MS58: Approaches to Reducing Communication in Krylov Subspace Methods

Organizers: Laura Grigori (INRIA) and Erin Carson (NYU)

Talks:

1. The s-Step Lanczos Method and its Behavior in Finite Precision (***Erin Carson***, James W. Demmel)
2. Enlarged Krylov Subspace Methods for Reducing Communication (***Sophie M. Moufawad***, Laura Grigori, Frederic Nataf)
3. Preconditioning Communication-Avoiding Krylov Methods **(*Siva Rajamanickam*,** Ichitaro Yamazaki, Andrey Prokopenko, Erik G. Boman, Michael Heroux, Jack J. Dongarra)
4. Sparse Approximate Inverse Preconditioners for Communication-Avoiding Bicgstab Solvers **(*Maryam Mehri Dehnavi*,** Erin Carson, Nicholas Knight, James W. Demmel, David Fernandez)

The s-Step Lanczos Method and its Behavior in Finite Precision

Erin Carson, NYU

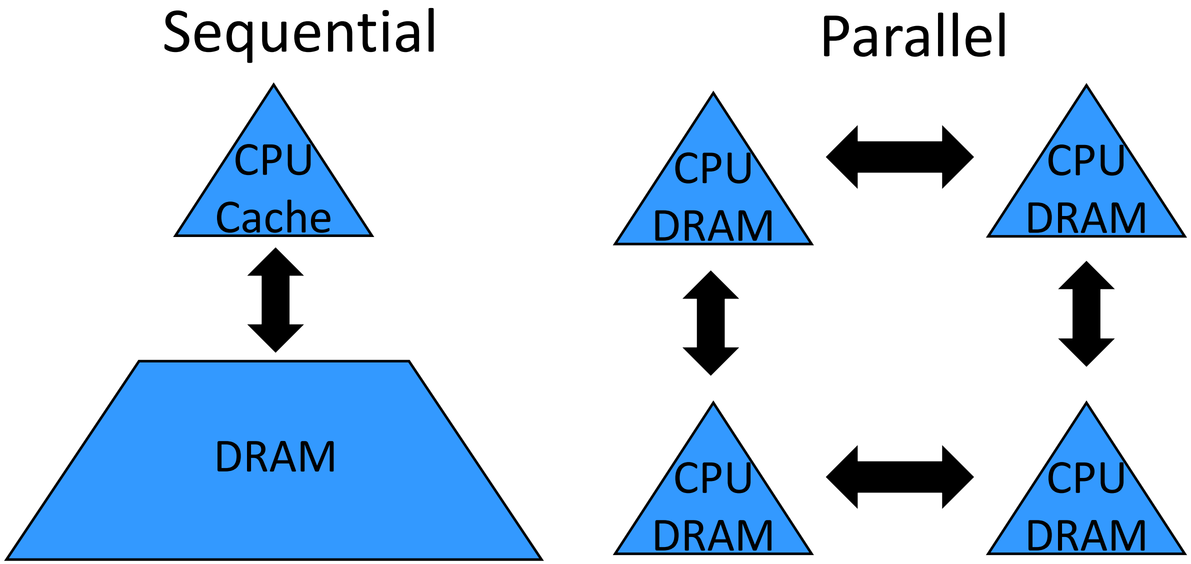
James Demmel, UC Berkeley

SIAM LA ‘15

October 30, 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

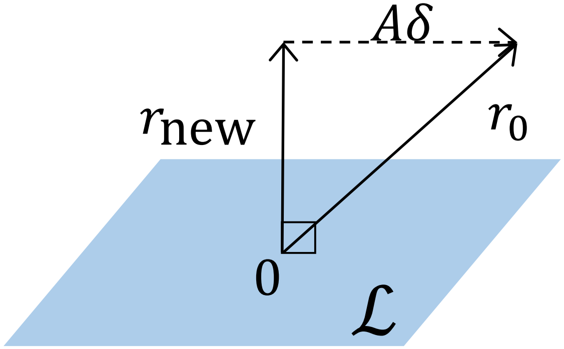
- General class of iterative solvers: used for linear systems, eigenvalue problems, singular value problems, least squares, etc.

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

- Projection process onto the expanding **Krylov subspace**

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

- 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:

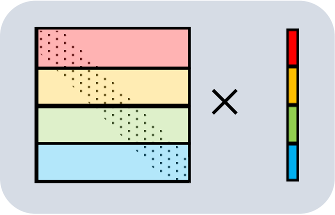
Krylov Solvers: Limited by Communication

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



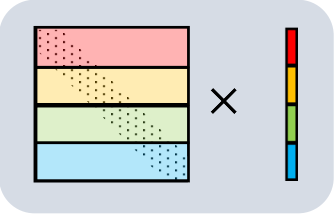
Krylov Solvers: Limited by Communication

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

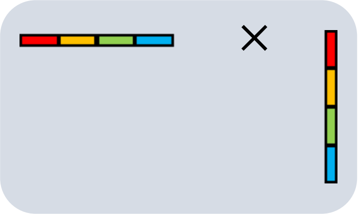


“Orthogonalize (with respect to some $\mathcal{L}_m$)”

$\rightarrow$ Inner products

Parallel: global reduction (All-Reduce)

Sequential: multiple reads/writes to slow memory



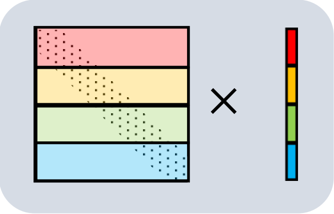
Krylov Solvers: Limited by Communication

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

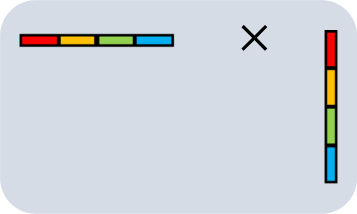


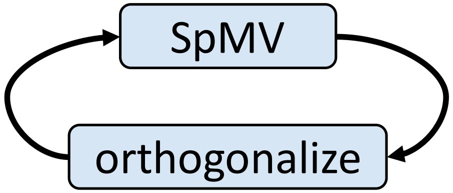
"Orthogonalize (with respect to some $\mathcal{L}_m$)"

$\rightarrow$ Inner products

Parallel: global reduction (All-Reduce)

Sequential: multiple reads/writes to slow memory





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

The Classical Lanczos Method

```
Given: initial vector $v_1$ with $||v_1||_2 = 1$
$u_1 = Av_1$
**for** $i = 1, 2, \dots,$ until convergence **do**
    $\alpha_i = v_i^T u_i$
    $w_i = u_i - \alpha_i v_i$
    $\beta_{i+1} = ||w_i||_2$
    $v_{i+1} = w_i / \beta_{i+1}$
    $u_{i+1} = Av_{i+1} - \beta_{i+1} v_i$
**end for**
```

The Classical Lanczos Method

```
Given: initial vector $v_1$ with $||v_1||_2 = 1$
$u_1 = Av_1$
**for** $i = 1, 2, \ldots,$ until convergence **do**
        $\alpha_i = v_i^T u_i$
        $w_i = u_i - \alpha_i v_i$
        $\beta_{i+1} = ||w_i||_2$
        $v_{i+1} = w_i / \beta_{i+1}$
        $u_{i+1} = \boxed{Av_{i+1}} - \beta_{i+1} v_i$
**end for**
```

SpMV

The Classical Lanczos Method

```
Given: initial vector $v_1$ with $||v_1||_2 = 1$
$u_1 = Av_1$
**for** $i = 1, 2, \dots,$ until convergence **do**
        $\alpha_i = \boxed{v_i^T u_i}$
        $w_i = u_i - \alpha_i v_i$
        $\beta_{i+1} = \boxed{||w_i||_2}$
        $v_{i+1} = w_i / \beta_{i+1}$
        $u_{i+1} = \boxed{Av_{i+1}} - \beta_{i+1} v_i$
**end for**
```

Inner products

SpMV

Communication-Avoiding KSMs

- Idea: Compute blocks of s iterations at once
  - Communicate every s iterations instead of every iteration
  - **Reduces communication cost by $\boldsymbol{O(s)}$!**
    - (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 $\boldsymbol{O(s)}$!**
        - (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 $\boldsymbol{O(s)}$!**
    - (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-Lanczos

- Main idea: Unroll iteration loop by a factor of s; split iteration loop into an outer loop (k) and an inner loop (j)
- Key observation: starting at some iteration $i \equiv sk + j$,

$$v_{sk+j},\ u_{sk+j} \in \mathcal{K}_{s+1}(A, v_{sk+1}) + \mathcal{K}_{s+1}(A, u_{sk+1}) \quad \text{for} \quad j \in \{1, \dots, s+1\}$$

Communication-Avoiding KSMs: CA-Lanczos

- Main idea: Unroll iteration loop by a factor of s; split iteration loop into an outer loop (k) and an inner loop (j)
- Key observation: starting at some iteration $i \equiv sk + j$,

$$v_{sk+j},\ u_{sk+j} \in \mathcal{K}_{s+1}(A, v_{sk+1}) + \mathcal{K}_{s+1}(A, u_{sk+1}) \quad \text{for} \quad j \in \{1, \dots, s+1\}$$

Outer loop k: Communication step

Expand solution space s dimensions at once

- Compute "basis matrix" $\mathcal{Y}_k$ with columns spanning

$$\mathcal{K}_{s+1}(A, v_{sk+1}) + \mathcal{K}_{s+1}(A, u_{sk+1})$$

- 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:

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

- Communication cost of **one global reduction**

Communication-Avoiding KSMs: CA-Lanczos

Inner loop: Computation steps, no communication!

Perform s iterations of updates

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

$$v_{sk+j} = \mathcal{Y}_k v'_{k,j}\,, \qquad u_{sk+j} = \mathcal{Y}_k u'_{k,j}, \qquad w_{sk+j} = \mathcal{Y}_k w'_j$$

Communication-Avoiding KSMs: CA-Lanczos

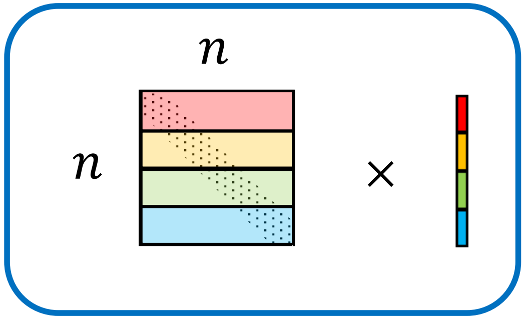
Inner loop: Computation steps, no communication!

Perform s iterations of updates

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

$$v_{sk+j} = \mathcal{Y}_k v'_{k,j}\,, \qquad u_{sk+j} = \mathcal{Y}_k u'_{k,j}, \qquad w_{sk+j} = \mathcal{Y}_k w'_j$$

Av_{i+1}



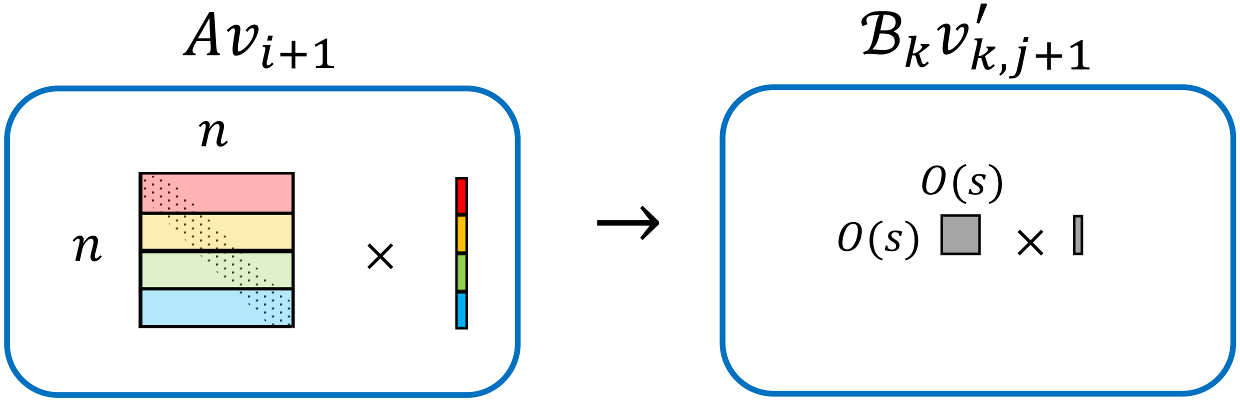
Communication-Avoiding KSMs: CA-Lanczos

Inner loop: Computation steps, no communication!

Perform s iterations of updates

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

$$v_{sk+j} = \mathcal{Y}_k v'_{k,j}\,, \qquad u_{sk+j} = \mathcal{Y}_k u'_{k,j}, \qquad w_{sk+j} = \mathcal{Y}_k w'_j$$



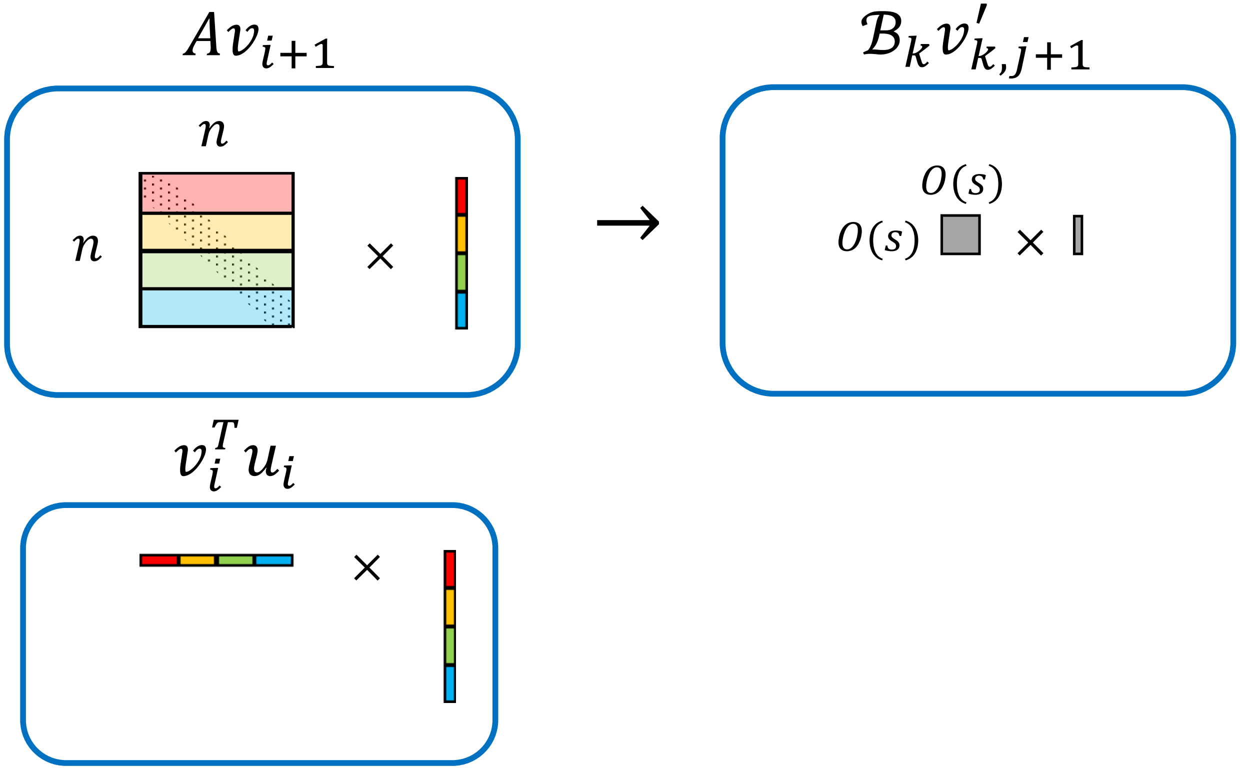
Communication-Avoiding KSMs: CA-Lanczos

Inner loop: Computation steps, no communication!

Perform s iterations of updates

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

$$v_{sk+j} = \mathcal{Y}_k v'_{k,j}\,, \qquad u_{sk+j} = \mathcal{Y}_k u'_{k,j}, \qquad w_{sk+j} = \mathcal{Y}_k w'_j$$



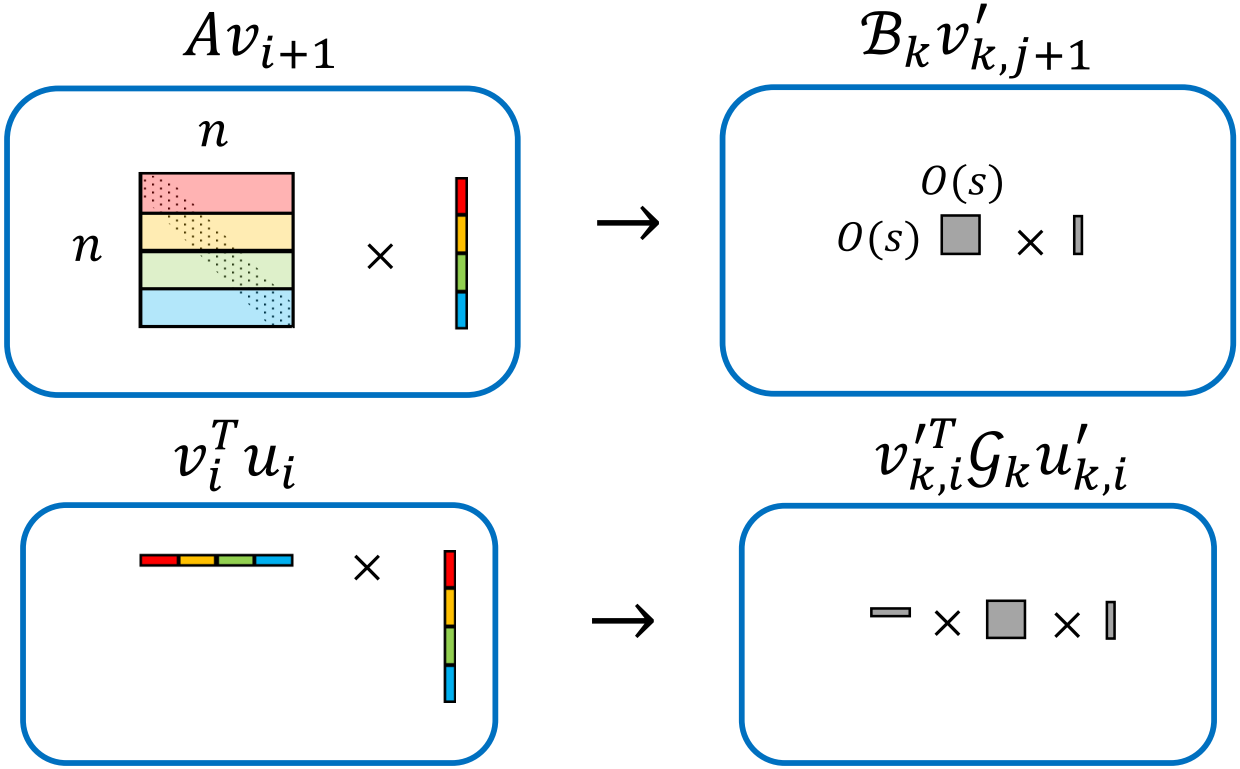
Communication-Avoiding KSMs: CA-Lanczos

Inner loop: Computation steps, no communication!

Perform s iterations of updates

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

$$v_{sk+j} = \mathcal{Y}_k v'_{k,j}\,, \qquad u_{sk+j} = \mathcal{Y}_k u'_{k,j}, \qquad w_{sk+j} = \mathcal{Y}_k w'_j$$



Av_{i+1} → $\mathcal{B}_k v'_{k,j+1}$

$v_i^T u_i$ → $v'^T_{k,i}\mathcal{G}_k u'_{k,i}$

The CA-Lanczos Method

```
Given: initial vector $v_1$ with $||v_1||_2 = 1$
$u_1 = Av_1$
for $k = 0, 1, \dots,$ until convergence do
        Compute $\mathcal{Y}_k$,     compute $\mathcal{G}_k = \mathcal{Y}_k^T \mathcal{Y}_k$
        Let $v'_{k,1} = e_1,\ u'_{k,1} = e_{s+2}$
        for $j = 1, \dots, s$ do
                $\alpha_{sk+j} = v'^T_{k,j} \mathcal{G}_k u'_{k,j}$
                $w'_{k,j} = u'_{k,j} - \alpha_{sk+j} v'_{k,j}$
                $\beta_{sk+j+1} = \left(w'^T_{k,j} \mathcal{G}_k w'_{k,j}\right)^{1/2}$
                $v'_{k,j+1} = w'_{k,j} / \beta_{sk+j+1}$
                $u'_{k,j+1} = \mathcal{B}_k v'_{k,j+1} - \beta_{sk+j+1} v'_{k,j}$
        end for
        Compute $v_{sk+s+1} = \mathcal{Y}_k v'_{k,s+1},\ u_{sk+s+1} = \mathcal{Y}_k u'_{k,s+1}$
end for
```

The CA-Lanczos Method

```
Given: initial vector $v_1$ with $||v_1||_2 = 1$
$u_1 = Av_1$
for $k = 0, 1, \dots,$ until convergence do
        Compute $\mathcal{Y}_k$, compute $\mathcal{G}_k = \mathcal{Y}_k^T \mathcal{Y}_k$
        Let $v'_{k,1} = e_1,\; u'_{k,1} = e_{s+2}$
        for $j = 1, \dots, s$ do
                $\alpha_{sk+j} = v'^T_{k,j} \mathcal{G}_k u'_{k,j}$
                $w'_{k,j} = u'_{k,j} - \alpha_{sk+j} v'_{k,j}$
                $\beta_{sk+j+1} = \left(w'^T_{k,j} \mathcal{G}_k w'_{k,j}\right)^{1/2}$
                $v'_{k,j+1} = w'_{k,j} / \beta_{sk+j+1}$
                $u'_{k,j+1} = \mathcal{B}_k v'_{k,j+1} - \beta_{sk+j+1} v'_{k,j}$
        end for
        Compute $v_{sk+s+1} = \mathcal{Y}_k v'_{k,s+1},\; u_{sk+s+1} = \mathcal{Y}_k u'_{k,s+1}$
end for
```

via CA Matrix Powers Kernel

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

The CA-Lanczos Method

```
Given: initial vector $v_1$ with $\|v_1\|_2 = 1$
$u_1 = Av_1$
for $k = 0, 1, \dots,$ until convergence do
        Compute $\mathcal{Y}_k$, compute $\mathcal{G}_k = \mathcal{Y}_k^T \mathcal{Y}_k$
        Let $v'_{k,1} = e_1,\ u'_{k,1} = e_{s+2}$
        for $j = 1, \dots, s$ do
                $\alpha_{sk+j} = v'^T_{k,j} \mathcal{G}_k u'_{k,j}$
                $w'_{k,j} = u'_{k,j} - \alpha_{sk+j} v'_{k,j}$
                $\beta_{sk+j+1} = \left(w'^T_{k,j} \mathcal{G}_k w'_{k,j}\right)^{1/2}$
                $v'_{k,j+1} = w'_{k,j} / \beta_{sk+j+1}$
                $u'_{k,j+1} = \mathcal{B}_k v'_{k,j+1} - \beta_{sk+j+1} v'_{k,j}$
        end for
        Compute $v_{sk+s+1} = \mathcal{Y}_k v'_{k,s+1},\ u_{sk+s+1} = \mathcal{Y}_k u'_{k,s+1}$
end for
```

via CA Matrix Powers Kernel

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

Local computations: no communication!

Complexity Comparison

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

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

	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: accuracy limited

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: accuracy limited

   2. **Delay of convergence** → Tradeoff: increasing blocking factor s past a certain point: no speedup expected

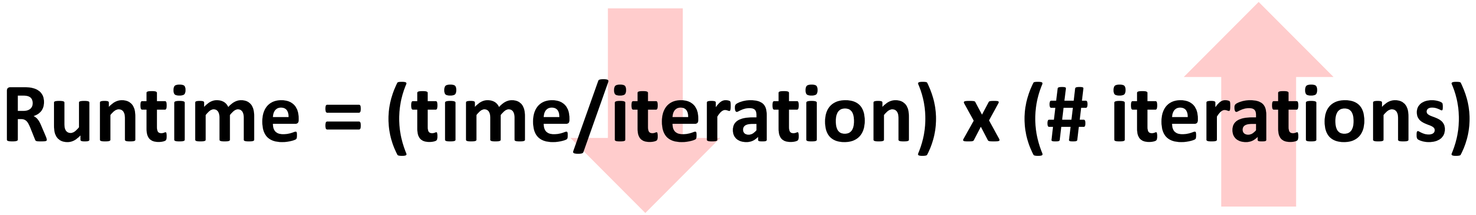
From Theory to Practice

- 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: accuracy limited
  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: accuracy limited
  2. **Delay of convergence** → Tradeoff: increasing blocking factor s past a certain point: no speedup expected



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

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)

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

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

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
$\theta\sigma \equiv \||A|\|_2$

Paige's Lanczos Convergence Analysis

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

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

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

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
$\theta\sigma \equiv \||A|\|_2$

Classic Lanczos (Paige, 1976):

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

Paige’s Lanczos Convergence Analysis

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

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

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

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
$\theta\sigma \equiv \||A|\|_2$

Classic Lanczos (Paige, 1976):

$$\varepsilon_0 = O(\varepsilon n)$$

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

CA-Lanczos:

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

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

Paige's Lanczos Convergence Analysis

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

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

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

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
$\theta\sigma \equiv \||A|\|_2$

Classic Lanczos (Paige, 1976):

$$\varepsilon_0 = O(\varepsilon n)$$

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

CA-Lanczos:

$$\varepsilon_0 = O\left(\varepsilon n \mathbf{\Gamma}^{\mathbf{2}}\right)$$

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

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

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

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

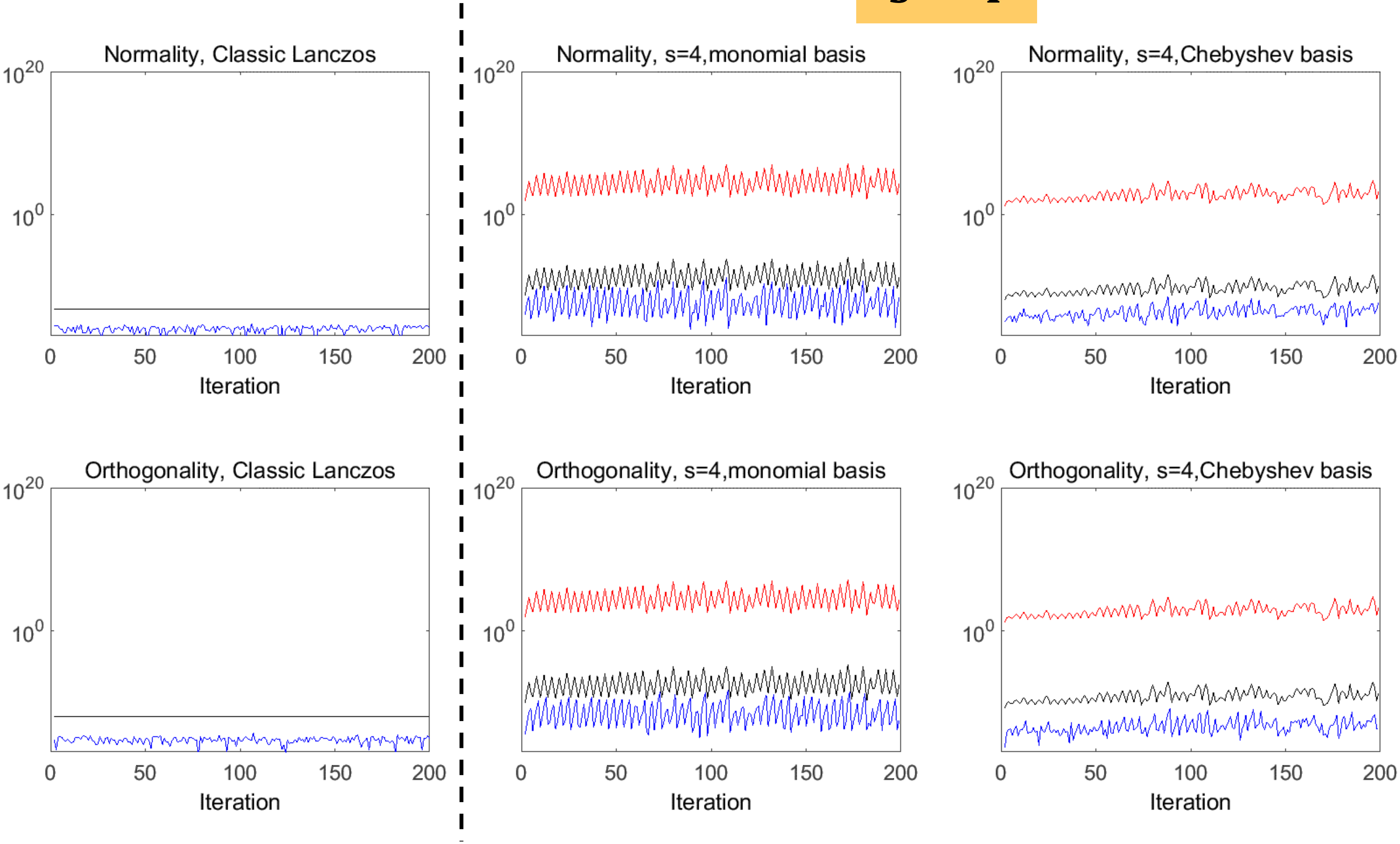
Problem: 2D Poisson,
$n = 256$,
random starting vector

🟦 Computed value
⬛ Bound
🟥 Amplification factor Γ^2

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

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

$s = 4$


Normality, Classic Lanczos

Normality, s=4,monomial basis

Normality, s=4,Chebyshev basis

Orthogonality, Classic Lanczos

Orthogonality, s=4,monomial basis

Orthogonality, s=4,Chebyshev basis

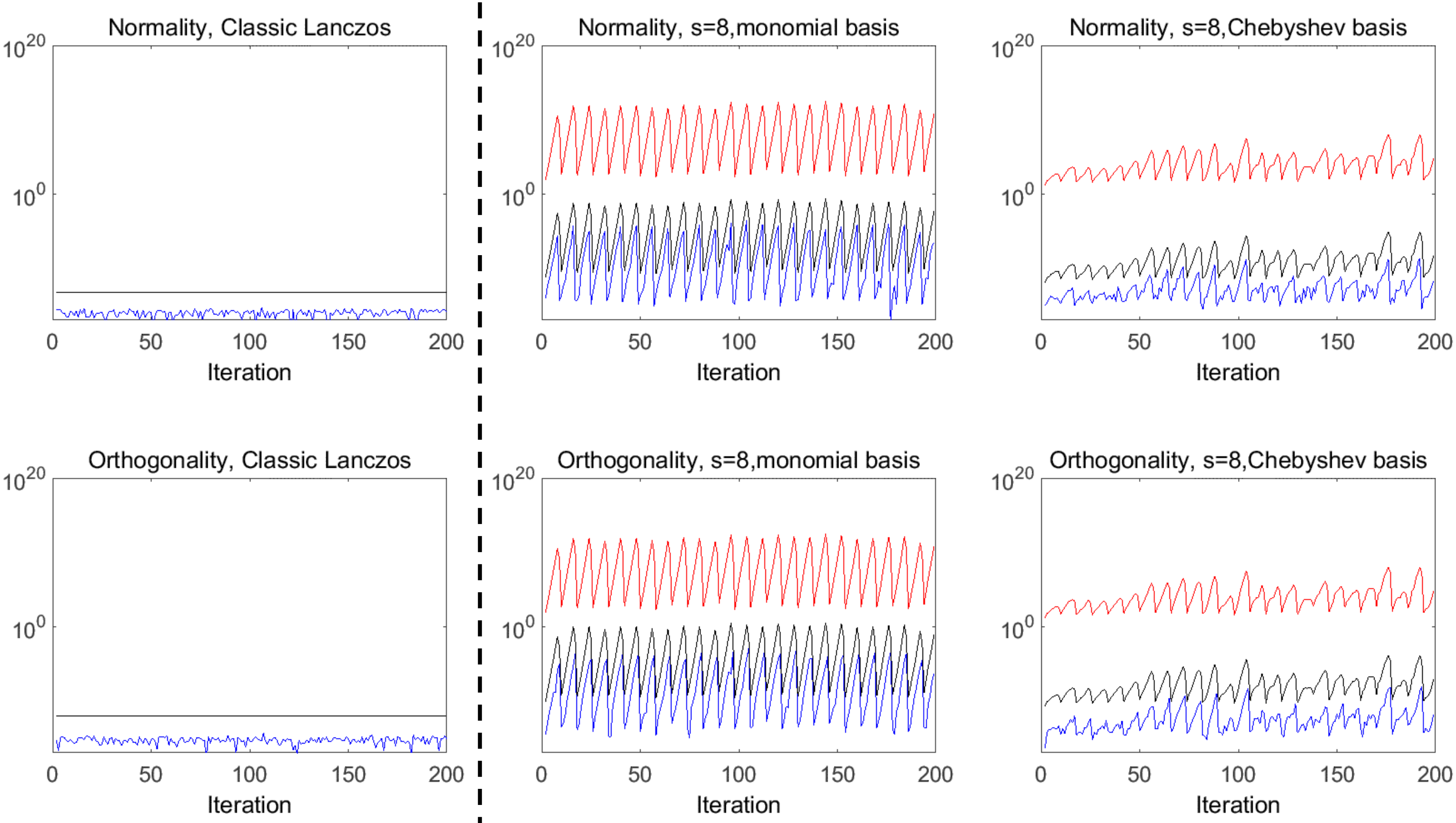
Problem: 2D Poisson,
$n = 256$,
random starting vector

- 🟦 Computed value
- ⬛ Bound
- 🟥 Amplification factor Γ

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

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

$\boldsymbol{s = 8}$



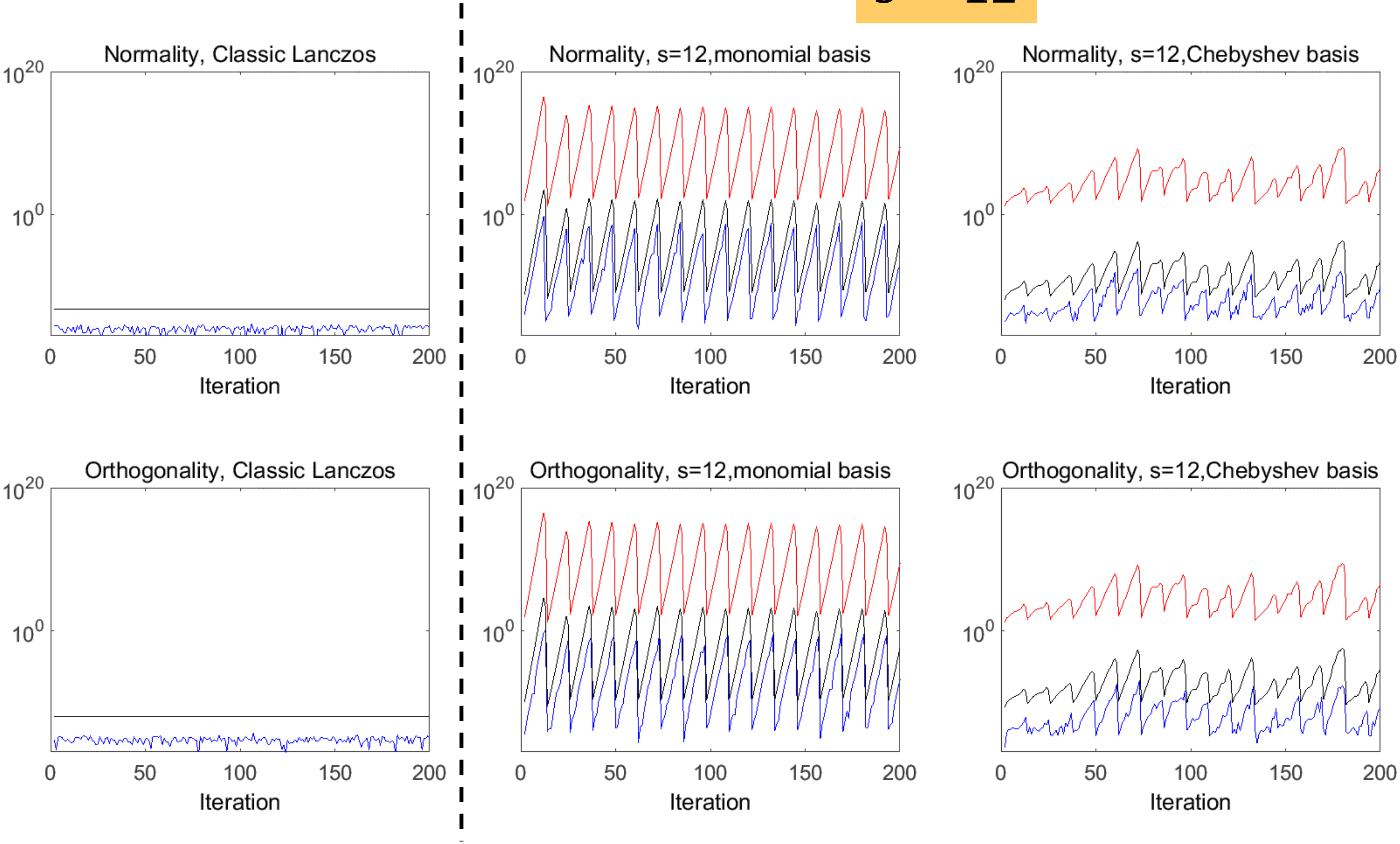
Problem: 2D Poisson,
$n = 256$,
random starting vector

— Computed value
— Bound
— Amplification factor Γ^2

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

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

$\boldsymbol{s = 12}$



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 → 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 → 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.

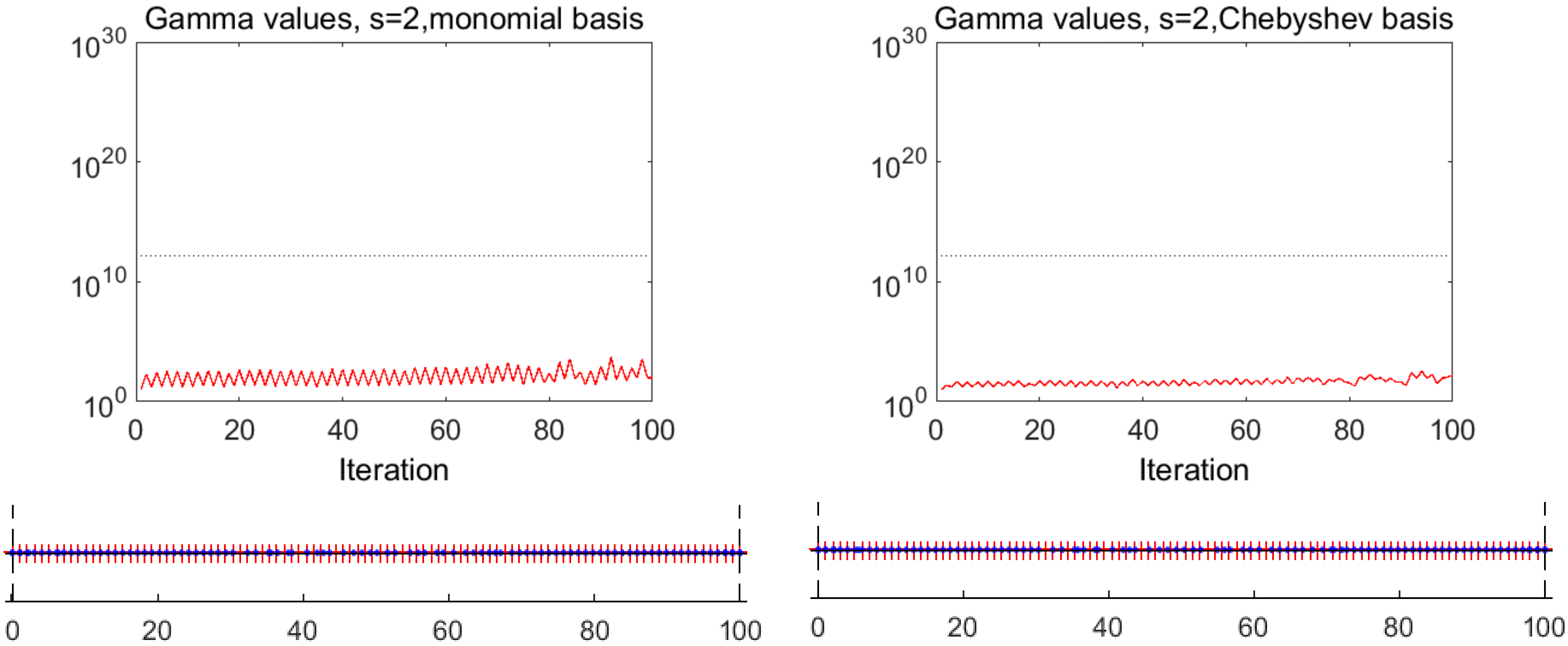
Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector

$\boldsymbol{s = 2}$

Top plots:

— Computed Γ^2

┈ $(24(\epsilon(n + 11s + 15))^{-1}$



Bottom Plots:

+ True eigenvalues

● Computed Ritz values

¦ Bounds on range of computed Ritz values

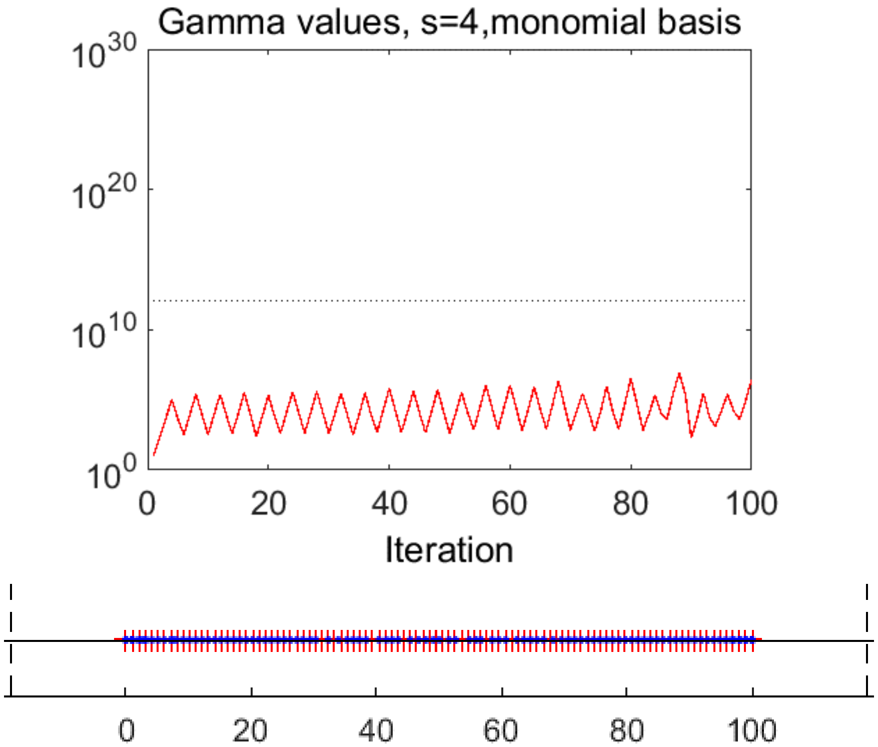
Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector

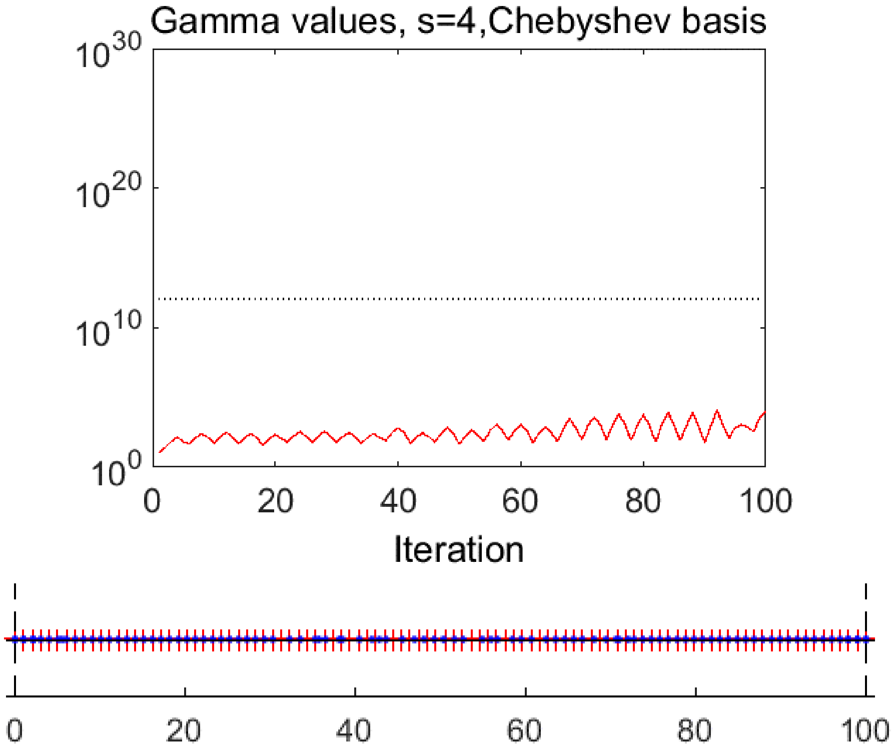
$\boldsymbol{s = 4}$

Top plots:

— Computed Γ^2

⋯ $(24(\epsilon(n + 11s + 15))^{-1}$





Bottom Plots:

+ True eigenvalues ● Computed Ritz values

¦ Bounds on range of computed Ritz values

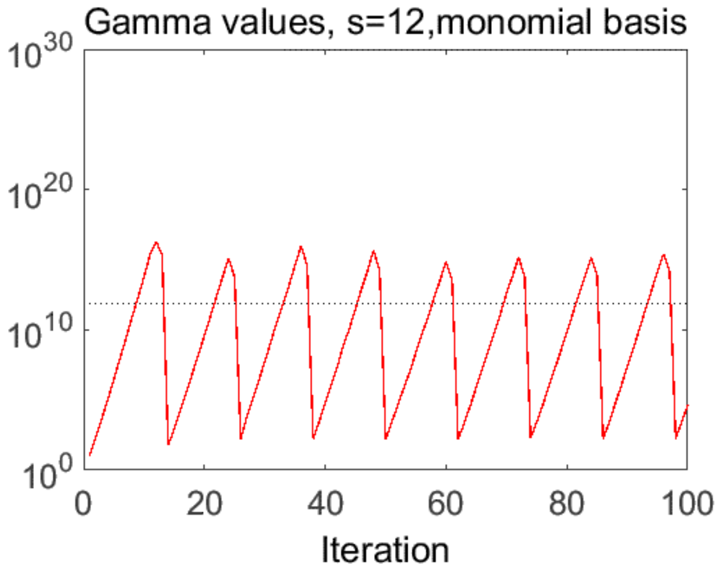
Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector

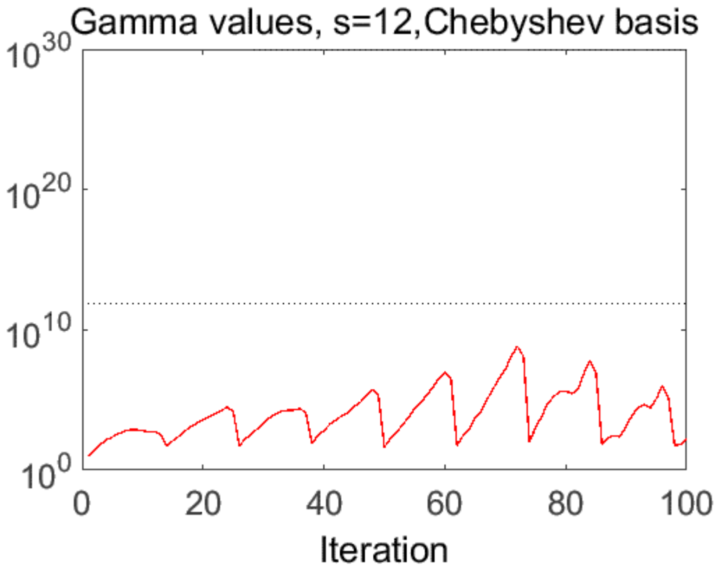
$s = 12$

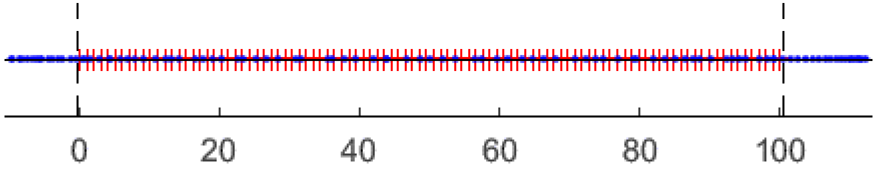
Top plots:

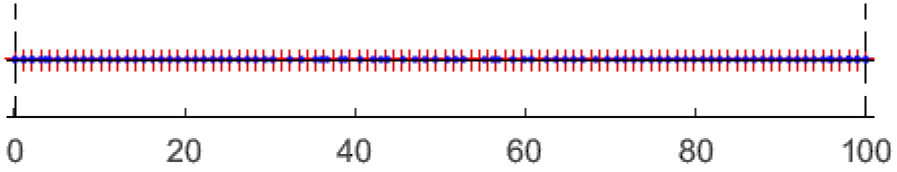
— Computed Γ^2

┄ $(24(\epsilon(n + 11s + 15))^{-1}$









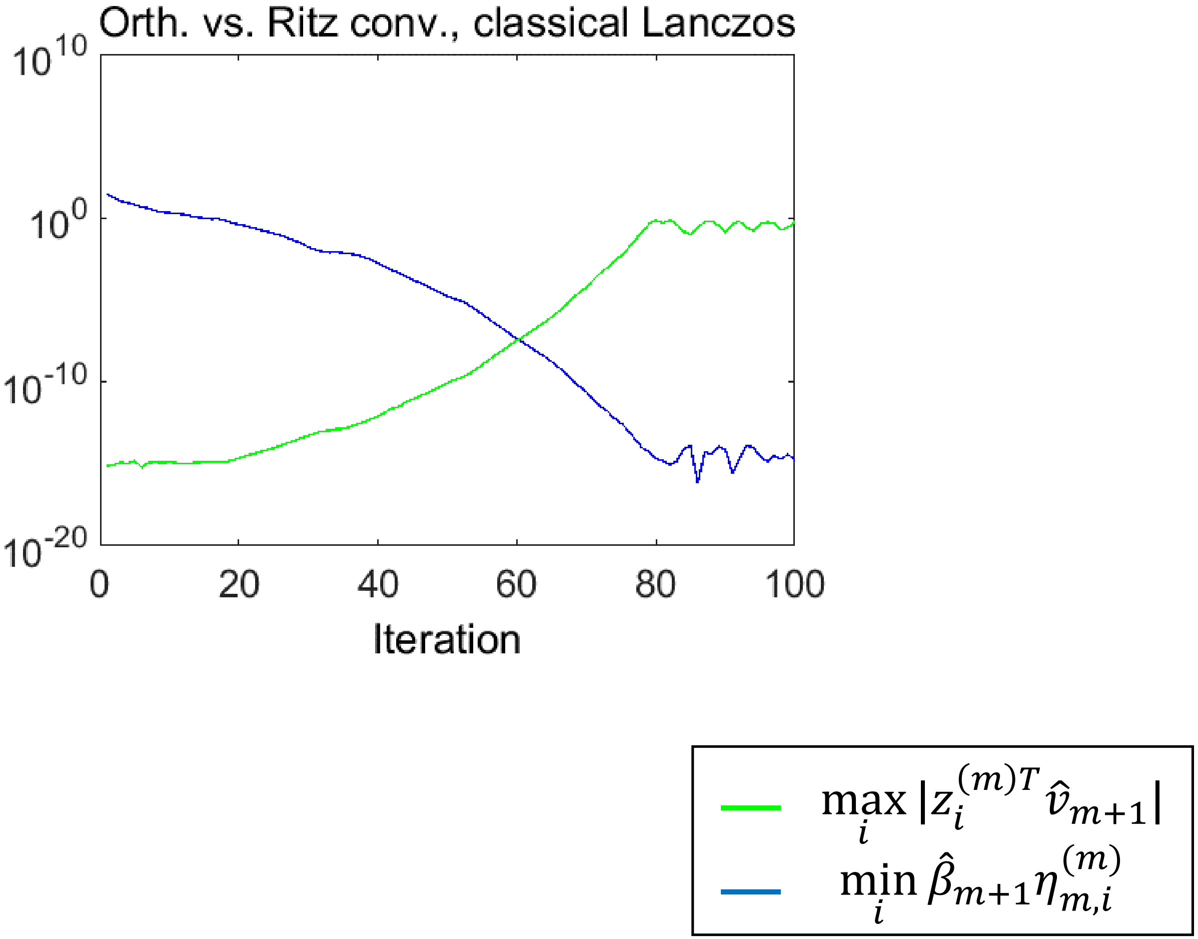
Bottom Plots:

+ True eigenvalues

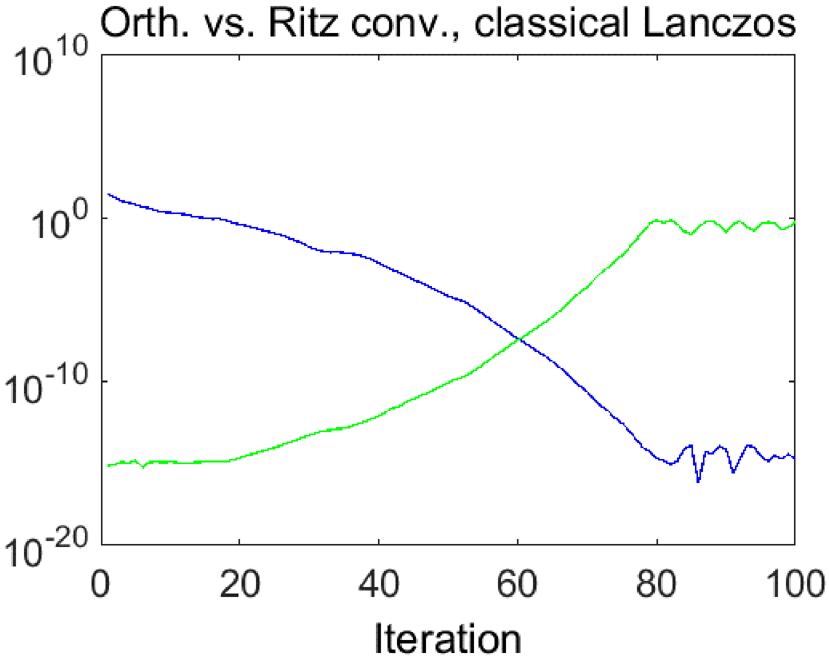
● Computed Ritz values

¦ Bounds on range of computed Ritz values

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



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

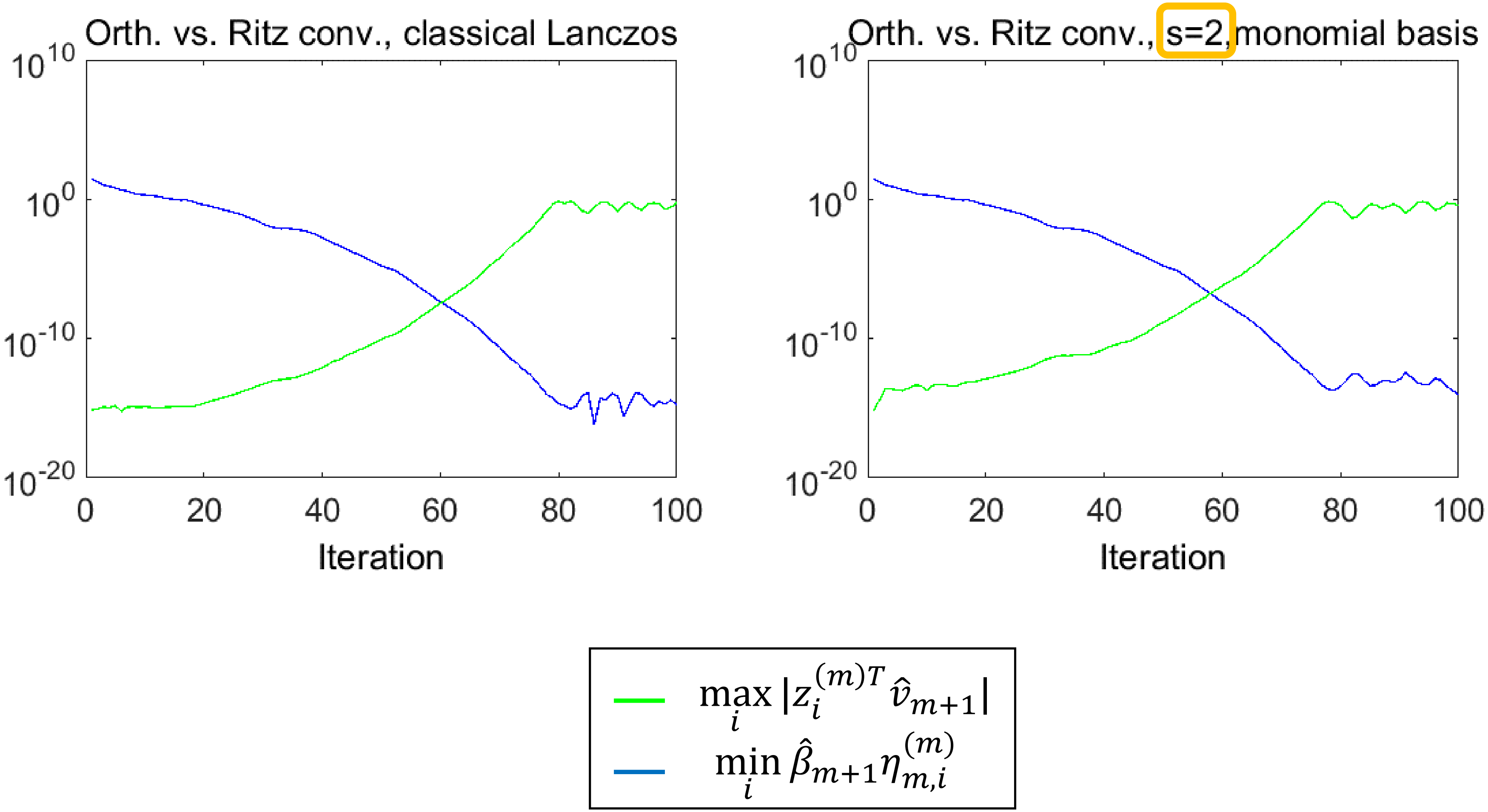


🟩 $\max_i |z_i^{(m)T} \hat{v}_{m+1}|$ ← Measure of loss of orthogonality

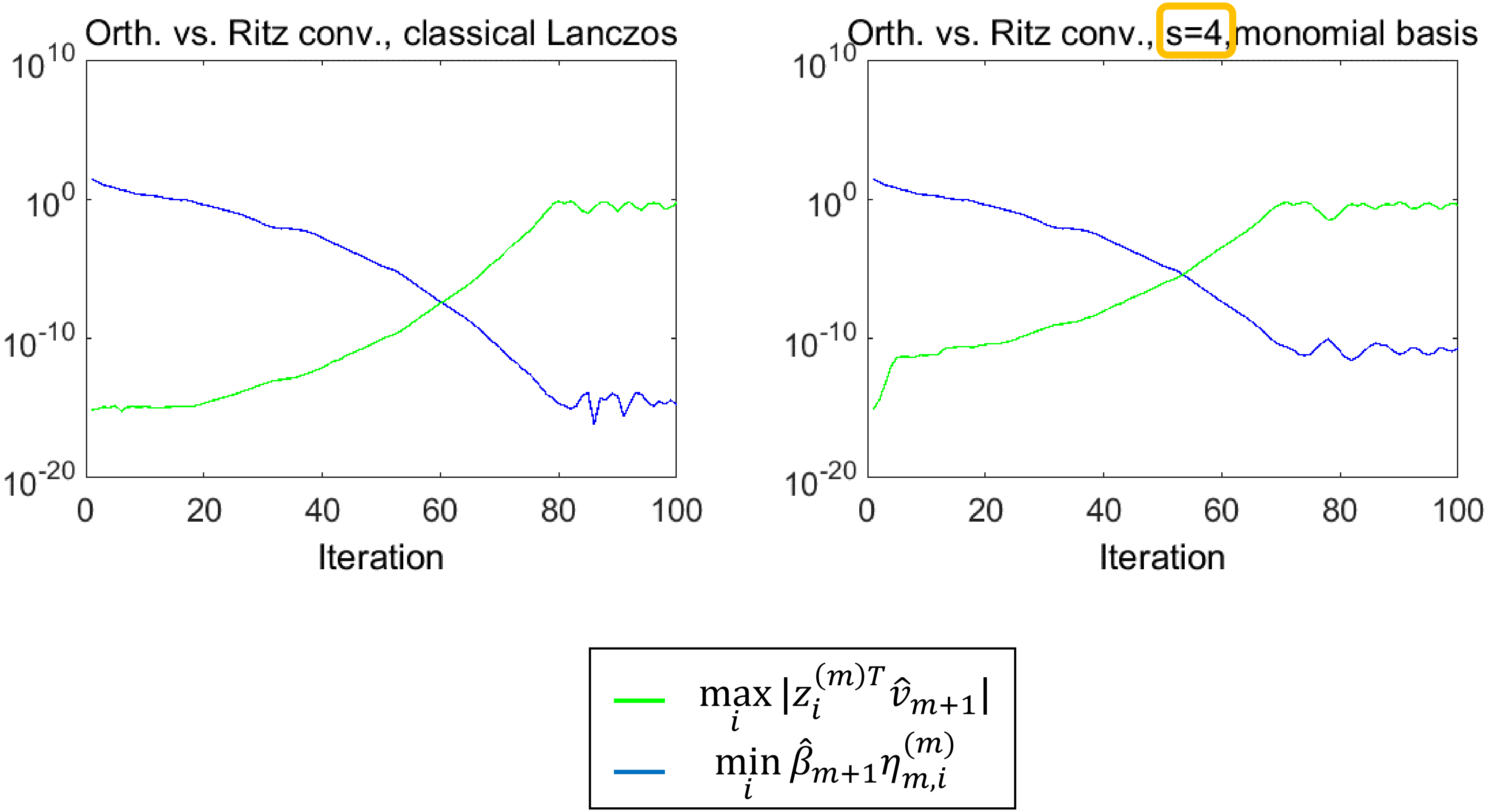
🟦 $\min_i \hat{\beta}_{m+1} \eta_{m,i}^{(m)}$

Measure of Ritz value convergence →

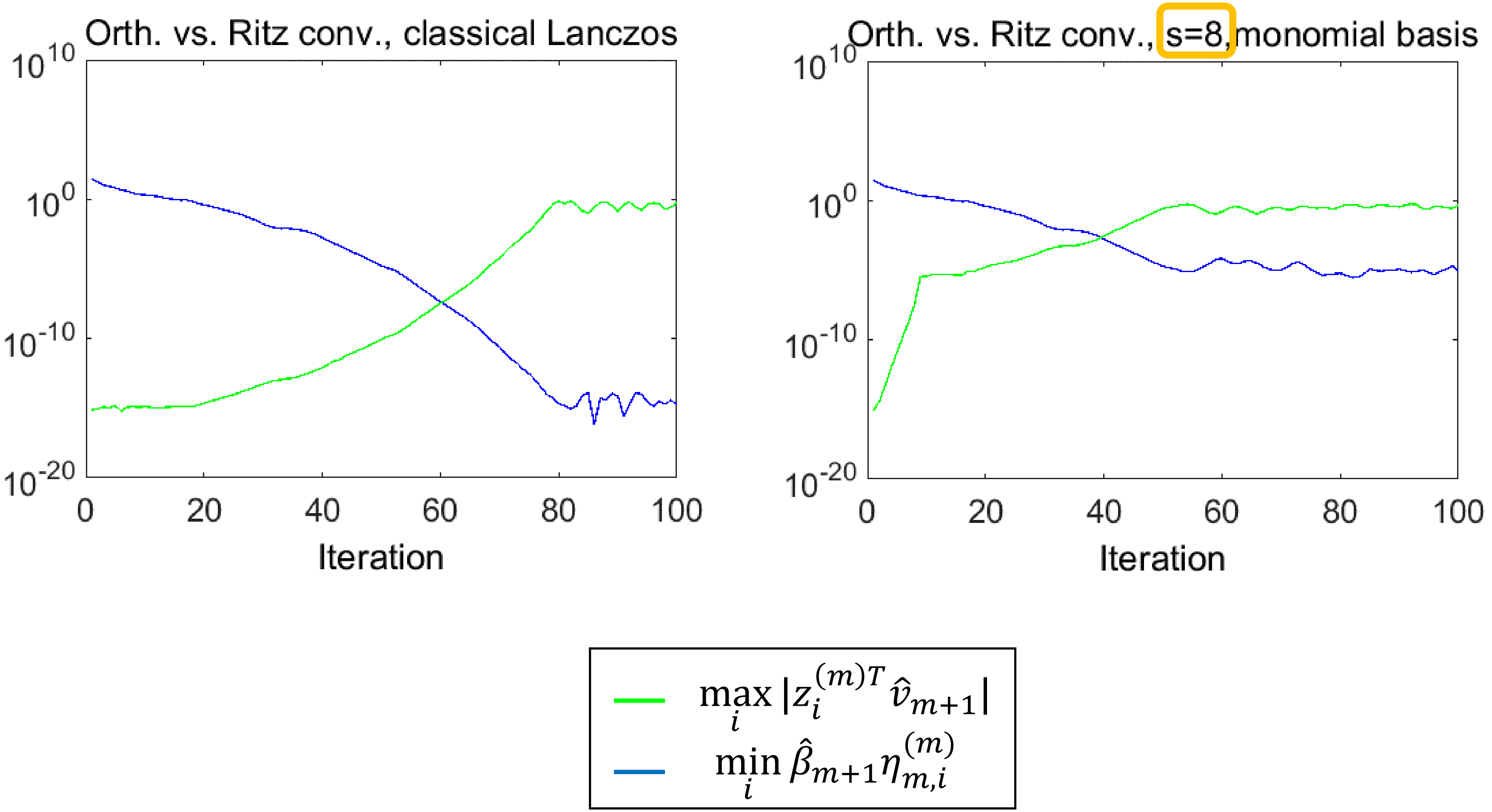
Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector



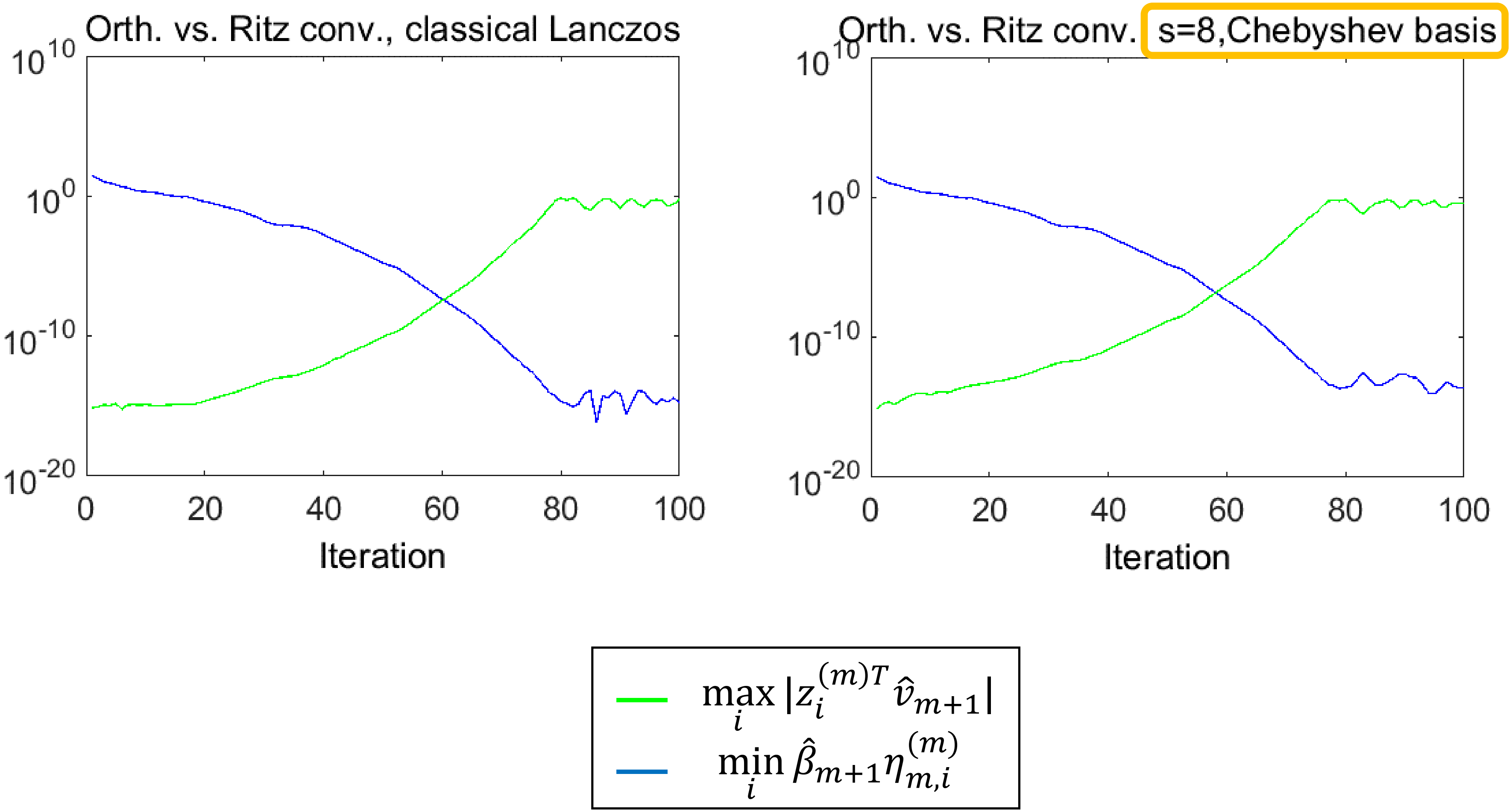
Problem: Diagonal matrix with $n = 100$ with evenly spaced eigenvalues between $\lambda_{min} = 0.1$ and $\lambda_{max} = 100$; random starting vector



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



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

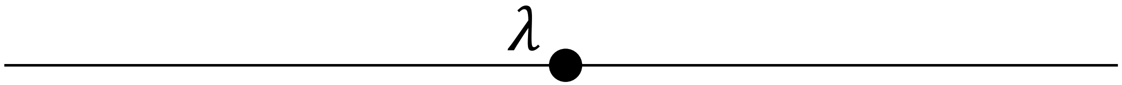


Extending the results of Greenbaum (1989):

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

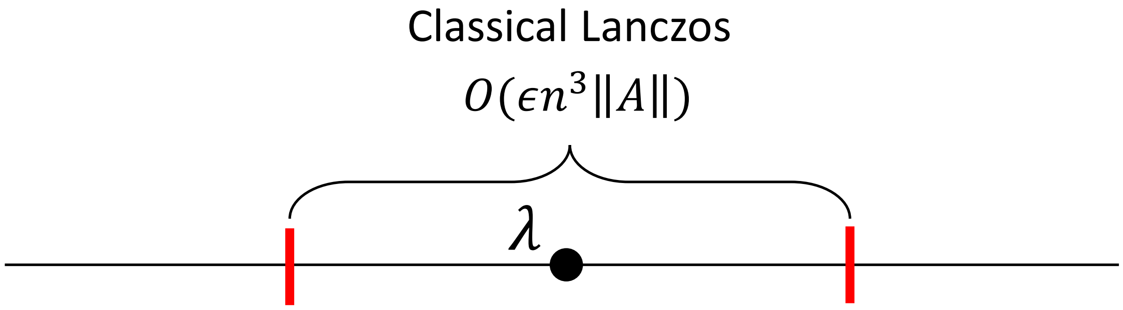
Extending the results of Greenbaum (1989):

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



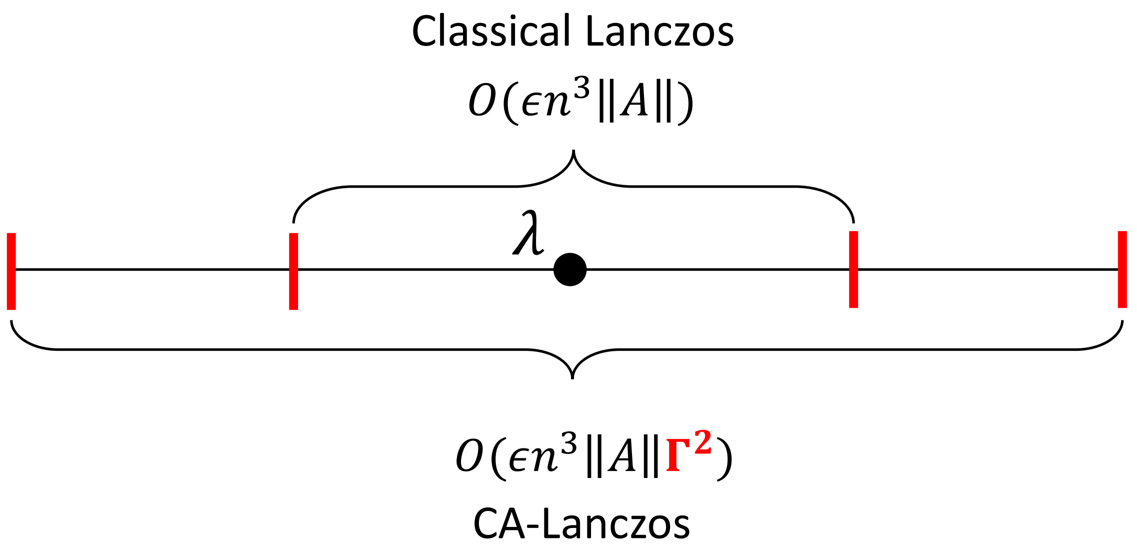
Extending the results of Greenbaum (1989):

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



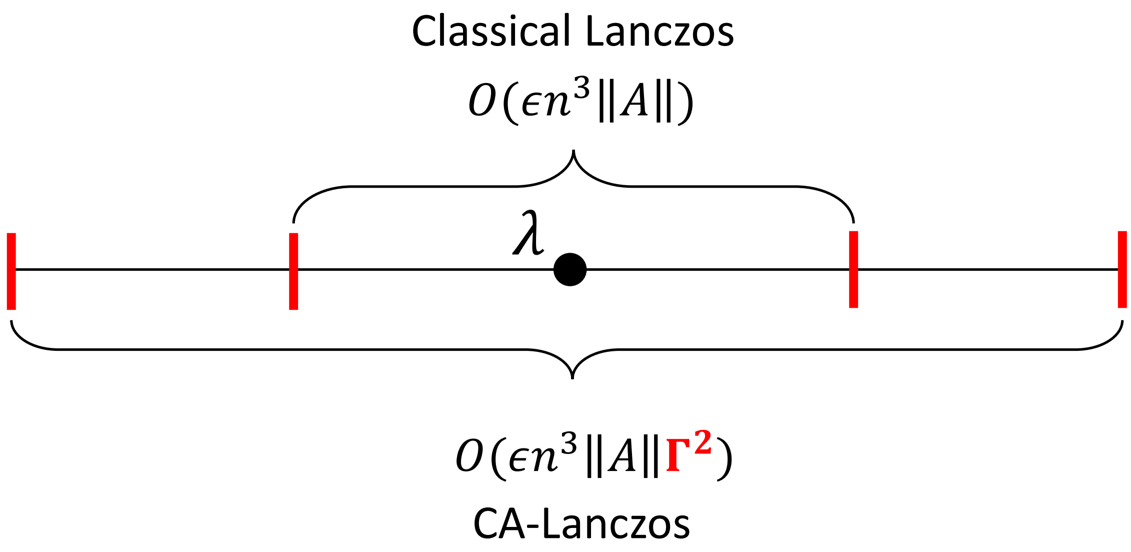
Extending the results of Greenbaum (1989):

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



Ongoing work…

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”

Thank you!

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