative to $\varepsilon N \|A\| \|x_{m+1}\|$
  2. **Don't replace so often** that original convergence mechanism of updated residuals is destroyed (avoid large perturbations to finite precision CG recurrence)
- We can implement an analogous strategy for CA-CG and CA-BICG based on derived bound on deviation of residuals
  - Estimating quantities in bound has **negligible cost** → residual replacement strategy does not asymptotically increase communication or computation!

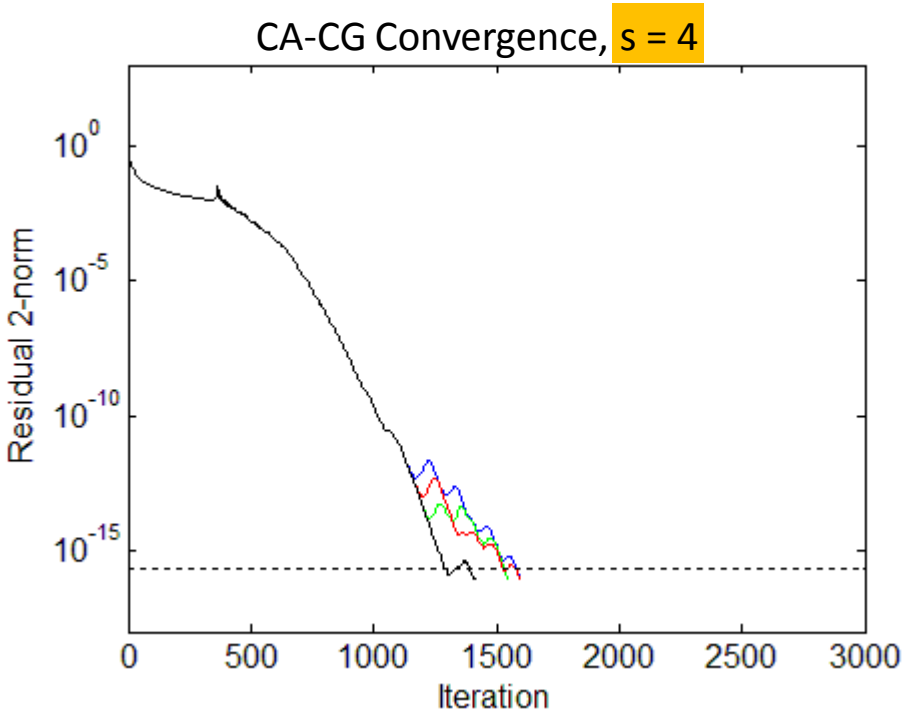


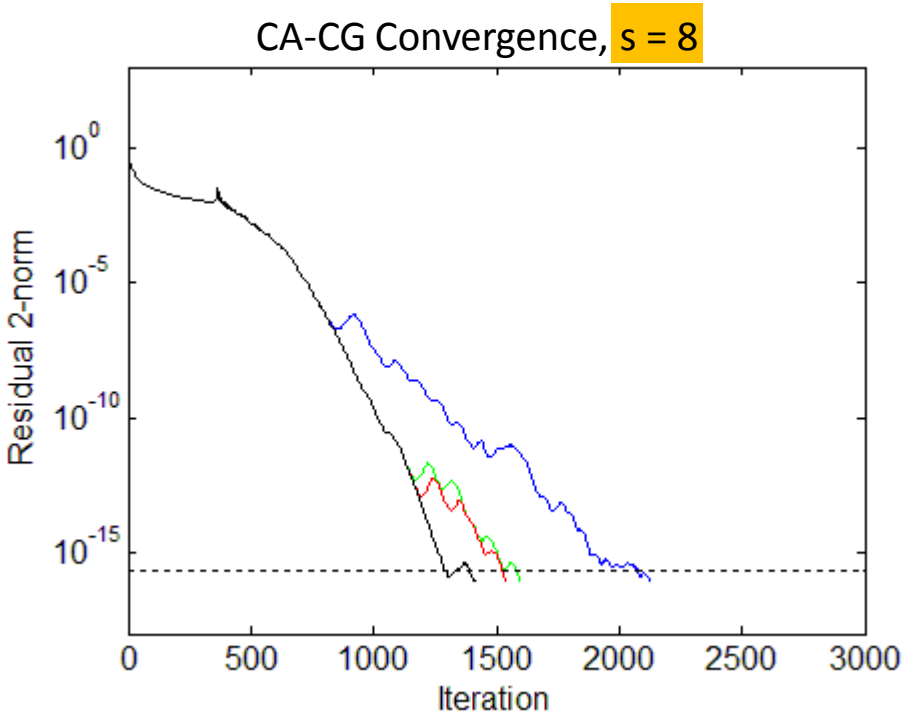


CG true
CG updated
CA-CG (monomial) true
CA-CG (monomial) updated
CA-CG (Newton) true
CA-CG (Newton) updated
CA-CG (Chebyshev) true
CA-CG (Chebyshev) updated

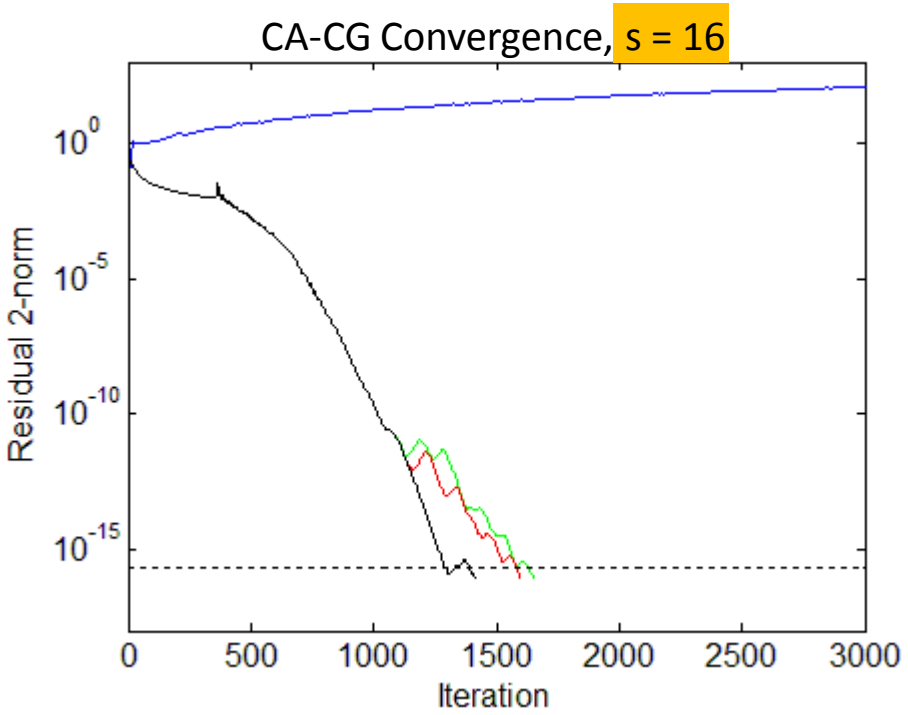


Model Problem: 2D Poisson (5-pt stencil),
$n = 512^2$, nnz $\approx 10^6$, $\kappa(A) \approx 10^4$
$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$

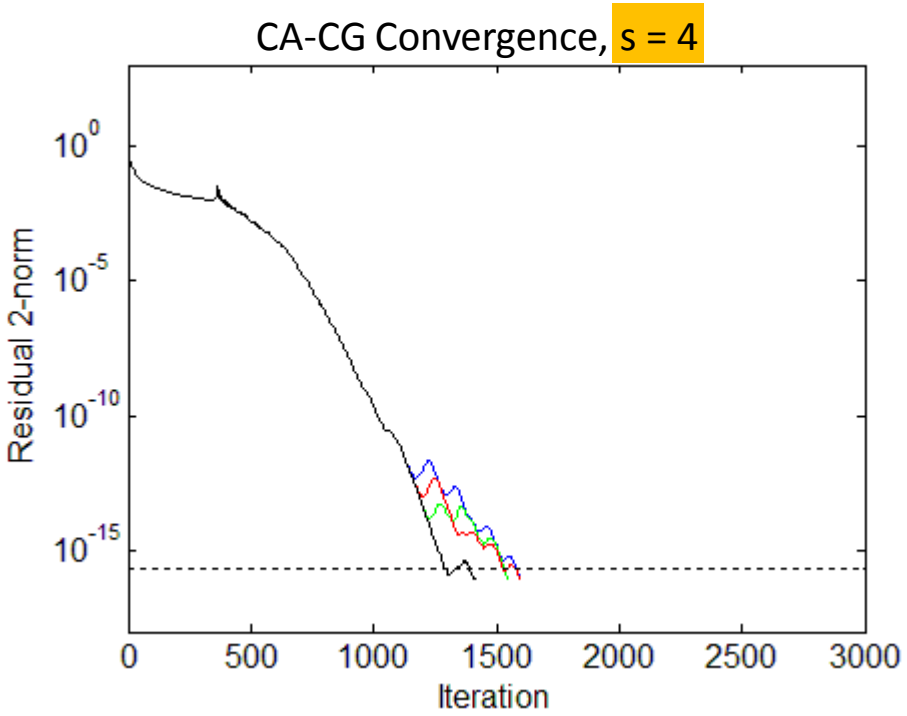


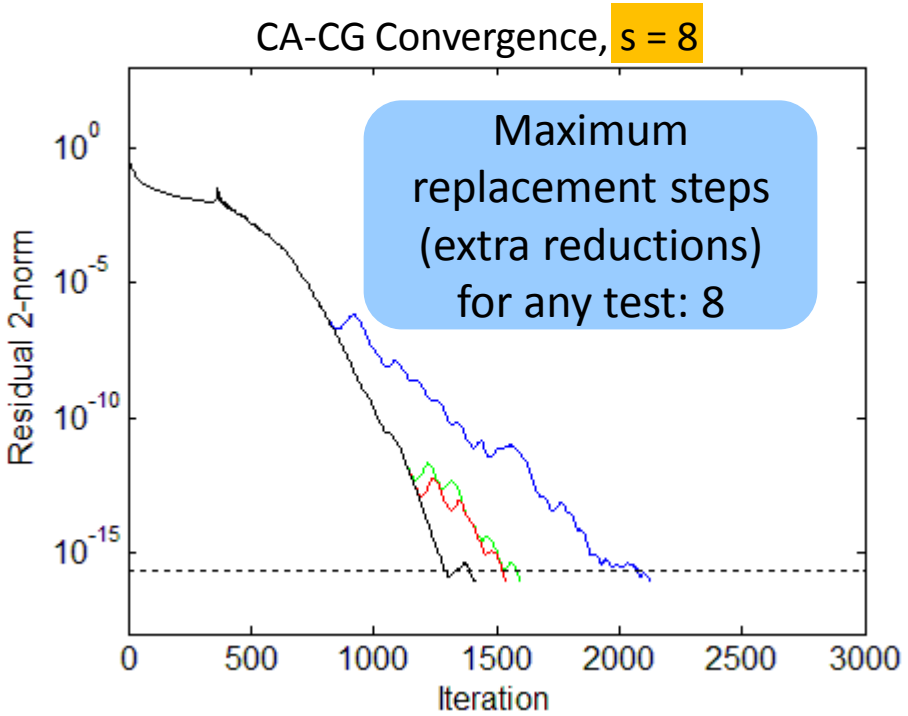


- CG-**RR** true
- CG-**RR** updated
- CA-CG-**RR** (monomial) true
- CA-CG-**RR** (monomial) updated
- CA-CG-**RR** (Newton) true
- CA-CG-**RR** (Newton) updated
- CA-CG-**RR** (Chebyshev) true
- CA-CG-**RR** (Chebyshev) updated

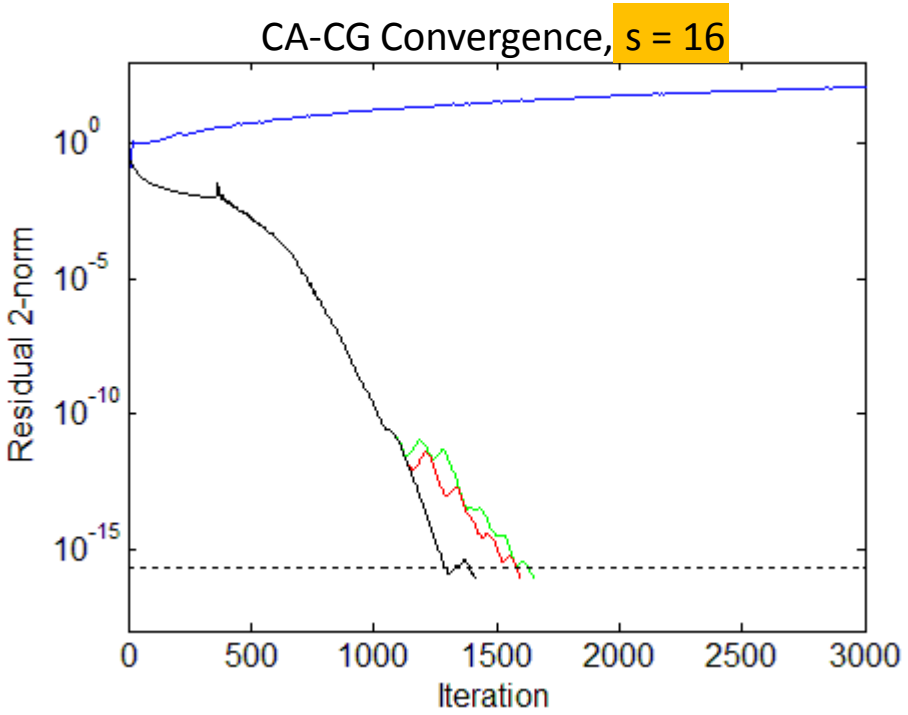


Model Problem: 2D Poisson (5-pt stencil),
$n = 512^2$, nnz $\approx 10^6$, $\kappa(A) \approx 10^4$
$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$



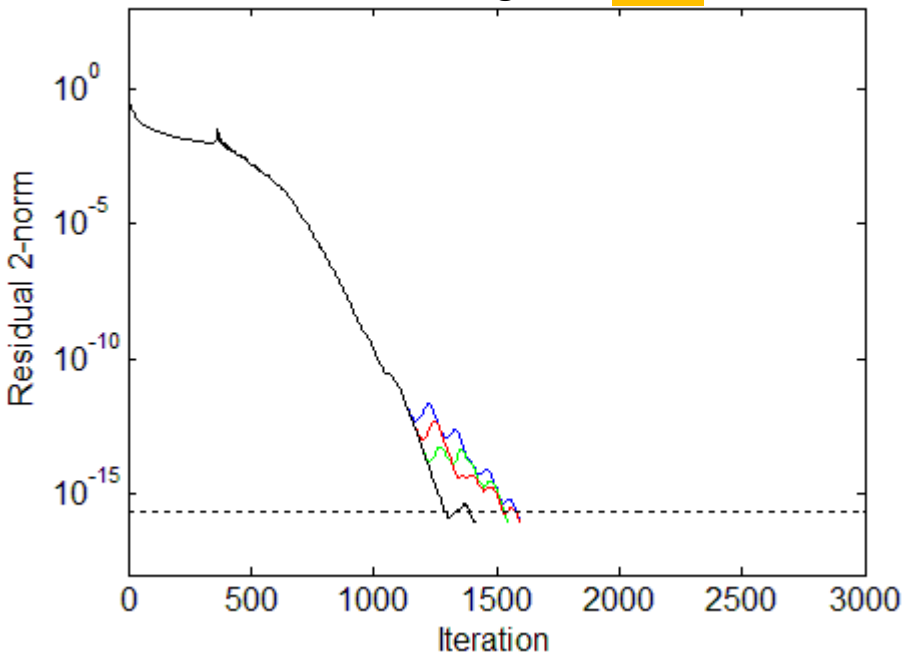


- CG-**RR** true
- CG-**RR** updated
- CA-CG-**RR** (monomial) true
- CA-CG-**RR** (monomial) updated
- CA-CG-**RR** (Newton) true
- CA-CG-**RR** (Newton) updated
- CA-CG-**RR** (Chebyshev) true
- CA-CG-**RR** (Chebyshev) updated

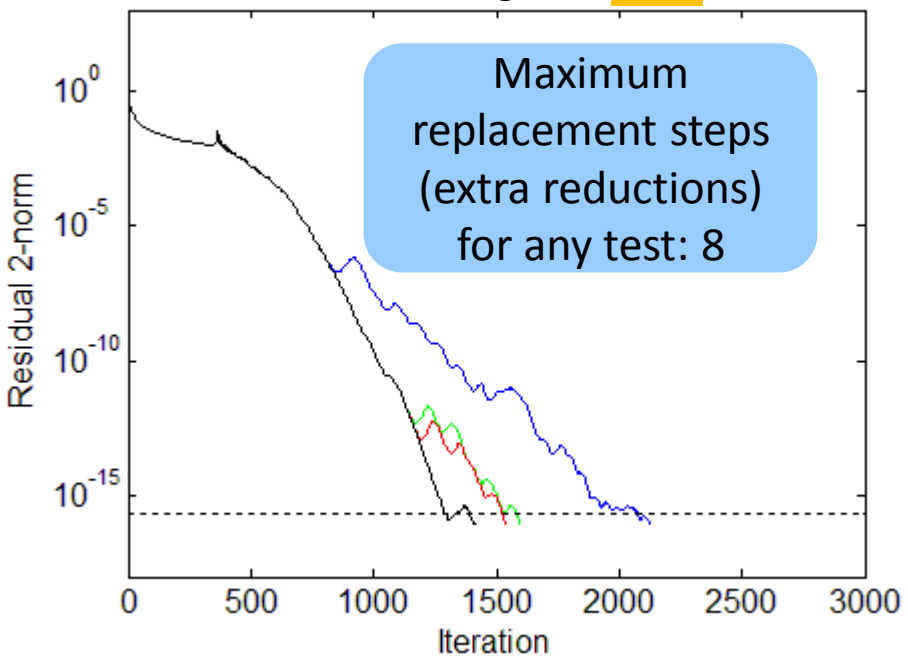


Model Problem: 2D Poisson (5-pt stencil),
$n = 512^2$, nnz $\approx 10^6$, $\kappa(A) \approx 10^4$
$b = A(1\sqrt{n} \cdot \text{ones}(n, 1))$

CA-CG Convergence, s = 4



CA-CG Convergence, s = 8

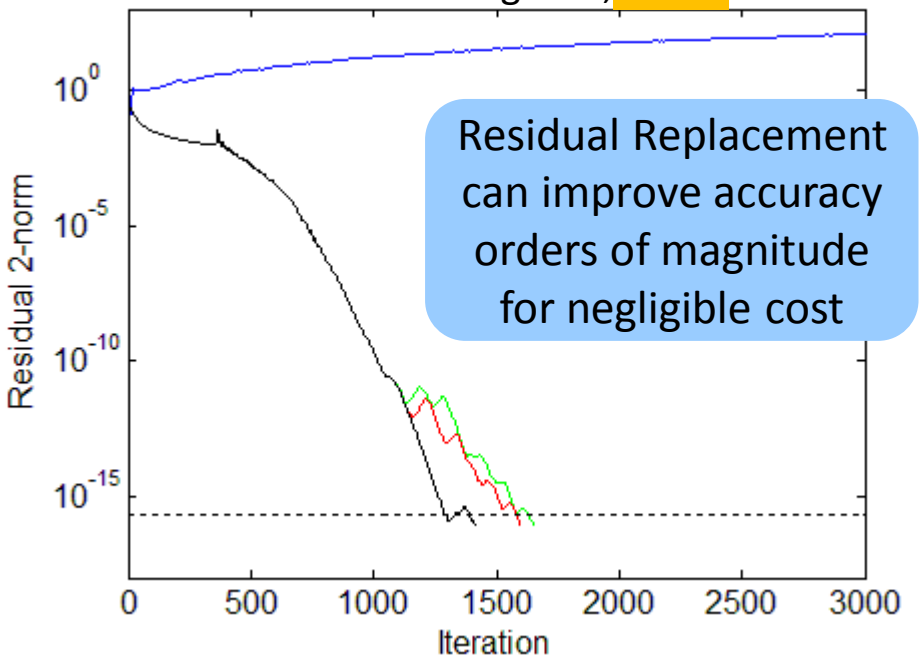


Maximum replacement steps (extra reductions) for any test: 8

- CG-**RR** true
- CG-**RR** updated
- CA-CG-**RR** (monomial) true
- CA-CG-**RR** (monomial) updated
- CA-CG-**RR** (Newton) true
- CA-CG-**RR** (Newton) updated
- CA-CG-**RR** (Chebyshev) true
- CA-CG-**RR** (Chebyshev) updated

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

CA-CG Convergence, s = 16



Residual Replacement can improve accuracy orders of magnitude for negligible cost

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.

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 CA-Lanczos?

Paige's Lanczos Convergence Analysis

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

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

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

Paige's Lanczos Convergence Analysis

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

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

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

Classic Lanczos rounding error result of Paige (1976):

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

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

where $\sigma \equiv \|A\|_2$, $\theta\sigma \equiv \||A|\|_2$, $\varepsilon_0 \equiv 2\varepsilon(n+4)$, and $\varepsilon_1 \equiv 2\varepsilon(N\theta + 7)$

Paige's Lanczos Convergence Analysis

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

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

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

Classic Lanczos rounding error result of Paige (1976):

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

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

where $\sigma \equiv \|A\|_2$, $\theta\sigma \equiv \||A|\|_2$, $\varepsilon_0 \equiv 2\varepsilon(n+4)$, and $\varepsilon_1 \equiv 2\varepsilon(N\theta + 7)$

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

Paige's Lanczos Convergence Analysis

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

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

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

Classic Lanczos rounding error result of Paige (1976):

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

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

where $\sigma \equiv \|A\|_2$, $\theta\sigma \equiv \||A|\|_2$, $\varepsilon_0 \equiv 2\varepsilon(n+4)$, and $\varepsilon_1 \equiv 2\varepsilon(N\theta + 7)$

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

→ These results form the basis for Paige's influential results in (Paige, 1980).

CA-Lanczos Convergence Analysis

For CA-Lanczos, we have:

for $i \in \{1, \ldots, m{=}sk{+}j\}$,

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

$$\varepsilon_0 \equiv 2\varepsilon(n{+}11s{+}15)\,\Gamma^2 = O(\varepsilon n\Gamma^2),$$

$$\varepsilon_1 \equiv 2\varepsilon\big((\mathrm{N}{+}2s{+}5)\theta + (4s{+}9)\tau + 10s{+}16\big)\Gamma = O(\varepsilon N\theta\Gamma),$$

where $\sigma \equiv \|A\|_2$, $\theta\sigma \equiv \||A|\|_2$, $\tau\sigma \equiv \max_{\ell\le k}\||B_\ell|\|_2$, and

$$\Gamma \le \max_{\ell\le k}\|Y_\ell^+\|_2 \cdot \||Y_\ell|\|_2 \le (2s{+}1)\cdot\max_{\ell\le k}\kappa(Y_\ell)\,.$$

CA-Lanczos Convergence Analysis

For CA-Lanczos, we have:

for $i \in \{1, \ldots, m{=}sk{+}j\}$,

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

$\varepsilon_0 \equiv 2\varepsilon(n{+}11s{+}15)\,\Gamma^2 = O(\varepsilon n\Gamma^2)$, ← (vs. $O(\varepsilon n)$ for Lanczos)

$\varepsilon_1 \equiv 2\varepsilon\big((\mathrm{N}{+}2s{+}5)\theta + (4s{+}9)\tau + 10s{+}16\big)\Gamma = O(\varepsilon N\theta\Gamma)$, ← (vs. $O(\varepsilon N\theta)$ for Lanczos)

where $\sigma \equiv \|A\|_2$, $\theta\sigma \equiv \||A|\|_2$, $\tau\sigma \equiv \max_{\ell\le k}\||B_\ell|\|_2$, and

$$\Gamma \le \max_{\ell\le k}\|Y_\ell^+\|_2 \cdot \||Y_\ell|\|_2 \le (2s{+}1)\cdot\max_{\ell\le k}\kappa(Y_\ell)\,.$$

The Amplification Term Γ

- Roundoff errors in CA variant follow same pattern as classical variant, but amplified by factor of Γ or Γ^2
  - **Theoretically confirms empirical observations** on importance of basis conditioning (dating back to late '80s)

- A loose bound for the amplification term:

$$\Gamma \leq \max_{\ell \leq k} \|\mathcal{Y}_\ell^+\|_2 \cdot \||\mathcal{Y}_\ell|\|_2 \leq (2s+1) \cdot \max_{\ell \leq k} \kappa(\mathcal{Y}_\ell)$$

- What we really need: $\||\mathcal{Y}||y'|\|_2 \leq \Gamma \|\mathcal{Y}y'\|_2$ to hold for the computed basis $\mathcal{Y}$ and coordinate vector y' in every bound.

- **Tighter bound on Γ possible**; requires some 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

$$\Gamma_{k,j} \equiv \max_{x \in \{\hat{w}'_{k,j}, \hat{u}'_{k,j}, \hat{v}'_{k,j}, \hat{v}'_{k,j-1}\}} \frac{\left\|\left|\hat{\mathcal{Y}}_k\right||x|\right\|_2}{\left\|\hat{\mathcal{Y}}_k x\right\|_2}$$

Results for CA-Lanczos

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

  loss of orthogonality $\rightarrow$ eigenvalue convergence

  hold for CA-Lanczos?

Results for CA-Lanczos

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

  loss of orthogonality $\rightarrow$ eigenvalue convergence

  hold for CA-Lanczos?
- The answer is **YES!** …but

Results for CA-Lanczos

- Back to our question: Do Paige's results, e.g.,
  loss of orthogonality $\rightarrow$ eigenvalue convergence
  hold for CA-Lanczos?
- The answer is **YES!** …but
- Only if:
  - $\varepsilon_0 \equiv 2\varepsilon(n+11s+15)\,\Gamma^2 \leq \frac{1}{12}$
    - i.e., $\Gamma \leq \left(24\epsilon(n+11s+15)\right)^{-1/2} = O(n\epsilon)^{-1/2}$
  - Otherwise, e.g., can lose orthogonality due to computation with (numerically) rank-deficient basis

Results for CA-Lanczos

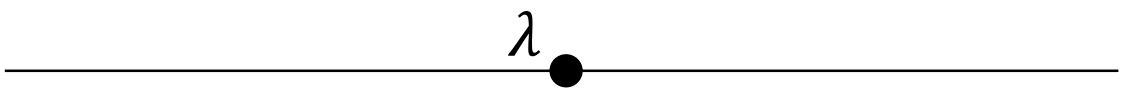
- Back to our question: Do Paige's results, e.g.,
  loss of orthogonality $\rightarrow$ eigenvalue convergence
  hold for CA-Lanczos?
- The answer is **YES!** …but
- Only if:
  - $\varepsilon_0 \equiv 2\varepsilon(n+11s+15)\,\Gamma^2 \leq \frac{1}{12}$
    - i.e., $\Gamma \leq \left(24\epsilon(n+11s+15)\right)^{-1/2} = O(n\epsilon)^{-1/2}$
  - Otherwise, e.g., can lose orthogonality due to computation with (numerically) rank-deficient basis
- Take-away: we can use this bound on Γ to design a better algorithm!
  - Mixed precision, selective reorthogonalization, dynamic basis size, etc.

Extending the results of Greenbaum (1989):

Eigenvalue approximations generated at each step by a perturbed Lanczos recurrence for A are equal to those generated by exact Lanczos applied to a matrices whose eigenvalues lie within intervals about the eigenvalues of A.

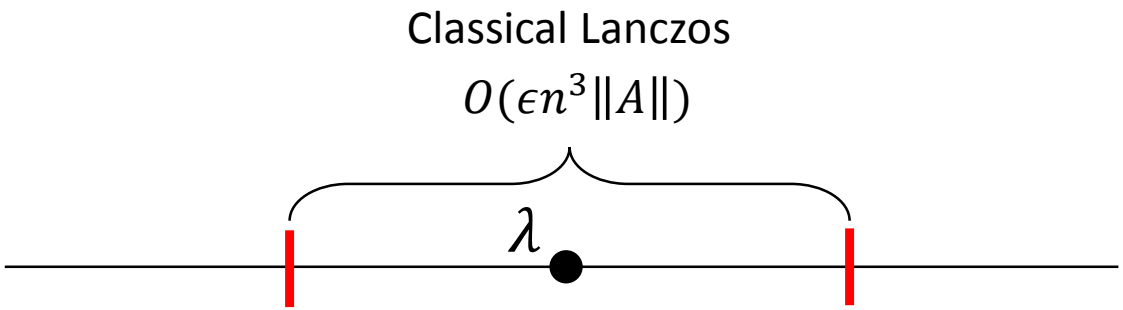
Extending the results of Greenbaum (1989):

Eigenvalue approximations generated at each step by a perturbed Lanczos recurrence for A are equal to those generated by exact Lanczos applied to a matrices whose eigenvalues lie within intervals about the eigenvalues of A.



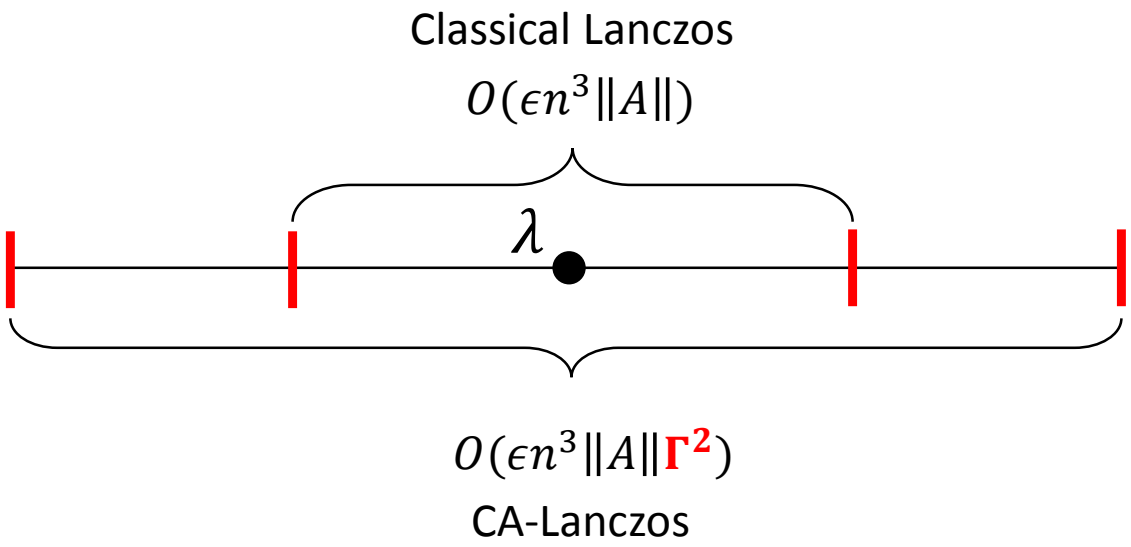
Extending the results of Greenbaum (1989):

Eigenvalue approximations generated at each step by a perturbed Lanczos recurrence for A are equal to those generated by exact Lanczos applied to a matrices whose eigenvalues lie within intervals about the eigenvalues of A.



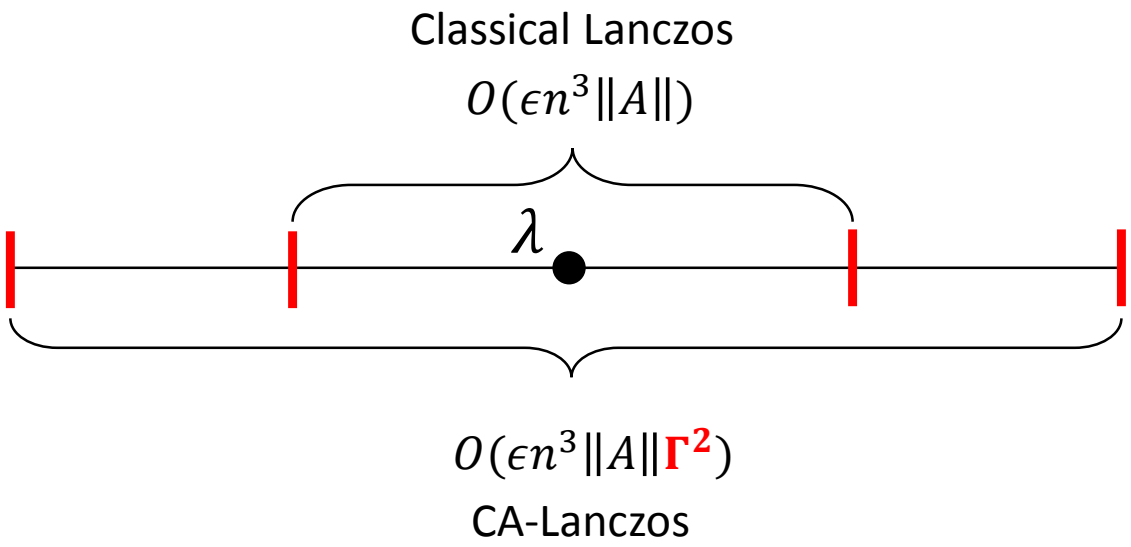
Extending the results of Greenbaum (1989):

Eigenvalue approximations generated at each step by a perturbed Lanczos recurrence for A are equal to those generated by exact Lanczos applied to a matrices whose eigenvalues lie within intervals about the eigenvalues of A.



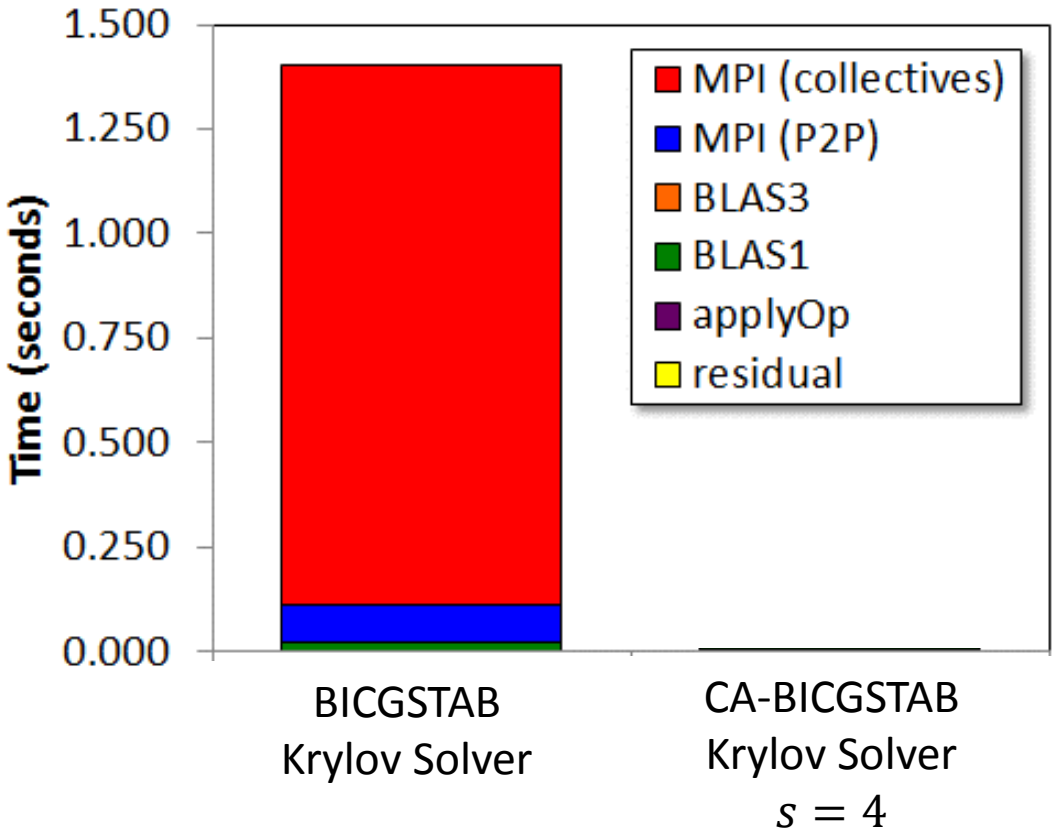
Extending the results of Greenbaum (1989):

Eigenvalue approximations generated at each step by a perturbed Lanczos recurrence for A are equal to those generated by exact Lanczos applied to a matrices whose eigenvalues lie within intervals about the eigenvalues of A.



Ongoing work…

- Timing for coarse grid solves in geometric multigrid method
- 3D Helmholtz equation with $n = 1.6 \cdot 10^6$
- 24K cores on NERSC's Hopper (Cray XE6)



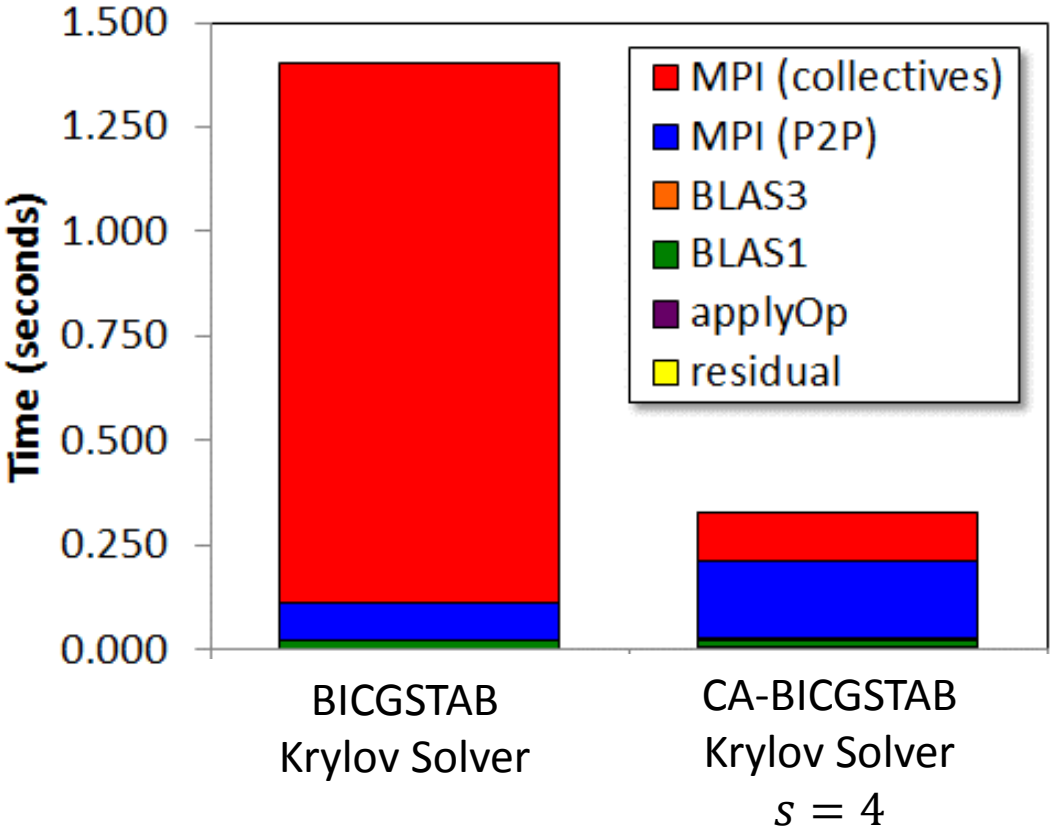
Problem specifics:

$Lu = (a\alpha - b\nabla \cdot \beta\nabla)u = f$

$\alpha = \beta = 1.0, a = b = 0.9$

- Periodic boundary conds.
- RHS: 3D triangle wave w/period spanning entire domain

- Timing for coarse grid solves in geometric multigrid method
- 3D Helmholtz equation with $n = 1.6 \cdot 10^6$
- 24K cores on NERSC's Hopper (Cray XE6)



Problem specifics:

$Lu = (a\alpha - b\nabla \cdot \beta\nabla)u = f$

$\alpha = \beta = 1.0, a = b = 0.9$

- Periodic boundary conds.
- RHS: 3D triangle wave w/period spanning entire domain

4.2x speedup in Krylov solve!

Future Directions

Broad research agenda: Design methods for large-scale problems that optimize performance subject to application-specific numerical constraints

- **New Algorithms/Applications**
  - Application of communication-avoiding ideas and solvers to new computational science domains
  - Design of new high-performance preconditioners
- **Finite-Precision Analysis**
  - Bounds on stability and convergence for other Krylov methods (particularly in the nonsymmetric case)
  - Extension of “Backwards-like” error analyses
- **Improving Usability**
  - Automating parameter selection via “numerical auto-tuning”
  - Integration into high-performance libraries

Thank you!

Happy Birthday, Jim!

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