Sparse Matrix Computations in the Exascale Era

Erin C. Carson

Seminar of Numerical Mathematics

Katedra numerické matematiky, Matematicko-fyzikální fakulta, Univerzita Karlova

November 15, 2018

This research was partially supported by OP RDE project No. CZ.02.2.69/0.0/0.0/16_027/0008495

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

Exascale Computing: The Modern Space Race

- "Exascale": 10^{18} floating point operations per second

Exascale Computing: The Modern Space Race

- "Exascale": 10^{18} floating point operations per second
- Will enable new frontiers in science and engineering
  - Environment and climate
  - Material, manufacturing, design
  - Healthcare, biology, biomedicine
  - Cosmology and astrophysics
  - High-energy physics

> *Nothing tends so much to the advancement of knowledge as the application of a new instrument.*
>
> \- Sir Humphry Davy

- Advancing knowledge, addressing social challenges, improving quality of life, influencing policy, economic competitiveness

Exascale Computing: The Modern Space Race

- "Exascale": 10^{18} floating point operations per second
- Will enable new frontiers in science and engineering
  - Environment and climate
  - Material, manufacturing, design
  - Healthcare, biology, biomedicine
  - Cosmology and astrophysics
  - High-energy physics

> *Nothing tends so much to the advancement of knowledge as the application of a new instrument.*
>
> \- Sir Humphry Davy

- Advancing knowledge, addressing social challenges, improving quality of life, influencing policy, economic competitiveness

- Much research investment toward achieving exascale within 5-10 years

  ⇨ EuroHPC declaration (2017): €1 billion investment in building exascale infrastructure by 2023

Exascale Computing: The Modern Space Race

- "Exascale": 10^{18} floating point operations per second
- Will enable new frontiers in science and engineering
  - Environment and climate
  - Material, manufacturing, design
  - Healthcare, biology, biomedicine
  - Cosmology and astrophysics
  - High-energy physics

> *Nothing tends so much to the advancement of knowledge as the application of a new instrument.*
>
> \- Sir Humphry Davy

- Advancing knowledge, addressing social challenges, improving quality of life, influencing policy, economic competitiveness

- Much research investment toward achieving exascale within 5-10 years

  ⇨ EuroHPC declaration (2017): €1 billion investment in building exascale infrastructure by 2023

- Challenges at all levels

  hardware to methods and algorithms to applications

Exascale Computing: The Modern Space Race

- "Exascale": 10^{18} floating point operations per second
- Will enable new frontiers in science and engineering
  - Environment and climate
  - Material, manufacturing, design
  - Healthcare, biology, biomedicine
  - Cosmology and astrophysics
  - High-energy physics

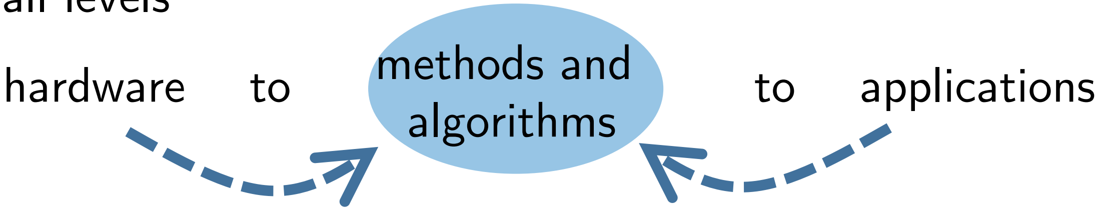
Nothing tends so much to the advancement of knowledge as the application of a new instrument.
- Sir Humphry Davy

- Advancing knowledge, addressing social challenges, improving quality of life, influencing policy, economic competitiveness

- Much research investment toward achieving exascale within 5-10 years

  ⇨ EuroHPC declaration (2017): €1 billion investment in building exascale infrastructure by 2023

- Challenges at all levels

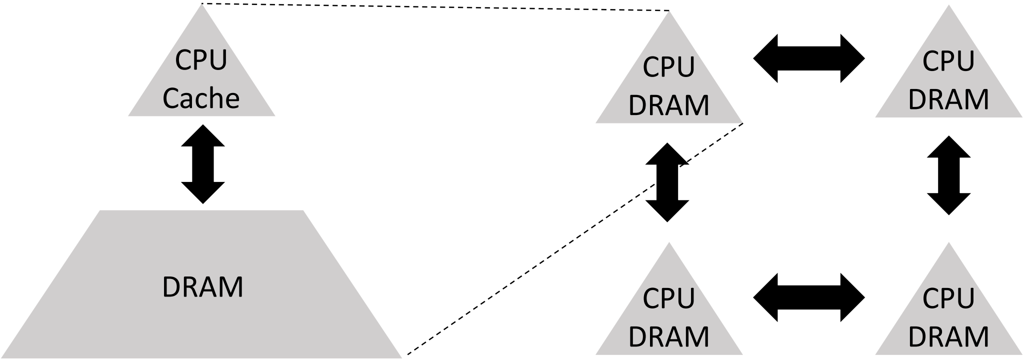


hardware to methods and algorithms to applications

Exascale System Projections

	Today's Systems	Predicted Exascale Systems*
System Peak	10^{16} flops/s	10^{18} flops/s
Node Memory Bandwidth	10^{2} GB/s	10^{3} GB/s
Interconnect Bandwidth	10^{1} GB/s	10^{2} GB/s
Memory Latency	10^{-7} s	$5 \cdot 10^{-8}$ s
Interconnect Latency	10^{-6} s	$5 \cdot 10^{-7}$ s

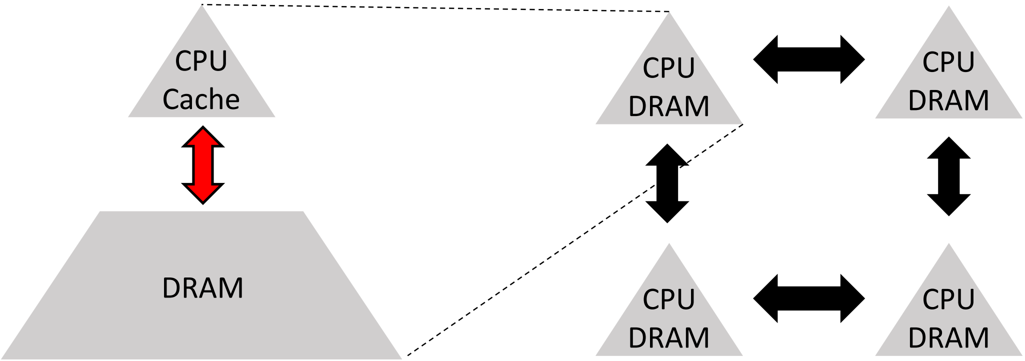
*Sources: from P. Beckman (ANL), J. Shalf (LBL), and D. Unat (LBL)



Exascale System Projections

	Today's Systems	Predicted Exascale Systems*
System Peak	10^{16} flops/s	10^{18} flops/s
Node Memory Bandwidth	10^{2} GB/s	10^{3} GB/s
Interconnect Bandwidth	10^{1} GB/s	10^{2} GB/s
Memory Latency	10^{-7} s	$5 \cdot 10^{-8}$ s
Interconnect Latency	10^{-6} s	$5 \cdot 10^{-7}$ s

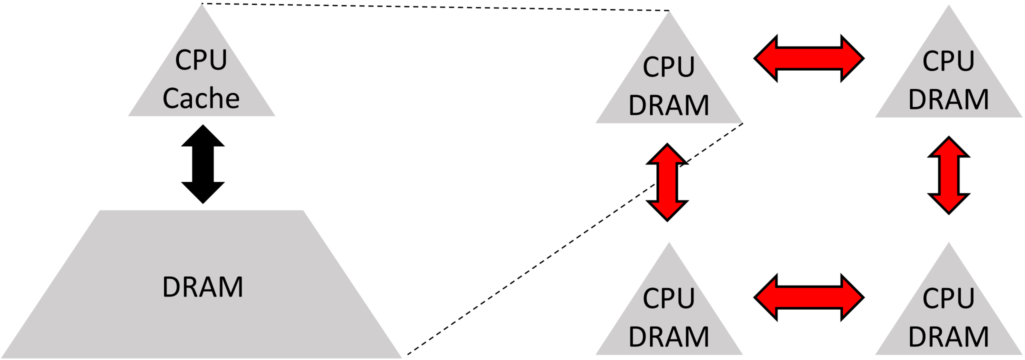
*Sources: from P. Beckman (ANL), J. Shalf (LBL), and D. Unat (LBL)



Exascale System Projections

	Today's Systems	Predicted Exascale Systems*
System Peak	10^{16} flops/s	10^{18} flops/s
Node Memory Bandwidth	10^{2} GB/s	10^{3} GB/s
Interconnect Bandwidth	10^{1} GB/s	10^{2} GB/s
Memory Latency	10^{-7} s	$5 \cdot 10^{-8}$ s
Interconnect Latency	10^{-6} s	$5 \cdot 10^{-7}$ s

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



Exascale System Projections

	Today's Systems	Predicted Exascale Systems*	Factor Improvement
System Peak	10^{16} flops/s	10^{18} flops/s	100
Node Memory Bandwidth	10^{2} GB/s	10^{3} GB/s	10
Interconnect Bandwidth	10^{1} GB/s	10^{2} GB/s	10
Memory Latency	10^{-7} s	$5 \cdot 10^{-8}$ s	2
Interconnect Latency	10^{-6} s	$5 \cdot 10^{-7}$ s	2

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

Exascale System Projections

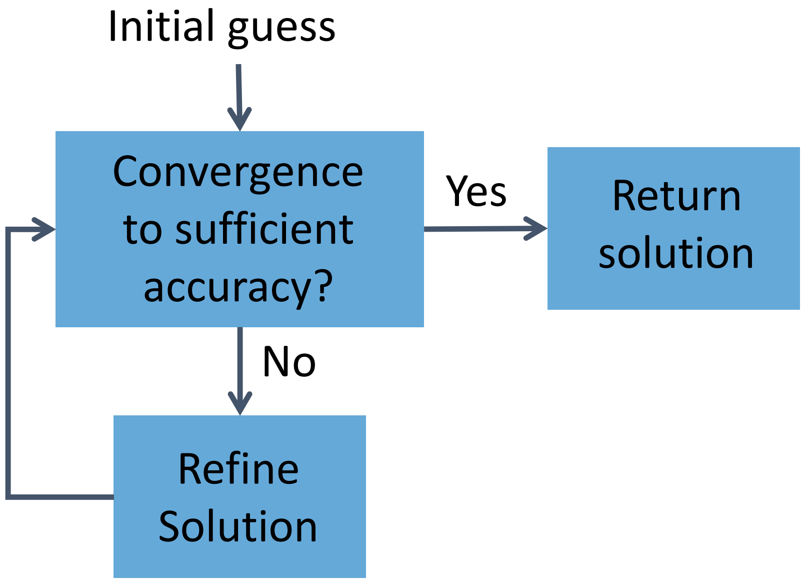
	Today's Systems	Predicted Exascale Systems*	Factor Improvement
System Peak	10^{16} flops/s	10^{18} flops/s	100
Node Memory Bandwidth	10^{2} GB/s	10^{3} GB/s	10
Interconnect Bandwidth	10^{1} GB/s	10^{2} GB/s	10
Memory Latency	10^{-7} s	$5 \cdot 10^{-8}$ s	2
Interconnect Latency	10^{-6} s	$5 \cdot 10^{-7}$ s	2

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

- Movement of data (communication) is much more expensive than floating point operations (computation), in terms of both **time** and **energy**
- Gaps will only grow larger
- Reducing time spent moving data/waiting for data will be essential for applications at exascale!

Iterative Solvers

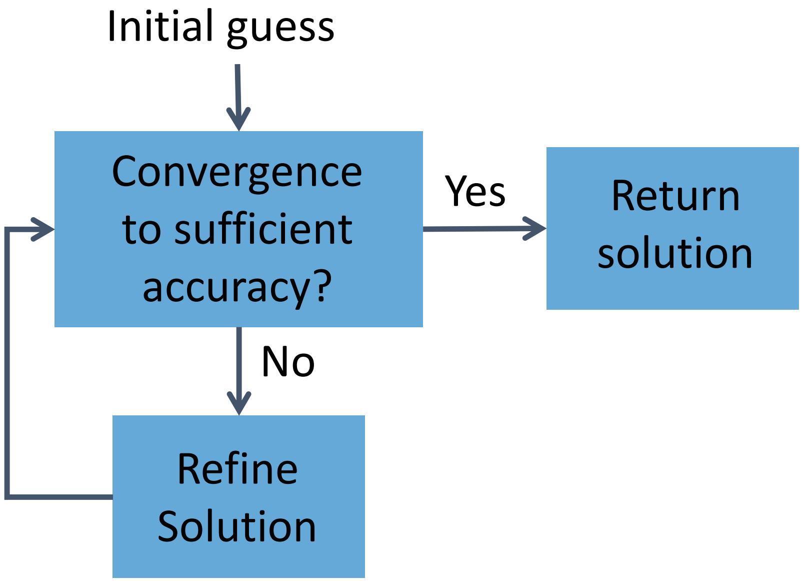
- Focus: Iterative solvers for sparse
  - Linear systems $Ax = b$ and
  - Eigenvalue problems $Ax = \lambda x$



Iterative Solvers

- Focus: Iterative solvers for sparse
  - Linear systems $Ax = b$ and
  - Eigenvalue problems $Ax = \lambda x$

- Iterative solvers used when
  - A is very large, very sparse
  - A is represented implicitly
  - Only approximate answer required
  - Solving nonlinear equations



Krylov Subspace Methods

Krylov Subspace Method: projection process onto the Krylov subspace

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

where A is an $N \times N$ matrix and r_0 is a length-N vector

Krylov Subspace Methods

Krylov Subspace Method: projection process onto the Krylov subspace

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

where A is an $N \times N$ matrix and r_0 is a length-N vector

In each iteration:

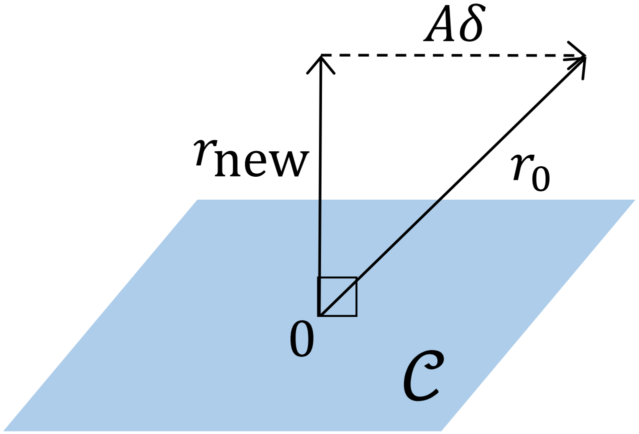
- Add a dimension to the Krylov subspace
  - Forms nested sequence of Krylov subspaces

$$\mathcal{K}_1(A, r_0) \subset \mathcal{K}_2(A, r_0) \subset \cdots \subset \mathcal{K}_i(A, r_0)$$

- Orthogonalize (with respect to some $\mathcal{C}_i$)
- Linear systems: Select approximate solution

$$x_i \in x_0 + \mathcal{K}_i(A, r_0)$$

using $r_i = b - Ax_i \perp \mathcal{C}_i$



Krylov Subspace Methods

Krylov Subspace Method: projection process onto the Krylov subspace

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

where A is an $N \times N$ matrix and r_0 is a length-N vector

In each iteration:

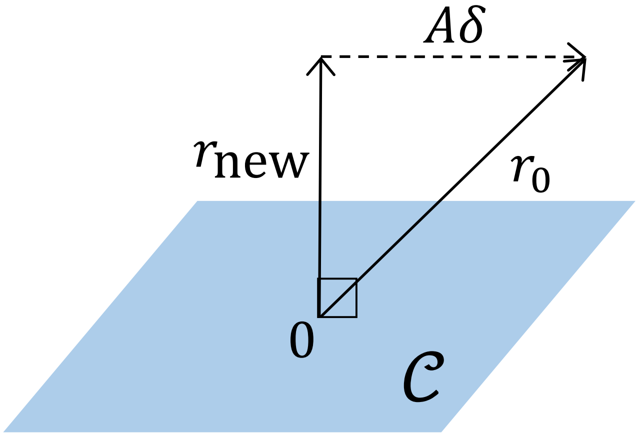
- Add a dimension to the Krylov subspace
  - Forms nested sequence of Krylov subspaces

$$\mathcal{K}_1(A, r_0) \subset \mathcal{K}_2(A, r_0) \subset \cdots \subset \mathcal{K}_i(A, r_0)$$

- Orthogonalize (with respect to some $\mathcal{C}_i$)
- Linear systems: Select approximate solution

$$x_i \in x_0 + \mathcal{K}_i(A, r_0)$$

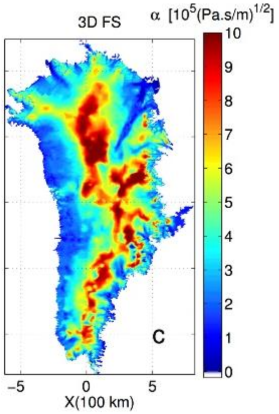
using $r_i = b - Ax_i \perp \mathcal{C}_i$



Conjugate gradient method: A is symmetric positive definite, $\mathcal{C}_i = \mathcal{K}_i(A, r_0)$

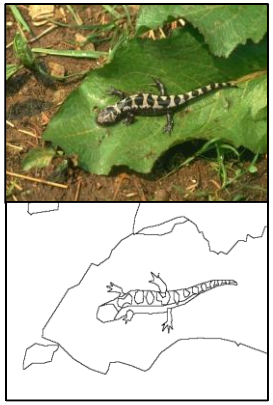
$$r_i \perp \mathcal{K}_i(A, r_0) \quad \Leftrightarrow \quad \|x - x_i\|_A = \min_{z \in x_0 + \mathcal{K}_i(A, r_0)} \|x - z\|_A \quad \Longrightarrow \quad \boldsymbol{r_{N+1} = 0}$$

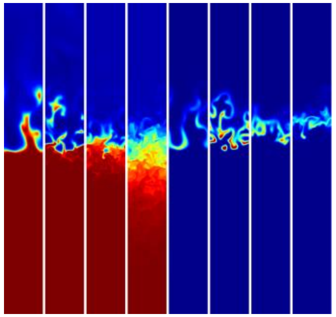
Krylov Subspace Methods in the Wild



Climate Modeling

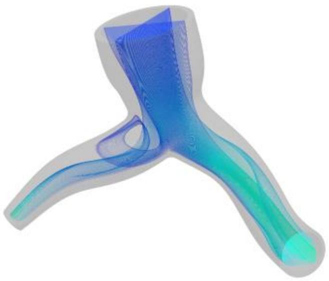
Computer Vision

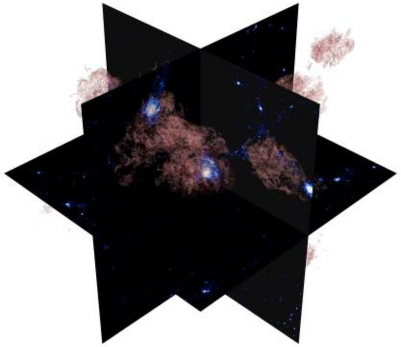




Chemical Engineering

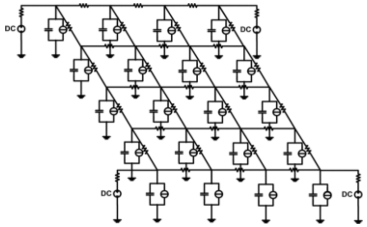
Medical Treatment

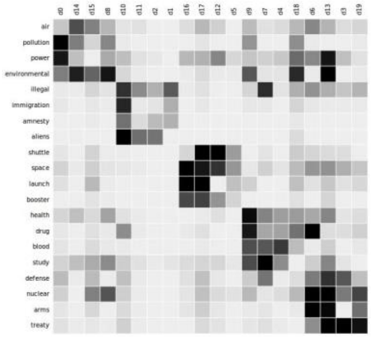




Computational Cosmology

Power Grid Modeling





Latent Semantic Analysis

Financial Portfolio Optimization

Conjugate Gradient on the World's Fastest Computer

Summit - IBM Power System AC922

Site:	Oak Ridge National Laboratory
Manufacturer:	IBM
Cores:	2,282,544
Memory:	2,801,664 GB
Processor:	IBM POWER9 22C 3.07GHz
Interconnect:	Dual-rail Mellanox EDR Infiniband
Performance	
Theoretical peak:	187,659 TFlops/s
LINPACK benchmark:	122,300 Tflops/s
HPCG benchmark:	2,926 Tflops/s

Conjugate Gradient on the World's Fastest Computer

Summit - IBM Power System AC922

→ current #1 on top500

Site:	Oak Ridge National Laboratory
Manufacturer:	IBM
Cores:	2,282,544
Memory:	2,801,664 GB
Processor:	IBM POWER9 22C 3.07GHz
Interconnect:	Dual-rail Mellanox EDR Infiniband
Performance	
Theoretical peak:	187,659 TFlops/s
LINPACK benchmark:	122,300 Tflops/s
HPCG benchmark:	2,926 Tflops/s

Conjugate Gradient on the World's Fastest Computer

Summit - IBM Power System AC922

current #1
on top500

Site:	Oak Ridge National Laboratory
Manufacturer:	IBM
Cores:	2,282,544
Memory:	2,801,664 GB
Processor:	IBM POWER9 22C 3.07GHz
Interconnect:	Dual-rail Mellanox EDR Infiniband
Performance	
Theoretical peak:	187,659 TFlops/s
LINPACK benchmark:	122,300 Tflops/s
HPCG benchmark:	2,926 Tflops/s

LINPACK benchmark
(dense $Ax = b$, direct)
65% efficiency

Conjugate Gradient on the World's Fastest Computer

Summit - IBM Power System AC922

current #1
on top500

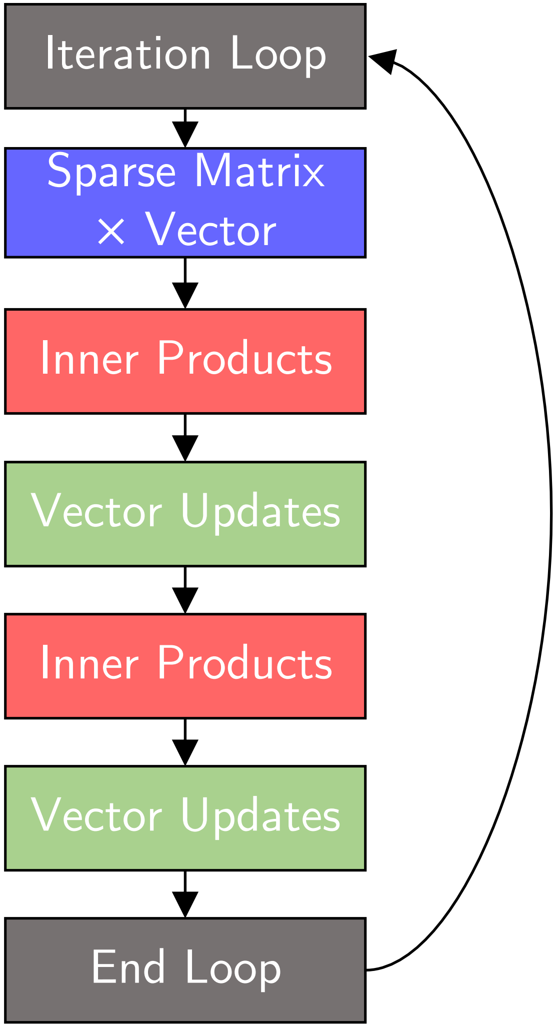
Site:	Oak Ridge National Laboratory
Manufacturer:	IBM
Cores:	2,282,544
Memory:	2,801,664 GB
Processor:	IBM POWER9 22C 3.07GHz
Interconnect:	Dual-rail Mellanox EDR Infiniband
Performance	
Theoretical peak:	187,659 TFlops/s
LINPACK benchmark:	122,300 Tflops/s
HPCG benchmark:	2,926 Tflops/s

LINPACK benchmark
(dense $Ax = b$, direct)
65% efficiency

HPCG benchmark
(sparse $Ax = b$, iterative)
1.5% efficiency

The Conjugate Gradient (CG) Method

```
$r_0 = b - Ax_0, \;\; p_0 = r_0$
for $i$ = 1:nmax
        $\alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T A p_{i-1}}$
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $p_i = r_i + \beta_i p_{i-1}$
end
```



Iteration Loop

Sparse Matrix × Vector

Inner Products

Vector Updates

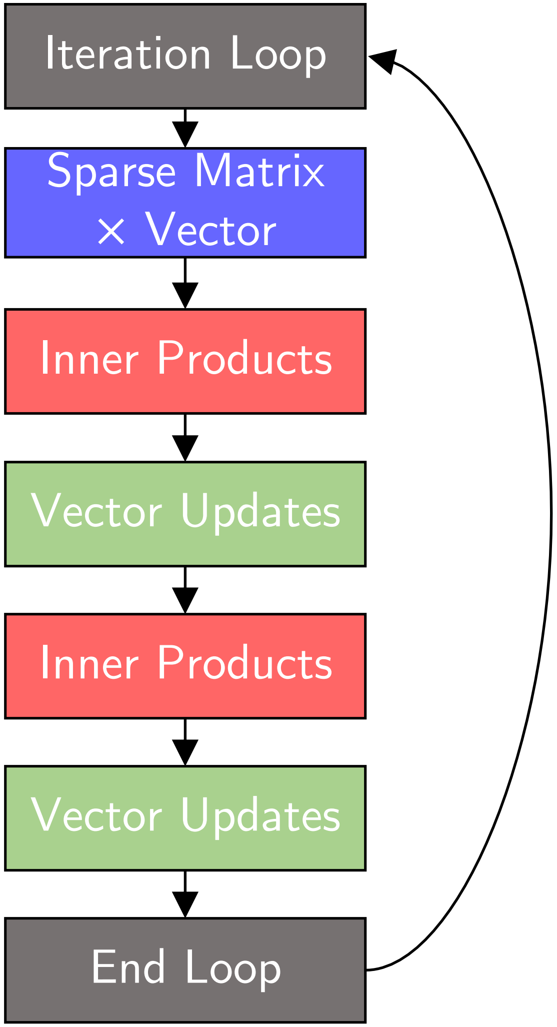
Inner Products

Vector Updates

End Loop

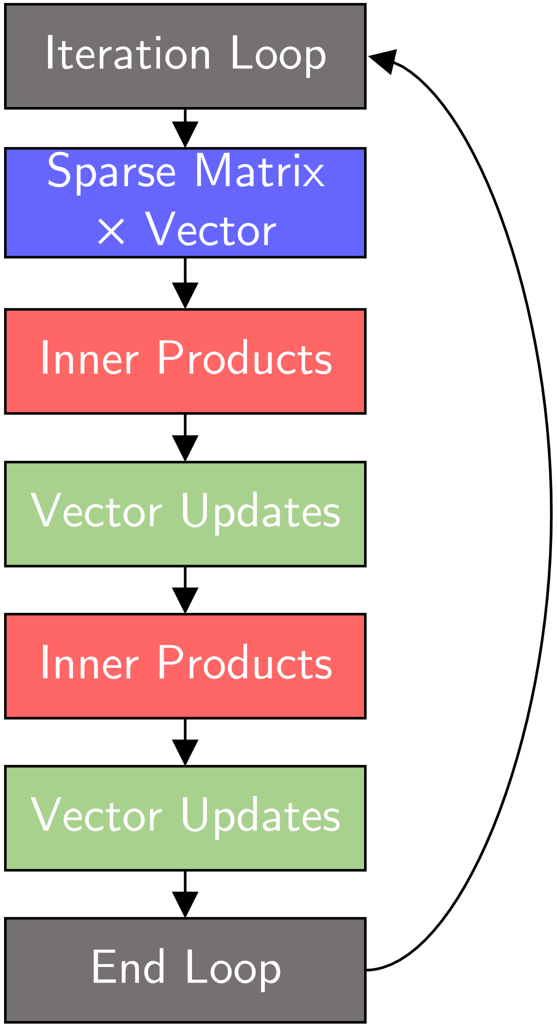
The Conjugate Gradient (CG) Method

```
$r_0 = b - Ax_0, \;\; p_0 = r_0$
for $i$ = 1:nmax
    $\alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T A p_{i-1}}$
    $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
    $r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$
    $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
    $p_i = r_i + \beta_i p_{i-1}$
end
```



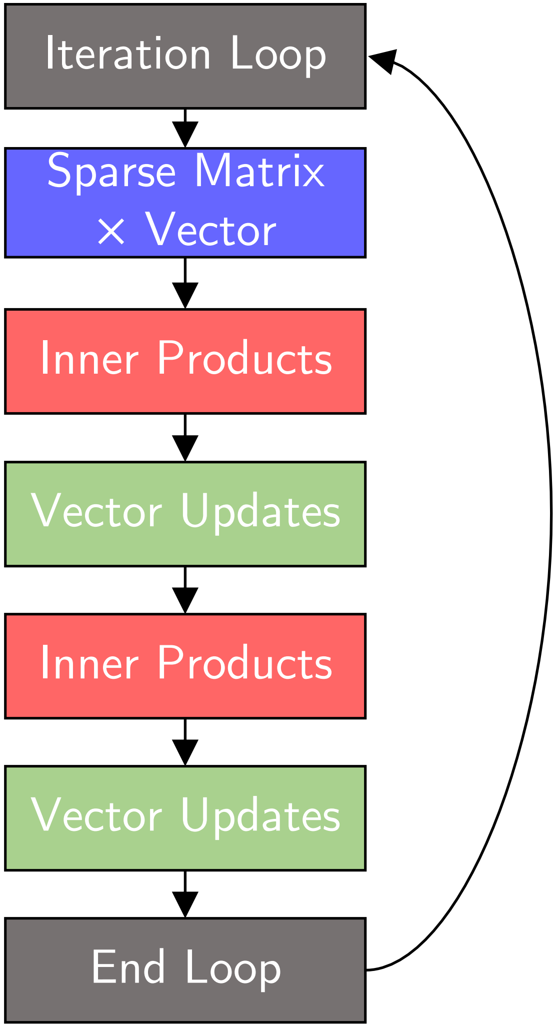
The Conjugate Gradient (CG) Method

```
$r_0 = b - Ax_0, \quad p_0 = r_0$
for $i$ = 1:nmax
        $\alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T A p_{i-1}}$
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $p_i = r_i + \beta_i p_{i-1}$
end
```



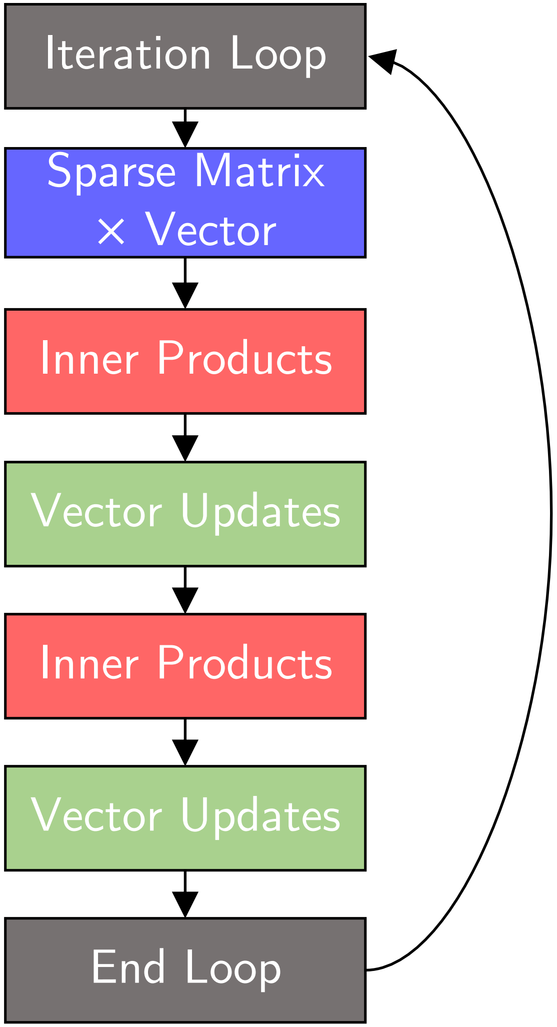
The Conjugate Gradient (CG) Method

```
$r_0 = b - Ax_0,\ \ p_0 = r_0$
for $i$ = 1:nmax
        $\alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T A p_{i-1}}$
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $p_i = r_i + \beta_i p_{i-1}$
end
```



The Conjugate Gradient (CG) Method

```
$r_0 = b - Ax_0, \;\; p_0 = r_0$
for $i$ = 1:nmax
        $\alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T A p_{i-1}}$
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $p_i = r_i + \beta_i p_{i-1}$
end
```



Iteration Loop

Sparse Matrix × Vector

Inner Products

Vector Updates

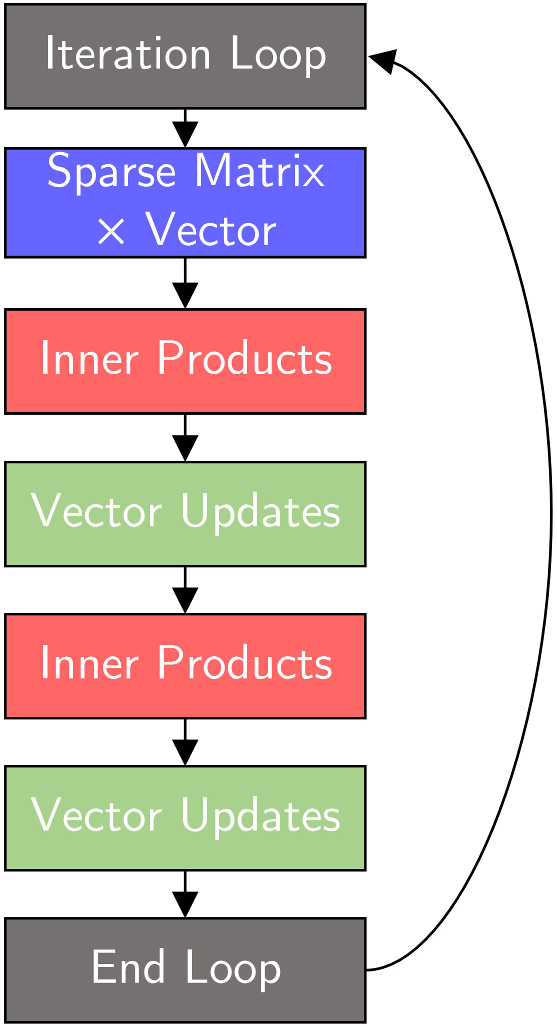
Inner Products

Vector Updates

End Loop

The Conjugate Gradient (CG) Method

```
$r_0 = b - Ax_0, \;\; p_0 = r_0$
for $i$ = 1:nmax
        $\alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T A p_{i-1}}$
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $p_i = r_i + \beta_i p_{i-1}$
end
```



Iteration Loop

Sparse Matrix × Vector

Inner Products

Vector Updates

Inner Products

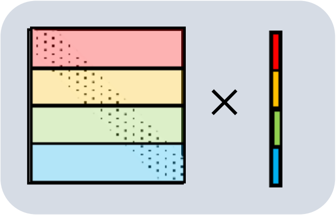
Vector Updates

End Loop

Cost Per Iteration

→ Sparse matrix-vector multiplication (SpMV)

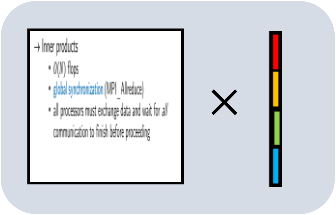
- O(nnz) flops
- Must communicate vector entries w/neighboring processors (nearest neighbor MPI collective)



Cost Per Iteration

→ Sparse matrix-vector multiplication (SpMV)

- O(nnz) flops
- Must communicate vector entries w/neighboring processors (nearest neighbor MPI collective)

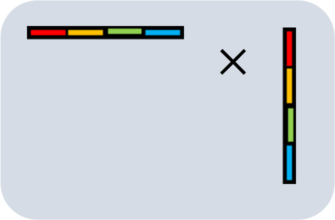


→ Inner products

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

→ Inner products

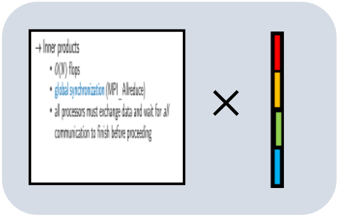
- $O(N)$ flops
- global synchronization (MPI_Allreduce)
- all processors must exchange data and wait for *all* communication to finish before proceeding



Cost Per Iteration

→ Sparse matrix-vector multiplication (SpMV)

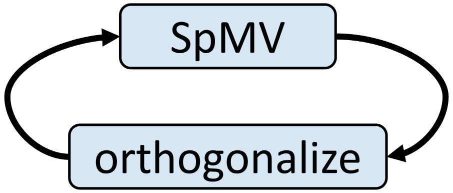
- $O(\text{nnz})$ flops
- Must communicate vector entries w/neighboring processors (nearest neighbor MPI collective)


→ Inner products

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

→ Inner products

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



Low computation/communication ratio

⇒ **Performance is communication-bound**

Synchronization-reducing variants

Communication cost has motivated many approaches to reducing synchronization in CG:

- Pipelined Krylov subspace methods

- s-step Krylov subspace methods

Synchronization-reducing variants

Communication cost has motivated many approaches to reducing synchronization in CG:

- Pipelined Krylov subspace methods
  - Uses modified coefficients and auxiliary vectors to reduce synchronization points to 1 per iteration
  - Modifications also allow decoupling of matrix-vector products and inner products - enables overlapping

- s-step Krylov subspace methods

Synchronization-reducing variants

Communication cost has motivated many approaches to reducing synchronization in CG:

- Pipelined Krylov subspace methods
    - Uses modified coefficients and auxiliary vectors to reduce synchronization points to 1 per iteration
    - Modifications also allow decoupling of matrix-vector products and inner products - enables overlapping

- s-step Krylov subspace methods
    - Compute iterations in blocks of s using a different Krylov subspace basis
    - Enables one synchronization per s iterations

Synchronization-reducing variants

Communication cost has motivated many approaches to reducing synchronization in CG:

- Pipelined Krylov subspace methods
  - Uses modified coefficients and auxiliary vectors to reduce synchronization points to 1 per iteration
  - Modifications also allow decoupling of matrix-vector products and inner products - enables overlapping

Both approaches are *mathematically equivalent* to classical CG

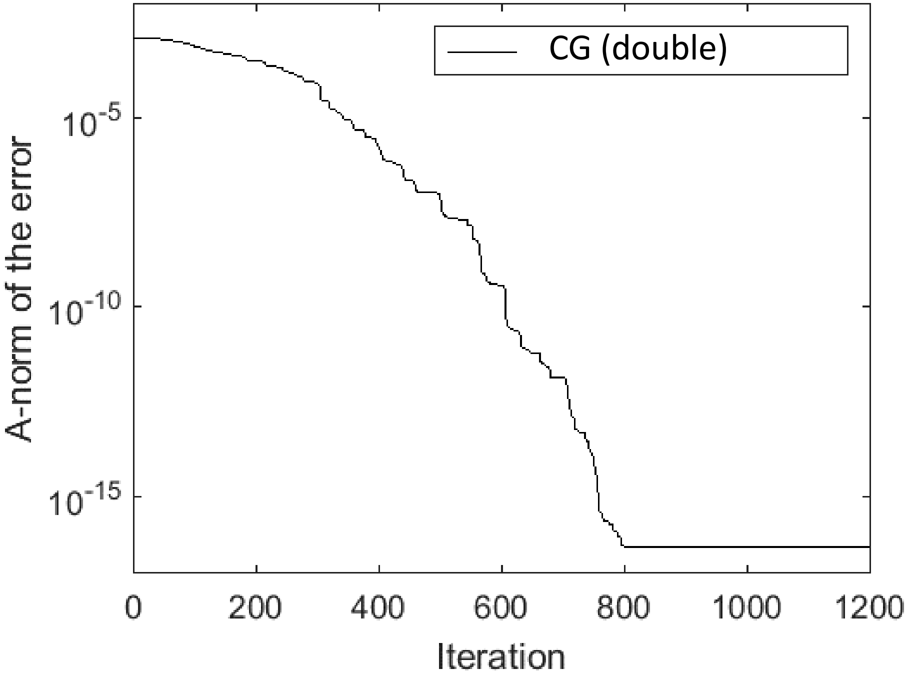
- s-step Krylov subspace methods
  - Compute iterations in blocks of s using a different Krylov subspace basis
  - Enables one synchronization per s iterations

The effects of finite precision

Well-known that roundoff error has two effects:

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

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



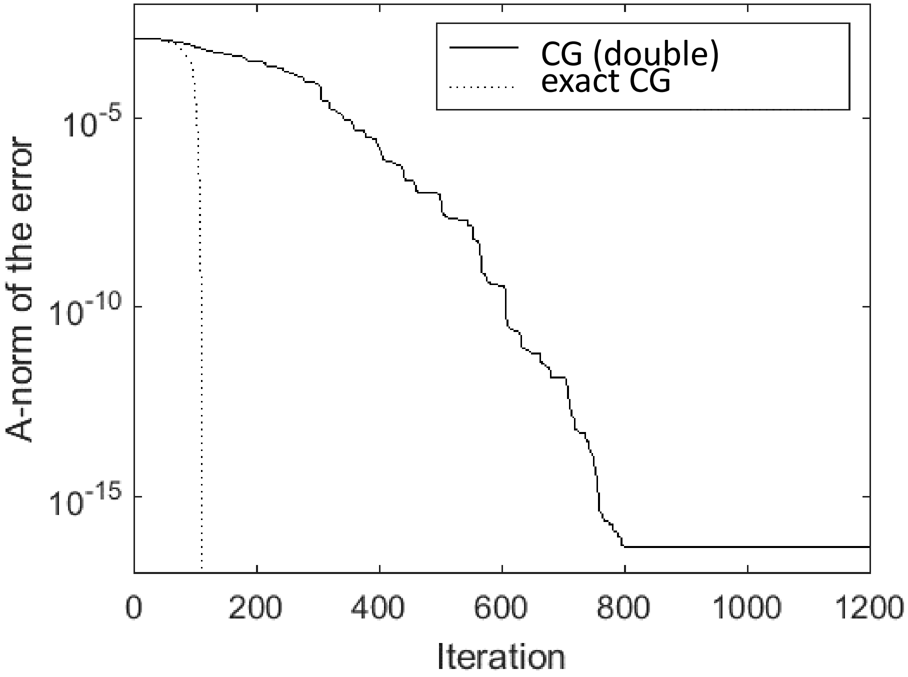
A: bcsstk03 from SuiteSparse,
b: equal components in the eigenbasis of A,$||b|| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

The effects of finite precision

Well-known that roundoff error has two effects:

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

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



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

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

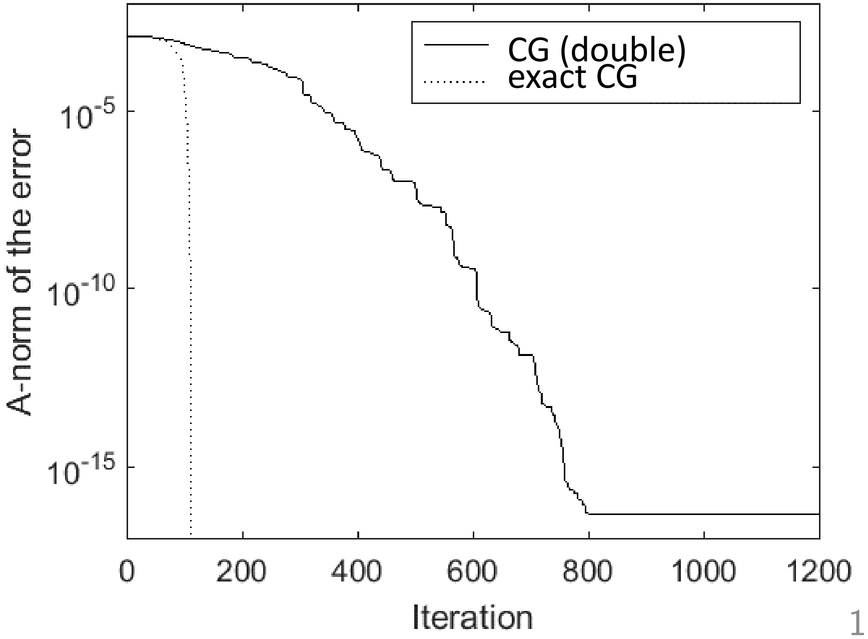
Optimizing high performance iterative solvers

- Synchronization-reducing variants are designed to reduce the time/iteration
- But this is not the whole story!
- What we really want to minimize is the **runtime, subject to some constraint on accuracy,**

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

- Changes to how the recurrences are computed can exacerbate finite precision effects of convergence delay and loss of accuracy

- Crucial that we understand and take into account how algorithm modifications will affect the convergence rate and attainable accuracy!



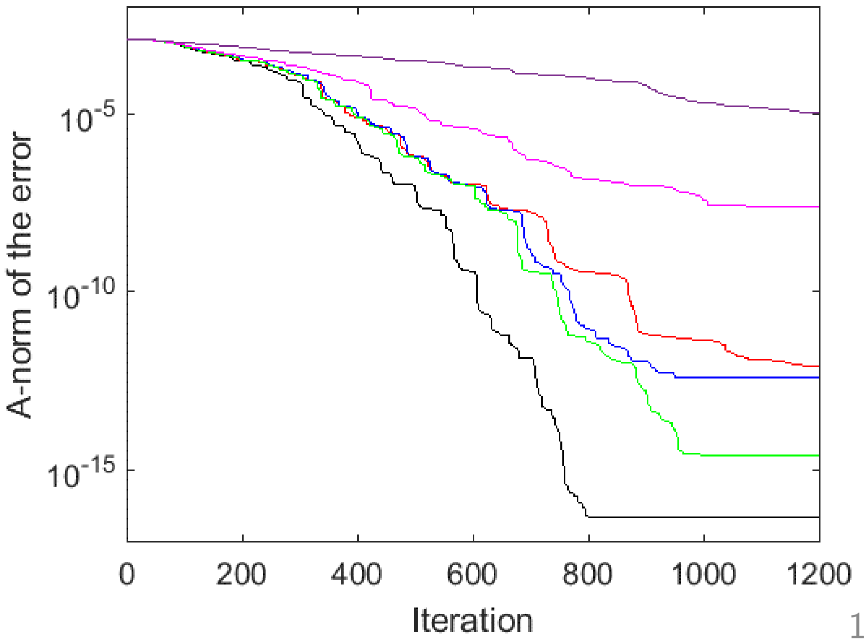
Optimizing high performance iterative solvers

- Synchronization-reducing variants are designed to reduce the time/iteration
- But this is not the whole story!
- What we really want to minimize is the **runtime, subject to some constraint on accuracy,**

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

- Changes to how the recurrences are computed can exacerbate finite precision effects of convergence delay and loss of accuracy

- Crucial that we understand and take into account how algorithm modifications will affect the convergence rate and attainable accuracy!



Maximum attainable accuracy

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

Maximum attainable accuracy

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

Maximum attainable accuracy

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

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

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

Maximum attainable accuracy

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

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

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

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

Maximum attainable accuracy

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

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

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

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

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

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

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

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

Pipelined CG (GVCG)

- Overall idea: use auxiliary recurrences and modified formulas for recurrence coefficients α_i and β_i to reduce/decouple synchronization points

Pipelined CG (GVCG)

- Overall idea: use auxiliary recurrences and modified formulas for recurrence coefficients α_i and β_i to reduce/decouple synchronization points
- Long history of related work:
  - Modified recurrence coefficient computation: Johnson [1983, 1984], van Rosendale [1983, 1984], Saad [1985]
  - CG with two 3-term recurrences (STCG) [Stiefel, 1952/53]; analyzed by Gutknecht and Strakoš [2000]

Pipelined CG (GVCG)

- Overall idea: use auxiliary recurrences and modified formulas for recurrence coefficients α_i and β_i to reduce/decouple synchronization points
- Long history of related work:
  - Modified recurrence coefficient computation: Johnson [1983, 1984], van Rosendale [1983, 1984], Saad [1985]
  - CG with two 3-term recurrences (STCG) [Stiefel, 1952/53]; analyzed by Gutknecht and Strakoš [2000]

Pipelined CG (GVCG)

- Overall idea: use auxiliary recurrences and modified formulas for recurrence coefficients α_i and β_i to reduce/decouple synchronization points
- Long history of related work:
    - Modified recurrence coefficient computation: Johnson [1983, 1984], van Rosendale [1983, 1984], Saad [1985]
    - CG with two 3-term recurrences (STCG) [Stiefel, 1952/53]; analyzed by Gutknecht and Strakoš [2000]

- Approach of Chronopoulos and Gear [1989]
    - Uses auxiliary vector $s_i \equiv Ap_i$ and different computation of α_i to reduce number of synchronizations per iteration from 2 to 1

Pipelined CG (GVCG)

- Overall idea: use auxiliary recurrences and modified formulas for recurrence coefficients α_i and β_i to reduce/decouple synchronization points
- Long history of related work:
  - Modified recurrence coefficient computation: Johnson [1983, 1984], van Rosendale [1983, 1984], Saad [1985]
  - CG with two 3-term recurrences (STCG) [Stiefel, 1952/53]; analyzed by Gutknecht and Strakoš [2000]

- Approach of Chronopoulos and Gear [1989]
  - Uses auxiliary vector $s_i \equiv Ap_i$ and different computation of α_i to reduce number of synchronizations per iteration from 2 to 1

- Pipelined CG of Ghysels and Vanroose [2014]
  - Uses 3 auxiliary vectors: Ap_i, Ar_i and $A^2 r_i$
  - Removes sequential dependency between matrix-vector products and inner products
  - Computations can then be *overlapped* using nonblocking (asynchronous) communication ⇒ hides the latency of global communications

GVCG (Ghysels and Vanroose 2014)

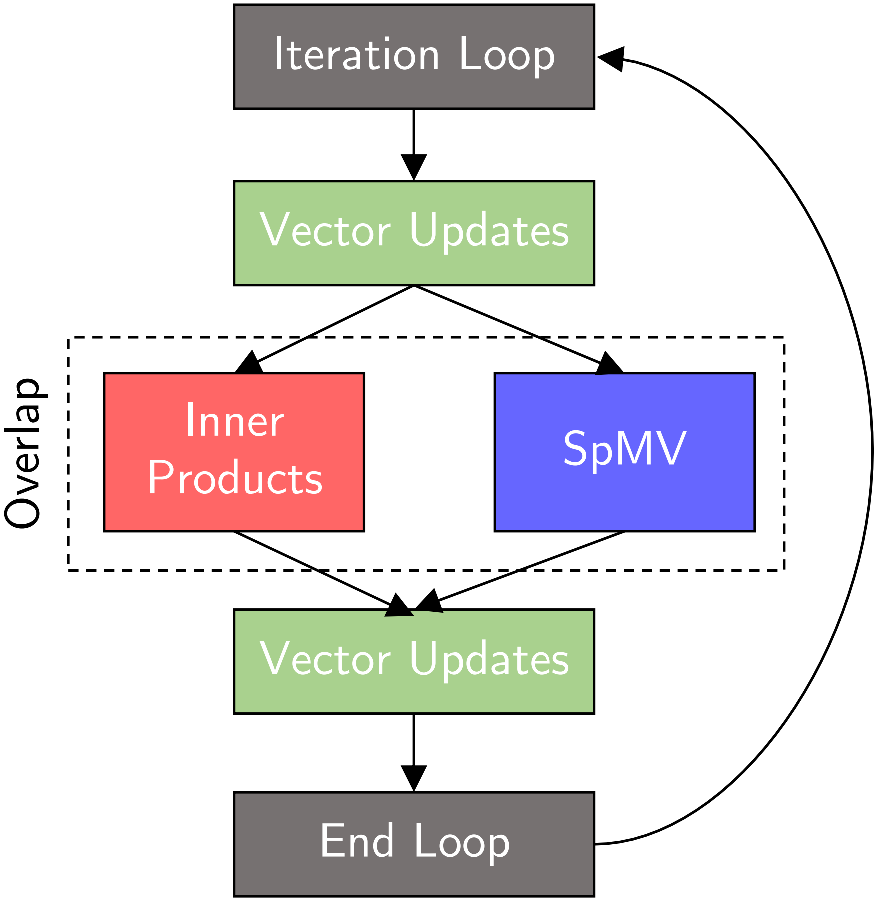
```
$r_0 = b - Ax_0,\ p_0 = r_0$
$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$
for $i$ = 1:nmax
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
        $w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$
        $q_i = Aw_i$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i}$
        $p_i = r_i + \beta_i p_{i-1}$
        $s_i = w_i + \beta_i s_{i-1}$
        $z_i = q_i + \beta_i z_{i-1}$
end
```

GVCG (Ghysels and Vanroose 2014)

```
$r_0 = b - Ax_0,\ p_0 = r_0$
$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$
for $i$ = 1:nmax
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
        $w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$
        $q_i = Aw_i$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i}$
        $p_i = r_i + \beta_i p_{i-1}$
        $s_i = w_i + \beta_i s_{i-1}$
        $z_i = q_i + \beta_i z_{i-1}$
end
```

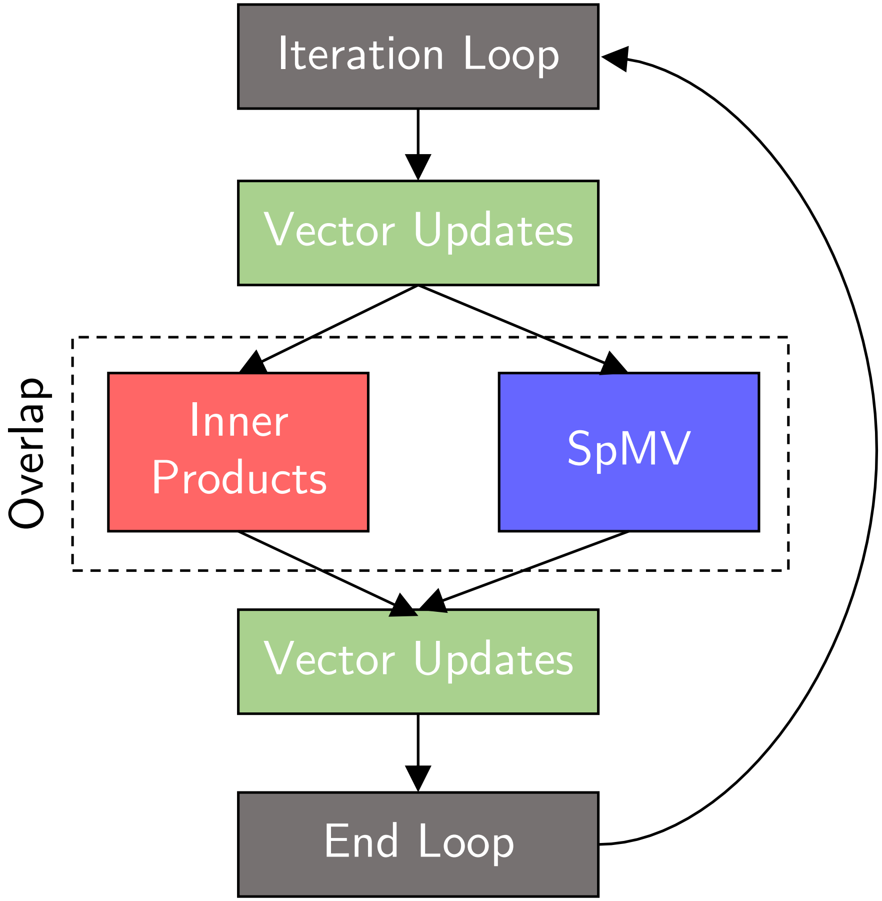
GVCG (Ghysels and Vanroose 2014)

```
$r_0 = b - Ax_0,\ p_0 = r_0$
$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$
for $i = 1$:nmax
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
        $w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$
        $q_i = Aw_i$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i}$
        $p_i = r_i + \beta_i p_{i-1}$
        $s_i = w_i + \beta_i s_{i-1}$
        $z_i = q_i + \beta_i z_{i-1}$
end
```



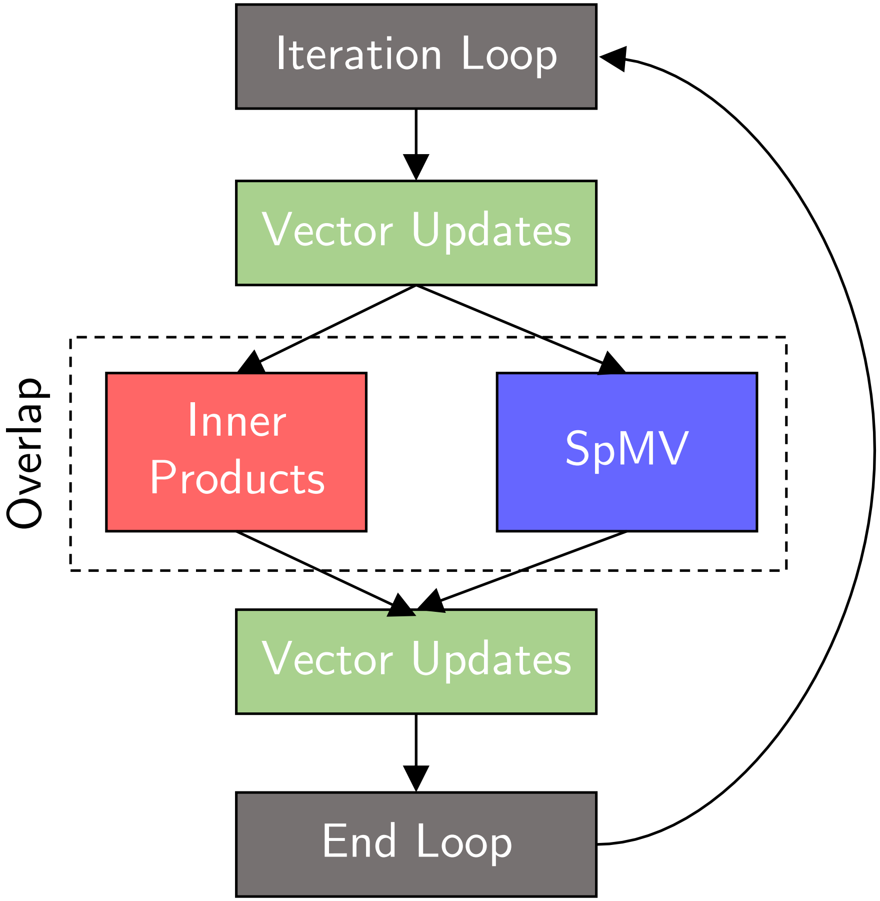
GVCG (Ghysels and Vanroose 2014)

```
$r_0 = b - Ax_0,\ p_0 = r_0$
$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$
for $i$ = 1:nmax
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
        $w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$
        $q_i = Aw_i$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i}$
        $p_i = r_i + \beta_i p_{i-1}$
        $s_i = w_i + \beta_i s_{i-1}$
        $z_i = q_i + \beta_i z_{i-1}$
end
```



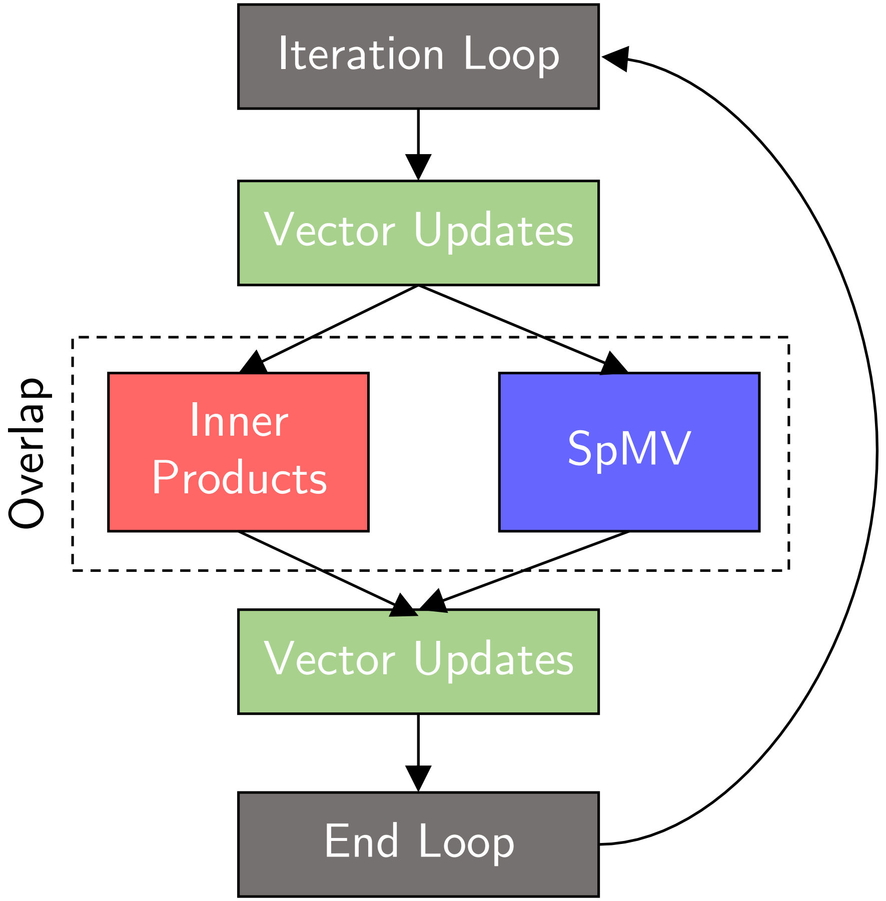
GVCG (Ghysels and Vanroose 2014)

```
$r_0 = b - Ax_0,\ p_0 = r_0$
$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$
for $i$ = 1:nmax
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
        $w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$
        $q_i = Aw_i$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i/\alpha_{i-1}) r_i^T r_i}$
        $p_i = r_i + \beta_i p_{i-1}$
        $s_i = w_i + \beta_i s_{i-1}$
        $z_i = q_i + \beta_i z_{i-1}$
end
```



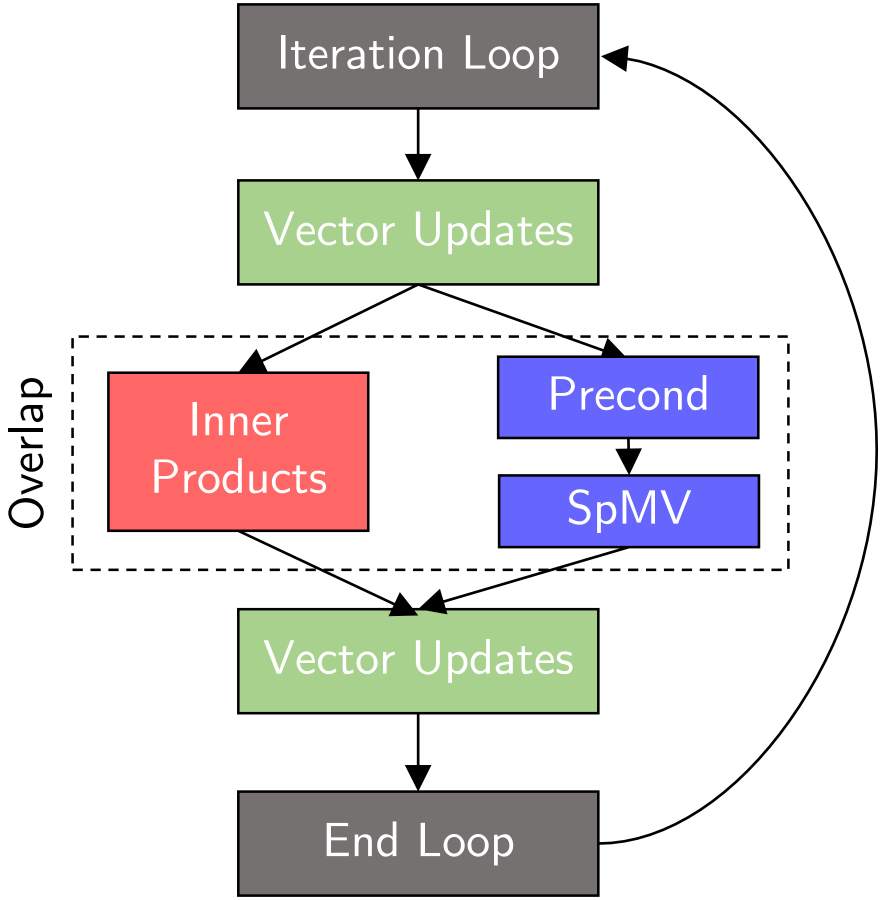
GVCG (Ghysels and Vanroose 2014)

```
$r_0 = b - Ax_0,\ p_0 = r_0$
$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$
for $i = 1$:nmax
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
        $w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$
        $q_i = Aw_i$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i}$
        $p_i = r_i + \beta_i p_{i-1}$
        $s_i = w_i + \beta_i s_{i-1}$
        $z_i = q_i + \beta_i z_{i-1}$
end
```



GVCG (Ghysels and Vanroose 2014)

```
$r_0 = b - Ax_0,\ p_0 = r_0$
$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$
for $i$ = 1:nmax
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
        $w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$
        $q_i = Aw_i$
        $\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$
        $\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i}$
        $p_i = r_i + \beta_i p_{i-1}$
        $s_i = w_i + \beta_i s_{i-1}$
        $z_i = q_i + \beta_i z_{i-1}$
end
```



Iteration Loop

Vector Updates

Overlap

Inner Products

Precond

SpMV

Vector Updates

End Loop

Attainable accuracy of pipelined CG

- What is the effect of adding auxiliary recurrences to the CG method?

Attainable accuracy of pipelined CG

- What is the effect of adding auxiliary recurrences to the CG method?
- To isolate the effects, we consider a simplified version of a pipelined method

```
$r_0 = b - Ax_0, p_0 = r_0, s_0 = Ap_0$
for $i$ = 1:nmax
        $\alpha_{i-1} = \frac{(r_{i-1}, r_{i-1})}{(p_{i-1}, s_{i-1})}$
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
        $\beta_i = \frac{(r_i, r_i)}{(r_{i-1}, r_{i-1})}$
        $p_i = r_i + \beta_i p_{i-1}$
        $s_i = Ar_i + \beta_i s_{i-1}$
end
```

Attainable accuracy of pipelined CG

- What is the effect of adding auxiliary recurrences to the CG method?
- To isolate the effects, we consider a simplified version of a pipelined method
    - Uses same update formulas for α and β as HSCG, but uses additional recurrence for Ap_i

```
$r_0 = b - Ax_0, p_0 = r_0, s_0 = Ap_0$
for $i$ = 1:nmax
        $\alpha_{i-1} = \frac{(r_{i-1}, r_{i-1})}{(p_{i-1}, s_{i-1})}$
        $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
        $r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
        $\beta_i = \frac{(r_i, r_i)}{(r_{i-1}, r_{i-1})}$
        $p_i = r_i + \beta_i p_{i-1}$
        $s_i = Ar_i + \beta_i s_{i-1}$
end
```

see [C., Rozložník, Strakoš, Tíchy, Tůma, 2018]

Attainable accuracy of simple pipelined CG

$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} + \boldsymbol{\delta x_i}$ $\qquad$ $\hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}\,\hat{s}_{i-1} + \boldsymbol{\delta r_i}$

Attainable accuracy of simple pipelined CG

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} + \boldsymbol{\delta x_i} \qquad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}\,\hat{s}_{i-1} + \boldsymbol{\delta r_i}$$

$$f_i = \hat{r}_i - (b - A\hat{x}_i)$$

Attainable accuracy of simple pipelined CG

$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} + \boldsymbol{\delta x_i}$ $\qquad$ $\hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}\,\hat{s}_{i-1} + \boldsymbol{\delta r_i}$

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

Attainable accuracy of simple pipelined CG

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} + \boldsymbol{\delta x_i} \qquad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}\,\hat{s}_{i-1} + \boldsymbol{\delta r_i}$$

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

where

$$G_i = \hat{S}_i - A\hat{P}_i, \quad d_i = [\hat{\alpha}_0, \dots, \hat{\alpha}_{i-1}]^T$$

Attainable accuracy of simple pipelined CG

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} + \boldsymbol{\delta x_i} \qquad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}\,\hat{s}_{i-1} + \boldsymbol{\delta r_i}$$

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

where

$$G_i = \hat{S}_i - A\hat{P}_i, \quad d_i = [\hat{\alpha}_0, \dots, \hat{\alpha}_{i-1}]^T$$

Classical CG: $f_i = f_0 + \sum_{m=1}^{i}(A\delta x_m + \delta r_m)$

Attainable accuracy of simple pipelined CG

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} + \boldsymbol{\delta x_i} \qquad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}\,\hat{s}_{i-1} + \boldsymbol{\delta r_i}$$

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

where

$$G_i = \hat{S}_i - A\hat{P}_i, \quad d_i = [\hat{\alpha}_0, \dots, \hat{\alpha}_{i-1}]^T$$

$$\boxed{\text{Classical CG:} \quad f_i = f_0 + \textstyle\sum_{m=1}^{i}(A\delta x_m + \delta r_m)}$$

Attainable accuracy of simple pipelined CG

$$\|G_i\| \le \frac{O(\varepsilon)}{1-O(\varepsilon)}\left(\kappa(\widehat{U}_i)\|A\|\left\|\widehat{P}_i\right\| + \|A\|\left\|\widehat{R}_i\right\|\left\|\widehat{U}_i^{-1}\right\|\right)$$

$$\widehat{U}_i = \begin{bmatrix} 1 & -\hat{\beta}_1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & 1 & -\hat{\beta}_{i-1} \\ 0 & \dots & 0 & 1 \end{bmatrix} \qquad \widehat{U}_i^{-1} = \begin{bmatrix} 1 & \hat{\beta}_1 & \dots & \dots & \hat{\beta}_1\hat{\beta}_2\cdots\hat{\beta}_{i-1} \\ 0 & 1 & \hat{\beta}_2 & \dots & \hat{\beta}_2\cdots\hat{\beta}_{i-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & \hat{\beta}_{i-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

Attainable accuracy of simple pipelined CG

$$\|G_i\| \leq \frac{O(\varepsilon)}{1 - O(\varepsilon)} \left(\kappa(\widehat{U}_i) \|A\| \|\widehat{P}_i\| + \|A\| \|\widehat{R}_i\| \|\widehat{U}_i^{-1}\| \right)$$

$$\widehat{U}_i = \begin{bmatrix} 1 & -\hat{\beta}_1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & 1 & -\hat{\beta}_{i-1} \\ 0 & \dots & 0 & 1 \end{bmatrix} \qquad \widehat{U}_i^{-1} = \begin{bmatrix} 1 & \hat{\beta}_1 & \dots & \dots & \hat{\beta}_1\hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\ 0 & 1 & \hat{\beta}_2 & \dots & \hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & \hat{\beta}_{i-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

$$\beta_\ell \beta_{\ell+1} \cdots \beta_j = \frac{\|r_j\|^2}{\|r_{\ell-1}\|^2}, \qquad \ell < j$$

Attainable accuracy of simple pipelined CG

$$\|G_i\| \leq \frac{O(\varepsilon)}{1 - O(\varepsilon)} \left(\kappa(\widehat{U}_i) \|A\| \left\|\widehat{P}_i\right\| + \|A\| \left\|\widehat{R}_i\right\| \left\|\widehat{U}_i^{-1}\right\| \right)$$

$$\widehat{U}_i = \begin{bmatrix} 1 & -\hat{\beta}_1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & 1 & -\hat{\beta}_{i-1} \\ 0 & \dots & 0 & 1 \end{bmatrix} \qquad \widehat{U}_i^{-1} = \begin{bmatrix} 1 & \hat{\beta}_1 & \dots & \dots & \hat{\beta}_1 \hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\ 0 & 1 & \hat{\beta}_2 & \dots & \hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & \hat{\beta}_{i-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

$$\beta_\ell \beta_{\ell+1} \cdots \beta_j = \frac{\left\|r_j\right\|^2}{\|r_{\ell-1}\|^2}, \qquad \ell < j$$

- Residual oscillations can cause these factors to be large!
- Errors in computed recurrence coefficients can be amplified!

Attainable accuracy of simple pipelined CG

$$\|G_i\| \leq \frac{O(\varepsilon)}{1-O(\varepsilon)}\left(\kappa(\widehat{U}_i)\|A\|\left\|\widehat{P}_i\right\| + \|A\|\left\|\widehat{R}_i\right\|\left\|\widehat{U}_i^{-1}\right\|\right)$$

$$\widehat{U}_i = \begin{bmatrix} 1 & -\hat{\beta}_1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & 1 & -\hat{\beta}_{i-1} \\ 0 & \dots & 0 & 1 \end{bmatrix} \qquad \widehat{U}_i^{-1} = \begin{bmatrix} 1 & \hat{\beta}_1 & \dots & \dots & \hat{\beta}_1\hat{\beta}_2\cdots\hat{\beta}_{i-1} \\ 0 & 1 & \hat{\beta}_2 & \dots & \hat{\beta}_2\cdots\hat{\beta}_{i-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & \hat{\beta}_{i-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

$$\beta_\ell\beta_{\ell+1}\cdots\beta_j = \frac{\left\|r_j\right\|^2}{\|r_{\ell-1}\|^2}, \qquad \ell < j$$

- Residual oscillations can cause these factors to be large!
- Errors in computed recurrence coefficients can be amplified!
  - Resembles results for attainable accuracy in STCG (3-term)

Attainable accuracy of simple pipelined CG

$$\|G_i\| \leq \frac{O(\varepsilon)}{1-O(\varepsilon)}\left(\kappa(\widehat{U}_i)\|A\|\|\widehat{P}_i\| + \|A\|\|\widehat{R}_i\|\|\widehat{U}_i^{-1}\|\right)$$

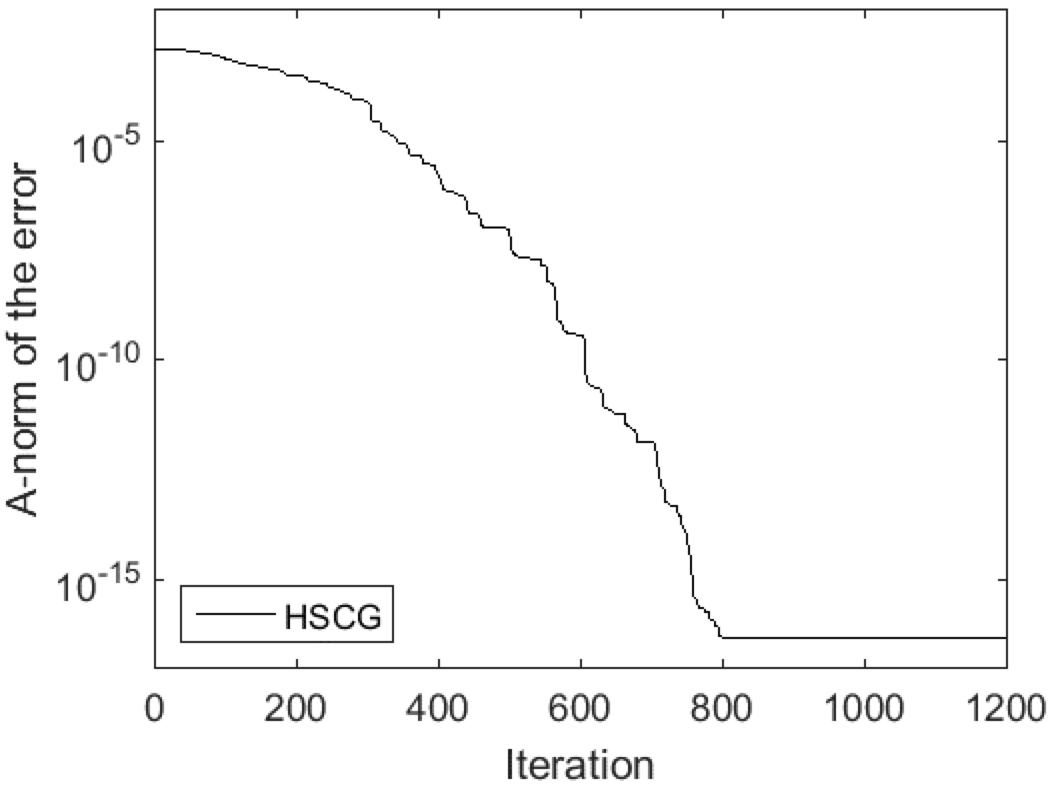
$$\widehat{U}_i = \begin{bmatrix} 1 & -\hat{\beta}_1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & 1 & -\hat{\beta}_{i-1} \\ 0 & \dots & 0 & 1 \end{bmatrix} \qquad \widehat{U}_i^{-1} = \begin{bmatrix} 1 & \hat{\beta}_1 & \dots & \dots & \hat{\beta}_1\hat{\beta}_2\cdots\hat{\beta}_{i-1} \\ 0 & 1 & \hat{\beta}_2 & \dots & \hat{\beta}_2\cdots\hat{\beta}_{i-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & \hat{\beta}_{i-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

$$\beta_\ell\beta_{\ell+1}\cdots\beta_j = \frac{\|r_j\|^2}{\|r_{\ell-1}\|^2}, \qquad \ell < j$$

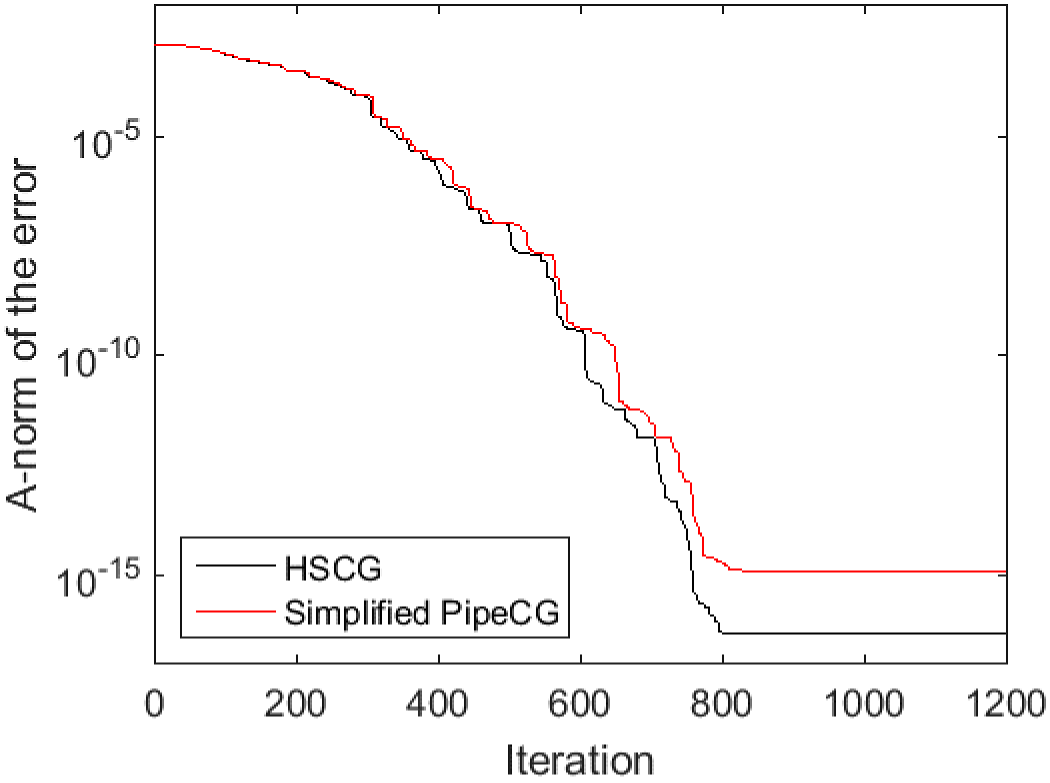
- Residual oscillations can cause these factors to be large!
- Errors in computed recurrence coefficients can be amplified!
  - Resembles results for attainable accuracy in STCG (3-term)

- Seemingly innocuous change can cause **drastic** loss of accuracy
- For analysis of attainable accuracy in GVCG, see [Cools et al., 2018]

Simple pipelined CG

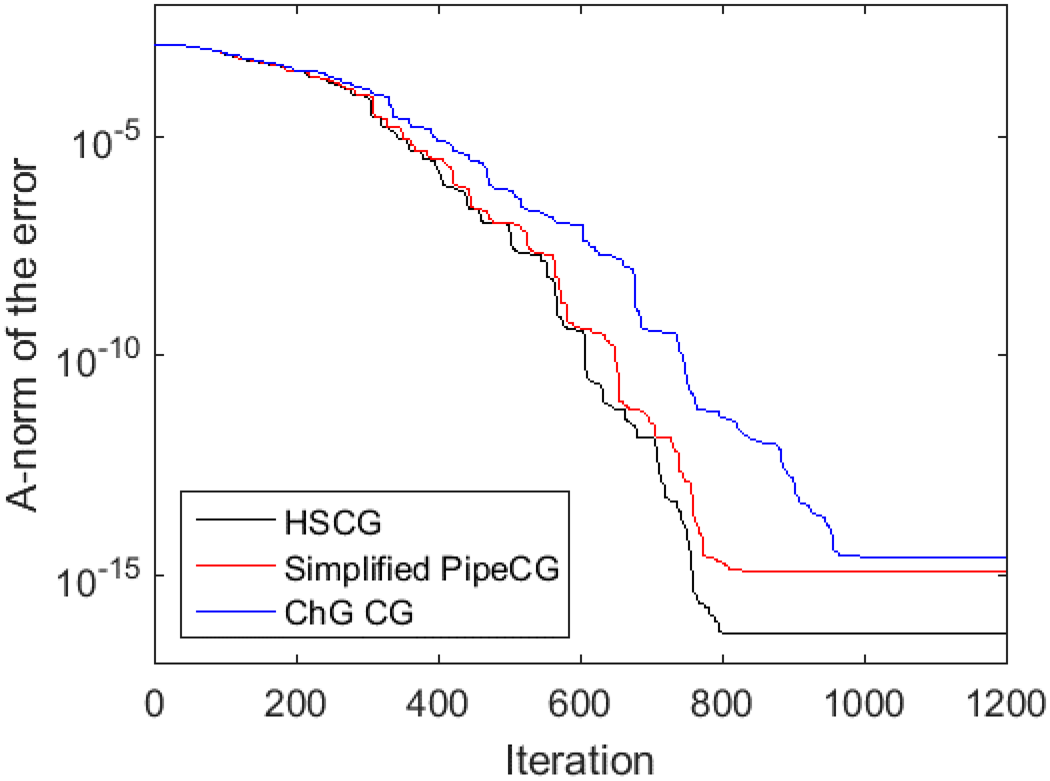


Simple pipelined CG



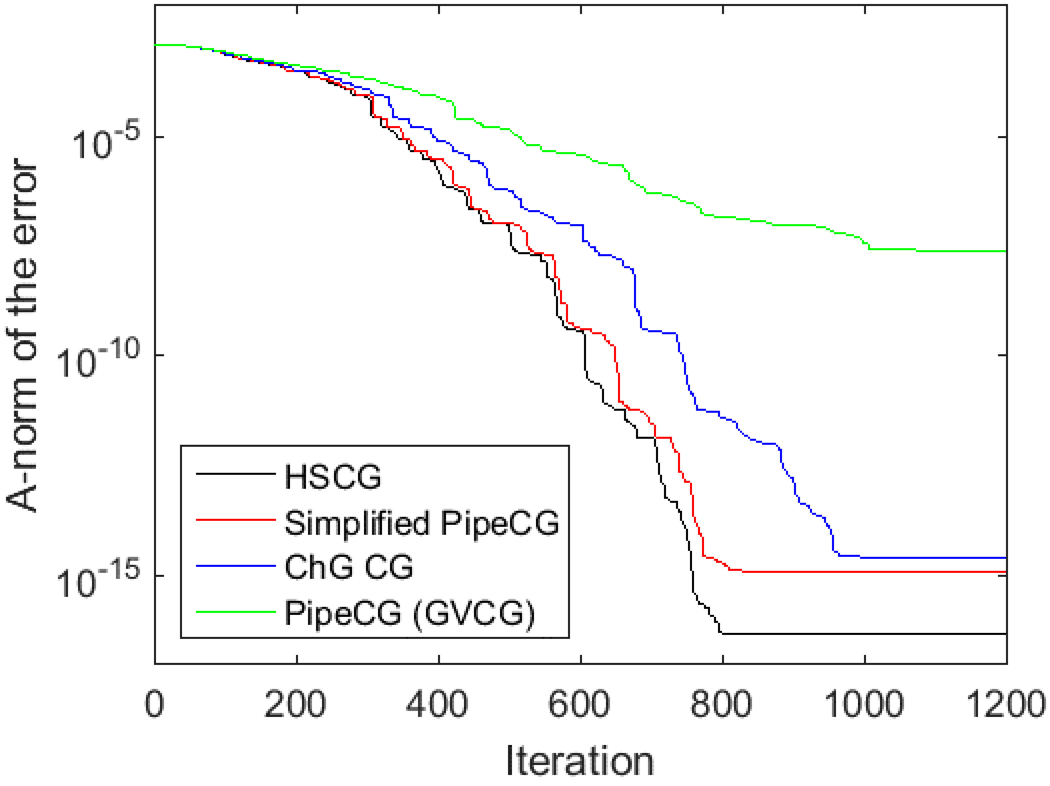
effect of using auxiliary vector $s_i \equiv Ap_i$

Simple pipelined CG



effect of changing formula for recurrence coefficient α and using auxiliary vector $s_i \equiv Ap_i$

Simple pipelined CG



effect of changing formula for recurrence coefficient α and using auxiliary vectors $s_i \equiv Ap_i, w_i \equiv Ar_i, z_i \equiv A^2 r_i$

Towards understanding convergence delay

- Coefficients α and β (related to entries of T_i) determine distribution functions $\omega^{(i)}(\lambda)$ which approximate distribution function $\omega(\lambda)$ determined by inputs A, b, x_0 in terms of the ith Gauss-Christoffel quadrature
- CG method = matrix formulation of Gauss-Christoffel quadrature (see, e.g., [Liesen & Strakoš, 2013])
- A-norm of CG error for $f(\lambda) = \lambda^{-1}$ given as scaled quadrature error

$$\int \lambda^{-1} d\omega(\lambda) = \sum_{\ell=1}^{i} \omega_\ell^{(i)} \left\{\theta_\ell^{(i)}\right\}^{-1} + \frac{\|x - x_i\|_A^2}{\|r_0\|^2}$$

Towards understanding convergence delay

- Coefficients α and β (related to entries of T_i) determine distribution functions $\omega^{(i)}(\lambda)$ which approximate distribution function $\omega(\lambda)$ determined by inputs A, b, x_0 in terms of the ith Gauss-Christoffel quadrature
- CG method = matrix formulation of Gauss-Christoffel quadrature (see, e.g., [Liesen & Strakoš, 2013])
- A-norm of CG error for $f(\lambda) = \lambda^{-1}$ given as scaled quadrature error

$$\int \lambda^{-1} d\omega(\lambda) = \sum_{\ell=1}^{i} \omega_\ell^{(i)} \left\{\theta_\ell^{(i)}\right\}^{-1} + \frac{\|x - x_i\|_A^2}{\|r_0\|^2}$$

Towards understanding convergence delay

- Coefficients α and β (related to entries of T_i) determine distribution functions $\omega^{(i)}(\lambda)$ which approximate distribution function $\omega(\lambda)$ determined by inputs A, b, x_0 in terms of the ith Gauss-Christoffel quadrature
- CG method = matrix formulation of Gauss-Christoffel quadrature (see, e.g., [Liesen & Strakoš, 2013])
- A-norm of CG error for $f(\lambda) = \lambda^{-1}$ given as scaled quadrature error

$$\int \lambda^{-1} d\omega(\lambda) = \sum_{\ell=1}^{i} \omega_\ell^{(i)} \left\{\theta_\ell^{(i)}\right\}^{-1} + \frac{\|x - x_i\|_A^2}{\|r_0\|^2}$$

- For particular CG implementation, can the computed $\widehat{\omega}^{(i)}(\lambda)$ be associated with some distribution function $\widehat{\omega}(\lambda)$ related to the distribution function $\omega(\lambda)$, i.e.,

$$\int \lambda^{-1} d\omega(\lambda) \approx \int \lambda^{-1} d\widehat{\omega}(\lambda) = \sum_{\ell=1}^{i} \widehat{\omega}_\ell^{(i)} \left\{\hat{\theta}_\ell^{(i)}\right\}^{-1} + \frac{\|x - \hat{x}_i\|_A^2}{\|r_0\|^2} + F_i$$

where F_i is small relative to error term?

Towards understanding convergence delay

- Coefficients α and β (related to entries of T_i) determine distribution functions $\omega^{(i)}(\lambda)$ which approximate distribution function $\omega(\lambda)$ determined by inputs A, b, x_0 in terms of the ith Gauss-Christoffel quadrature
- CG method = matrix formulation of Gauss-Christoffel quadrature (see, e.g., [Liesen & Strakoš, 2013])
- A-norm of CG error for $f(\lambda) = \lambda^{-1}$ given as scaled quadrature error

$$\int \lambda^{-1} d\omega(\lambda) = \sum_{\ell=1}^{i} \omega_\ell^{(i)} \left\{\theta_\ell^{(i)}\right\}^{-1} + \frac{\|x - x_i\|_A^2}{\|r_0\|^2}$$

- For particular CG implementation, can the computed $\widehat{\omega}^{(i)}(\lambda)$ be associated with some distribution function $\widehat{\omega}(\lambda)$ related to the distribution function $\omega(\lambda)$, i.e.,

$$\int \lambda^{-1} d\omega(\lambda) \approx \int \lambda^{-1} d\widehat{\omega}(\lambda) = \sum_{\ell=1}^{i} \widehat{\omega}_\ell^{(i)} \left\{\hat{\theta}_\ell^{(i)}\right\}^{-1} + \frac{\|x - \hat{x}_i\|_A^2}{\|r_0\|^2} + F_i$$

  where F_i is small relative to error term?
- For classical CG, yes; proved by Greenbaum [1989]

Towards understanding convergence delay

- Coefficients α and β (related to entries of T_i) determine distribution functions $\omega^{(i)}(\lambda)$ which approximate distribution function $\omega(\lambda)$ determined by inputs A, b, x_0 in terms of the ith Gauss-Christoffel quadrature
- CG method = matrix formulation of Gauss-Christoffel quadrature (see, e.g., [Liesen & Strakoš, 2013])
- A-norm of CG error for $f(\lambda) = \lambda^{-1}$ given as scaled quadrature error

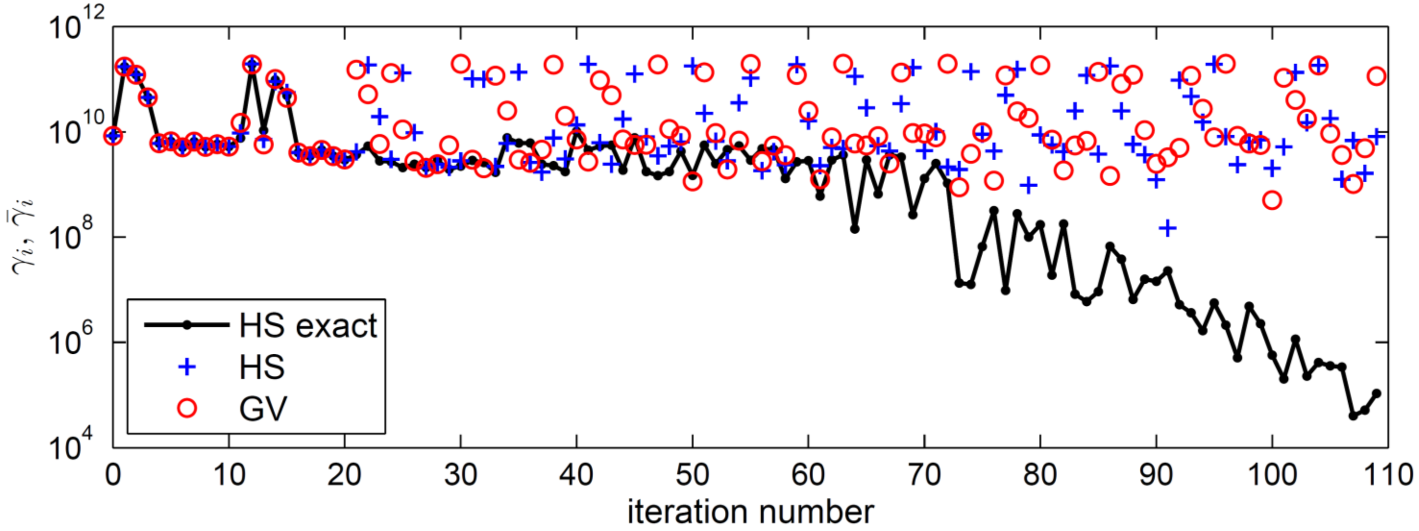
$$\int \lambda^{-1} d\omega(\lambda) = \sum_{\ell=1}^{i} \omega_\ell^{(i)} \left\{\theta_\ell^{(i)}\right\}^{-1} + \frac{\|x - x_i\|_A^2}{\|r_0\|^2}$$

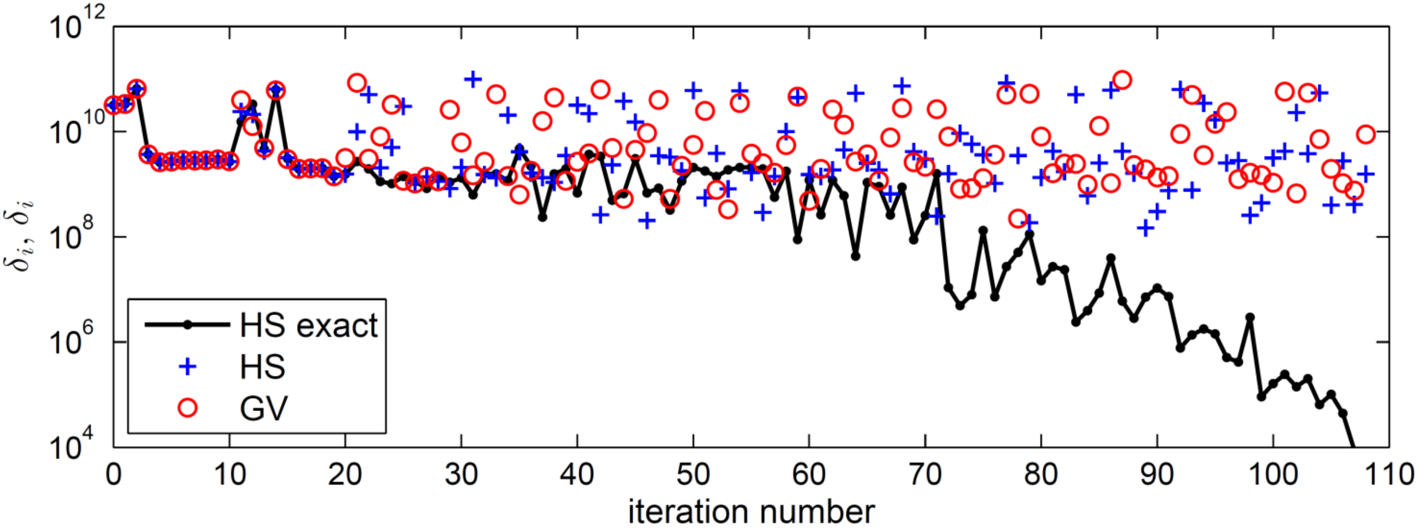
- For particular CG implementation, can the computed $\widehat{\omega}^{(i)}(\lambda)$ be associated with some distribution function $\widehat{\omega}(\lambda)$ related to the distribution function $\omega(\lambda)$, i.e.,

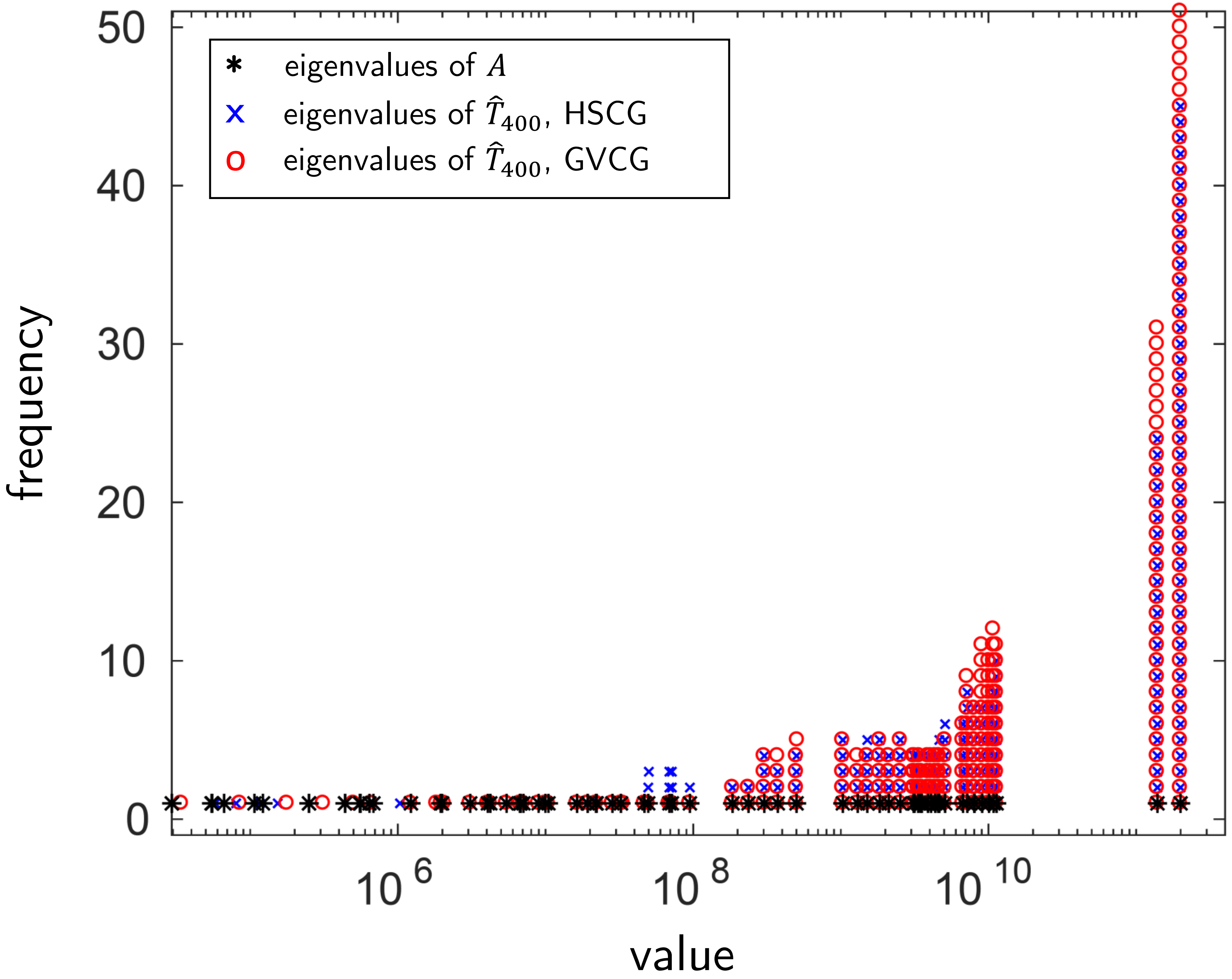
$$\int \lambda^{-1} d\omega(\lambda) \approx \int \lambda^{-1} d\widehat{\omega}(\lambda) = \sum_{\ell=1}^{i} \widehat{\omega}_\ell^{(i)} \left\{\hat{\theta}_\ell^{(i)}\right\}^{-1} + \frac{\|x - \hat{x}_i\|_A^2}{\|r_0\|^2} + F_i$$

  where F_i is small relative to error term?
- For classical CG, yes; proved by Greenbaum [1989]
- For pipelined CG, THOROUGH ANALYSIS NEEDED!

Differences in entries γ_i, δ_i in Jacobi matrices T_i in HSCG vs. GVCG (matrix bcsstk03)






frequency
value
0
10
20
30
40
50
10^6
10^8
10^{10}
* eigenvalues of A
x eigenvalues of $\hat{T}_{400}$, HSCG
o eigenvalues of $\hat{T}_{400}$, GVCG

s-step Krylov Subspace Methods

- Idea: Compute blocks of s iterations at once
  - Generate an $O(s)$ dimensional Krylov subspace basis; block orthogonalization
  - Communicate every s iterations instead of every iteration
  - Reduces number of synchronizations per iteration by a factor of s

s-step Krylov Subspace Methods

- Idea: Compute blocks of s iterations at once
  - Generate an $O(s)$ dimensional Krylov subspace basis; block orthogonalization
  - Communicate every s iterations instead of every iteration
  - Reduces number of synchronizations per iteration by a factor of s

- First related work: s-dimensional steepest descent, least squares
  - [Khabaza, 1963], [Forsythe, 1968], [Marchuk and Kuznecov, 1968]

- Flurry of work on s-step Krylov subspace methods in 1980's/1990's; e.g.,
  - [Van Rosendale, 1983]; [Chronopoulos and Gear, 1989], [de Sturler, 1991], [de Sturler and van der Vorst, 1995],...

s-step Krylov Subspace Methods

- Idea: Compute blocks of s iterations at once
  - Generate an $O(s)$ dimensional Krylov subspace basis; block orthogonalization
  - Communicate every s iterations instead of every iteration
  - Reduces number of synchronizations per iteration by a factor of s

- First related work: s-dimensional steepest descent, least squares
  - [Khabaza, 1963], [Forsythe, 1968], [Marchuk and Kuznecov, 1968]

- Flurry of work on s-step Krylov subspace methods in 1980's/1990's; e.g.,
  - [Van Rosendale, 1983]; [Chronopoulos and Gear, 1989], [de Sturler, 1991], [de Sturler and van der Vorst, 1995],...

Recent use in many applications

- combustion, cosmology [Williams, C., et al., IPDPS, 2014]
  - geoscience dynamics [Anciaux-Sedrakian et al., 2016]
  - far-field scattering [Zhang et al., 2016]
  - wafer defect detection [Zhang et al., 2016]

s-step Krylov Subspace Methods

- Idea: Compute blocks of s iterations at once
  - Generate an $O(s)$ dimensional Krylov subspace basis; block orthogonalization
  - Communicate every s iterations instead of every iteration
  - Reduces number of synchronizations per iteration by a factor of s

- First related work: s-dimensional steepest descent, least squares
  - [Khabaza, 1963], [Forsythe, 1968], [Marchuk and Kuznecov, 1968]

- Flurry of work on s-step Krylov subspace methods in 1980's/1990's; e.g.,
  - [Van Rosendale, 1983]; [Chronopoulos and Gear, 1989], [de Sturler, 1991], [de Sturler and van der Vorst, 1995],...

Recent use in many applications

- combustion, cosmology [Williams, C., et al., IPDPS, 2014]
  - geoscience dynamics [Anciaux-Sedrakian et al., 2016]
  - far-field scattering [Zhang et al., 2016]
  - wafer defect detection [Zhang et al., 2016]

up to **4.2x** on 24K cores on Cray XE6

s-step CG

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

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

s-step CG

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

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

s steps of s-step CG:

s-step CG

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

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

s steps of s-step CG:

Expand solution space s dimensions at once

Compute "basis" matrix $\mathcal{Y}$ such that

$$\operatorname{span}(\mathcal{Y}) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$$

according to the recurrence $A\underline{\mathcal{Y}} = \mathcal{Y}\,\mathcal{B}$

Compute inner products basis vectors in one synchronization

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

$O(1)$
messages

s-step CG

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

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

s steps of s-step CG:

Expand solution space s dimensions at once

Compute "basis" matrix $\mathcal{Y}$ such that

$$\mathrm{span}(\mathcal{Y}) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$$

according to the recurrence $A\underline{\mathcal{Y}} = \mathcal{Y}\,\mathcal{B}$

Compute inner products basis vectors in one synchronization

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

$O(1)$ messages

Compute s iterations of vector updates

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

$$x_{i+j} - x_i = \mathcal{Y}x_j', \qquad r_{i+j} = \mathcal{Y}r_j', \qquad p_{i+j} = \mathcal{Y}p_j'$$

no data movement

s-step CG

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

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

s steps of s-step CG:

Number of synchronizations per step reduced by factor of $O(s)$!

Expand solution space s dimensions at once

Compute "basis" matrix $\mathcal{Y}$ such that

$$\operatorname{span}(\mathcal{Y}) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$$

according to the recurrence $A\underline{\mathcal{Y}} = \mathcal{Y}\,\mathcal{B}$

Compute inner products basis vectors in one synchronization

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

$O(1)$ messages

Compute s iterations of vector updates

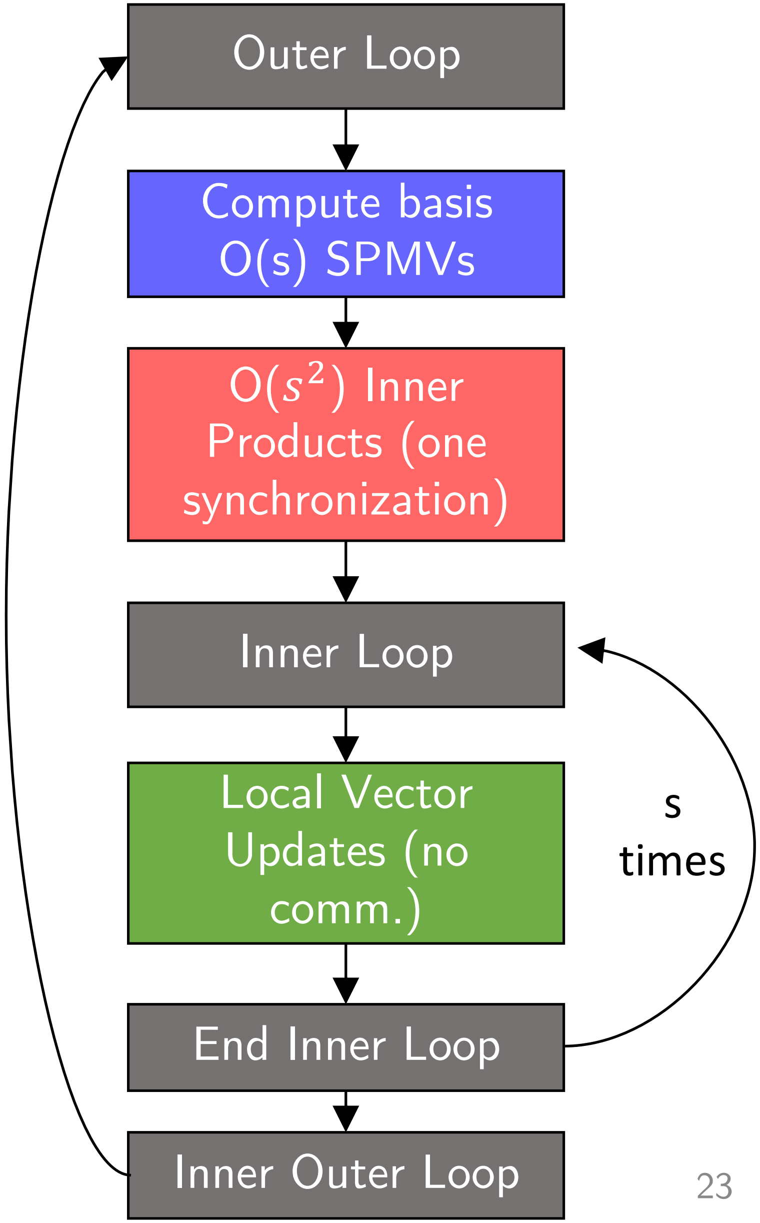
Perform s iterations of vector updates by updating coordinates in basis $\mathcal{Y}$:

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

no data movement

s-step CG

```
$r_0 = b - Ax_0, p_0 = r_0$
for $k$ = 0:nmax/$s$
    Compute $\mathcal{Y}_k$ and $\mathcal{B}_k$ such that $A\underline{\mathcal{Y}}_k = \mathcal{Y}_k\mathcal{B}_k$ and
        span($\mathcal{Y}_k$) $= \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$
    $\mathcal{G}_k = \mathcal{Y}_k^T\mathcal{Y}_k$
    $x'_0 = 0, r'_0 = e_{s+2}, p'_0 = e_1$
    for $j = 1{:}\,s$
        $\alpha_{sk+j-1} = \frac{r'^T_{j-1}\mathcal{G}_k r'_{j-1}}{p'^T_{j-1}\mathcal{G}_k\mathcal{B}_k p'_{j-1}}$
        $x'_j = x'_{j-1} + \alpha_{sk+j-1}p'_{j-1}$
        $r'_j = r'_{j-1} - \alpha_{sk+j-1}\mathcal{B}_k p'_{j-1}$
        $\beta_{sk+j} = \frac{r'^T_j\mathcal{G}_k r'_j}{r'^T_{j-1}\mathcal{G}_k r'_{j-1}}$
        $p'_j = r'_j + \beta_{sk+j}p'_{j-1}$
    end

$[x_{s(k+1)} - x_{sk}, r_{s(k+1)}, p_{s(k+1)}] = \mathcal{Y}_k[x'_s, r'_s, p'_s]$
end
```



Outer Loop

Compute basis O(s) SPMVs

O(s^2) Inner Products (one synchronization)

Inner Loop

Local Vector Updates (no comm.)

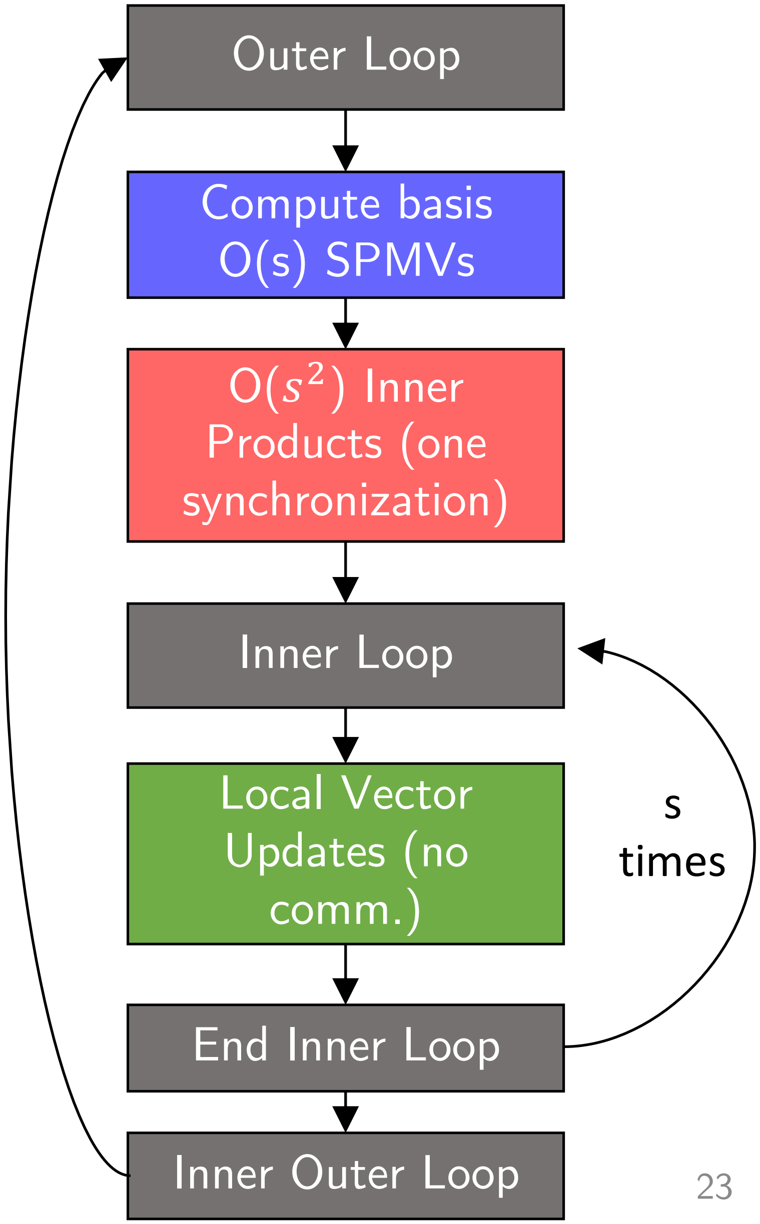
End Inner Loop

Inner Outer Loop

s times

s-step CG

```
$r_0 = b - Ax_0, p_0 = r_0$
for $k$ = 0:nmax/$s$
    Compute $\mathcal{Y}_k$ and $\mathcal{B}_k$ such that $A\underline{\mathcal{Y}}_k = \mathcal{Y}_k\mathcal{B}_k$ and
        span$(\mathcal{Y}_k) = \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$
    $\mathcal{G}_k = \mathcal{Y}_k^T\mathcal{Y}_k$
    $x'_0 = 0, r'_0 = e_{s+2}, p'_0 = e_1$
    for $j = 1{:}\,s$
        $\alpha_{sk+j-1} = \frac{r'^T_{j-1}\mathcal{G}_k r'_{j-1}}{p'^T_{j-1}\mathcal{G}_k\mathcal{B}_k p'_{j-1}}$
        $x'_j = x'_{j-1} + \alpha_{sk+j-1}p'_{j-1}$
        $r'_j = r'_{j-1} - \alpha_{sk+j-1}\mathcal{B}_k p'_{j-1}$
        $\beta_{sk+j} = \frac{r'^T_j\mathcal{G}_k r'_j}{r'^T_{j-1}\mathcal{G}_k r'_{j-1}}$
        $p'_j = r'_j + \beta_{sk+j}p'_{j-1}$
    end

$[x_{s(k+1)} - x_{sk}, r_{s(k+1)}, p_{s(k+1)}] = \mathcal{Y}_k[x'_s, r'_s, p'_s]$
end
```



Outer Loop

Compute basis O(s) SPMVs

O(s^2) Inner Products (one synchronization)

Inner Loop

Local Vector Updates (no comm.)

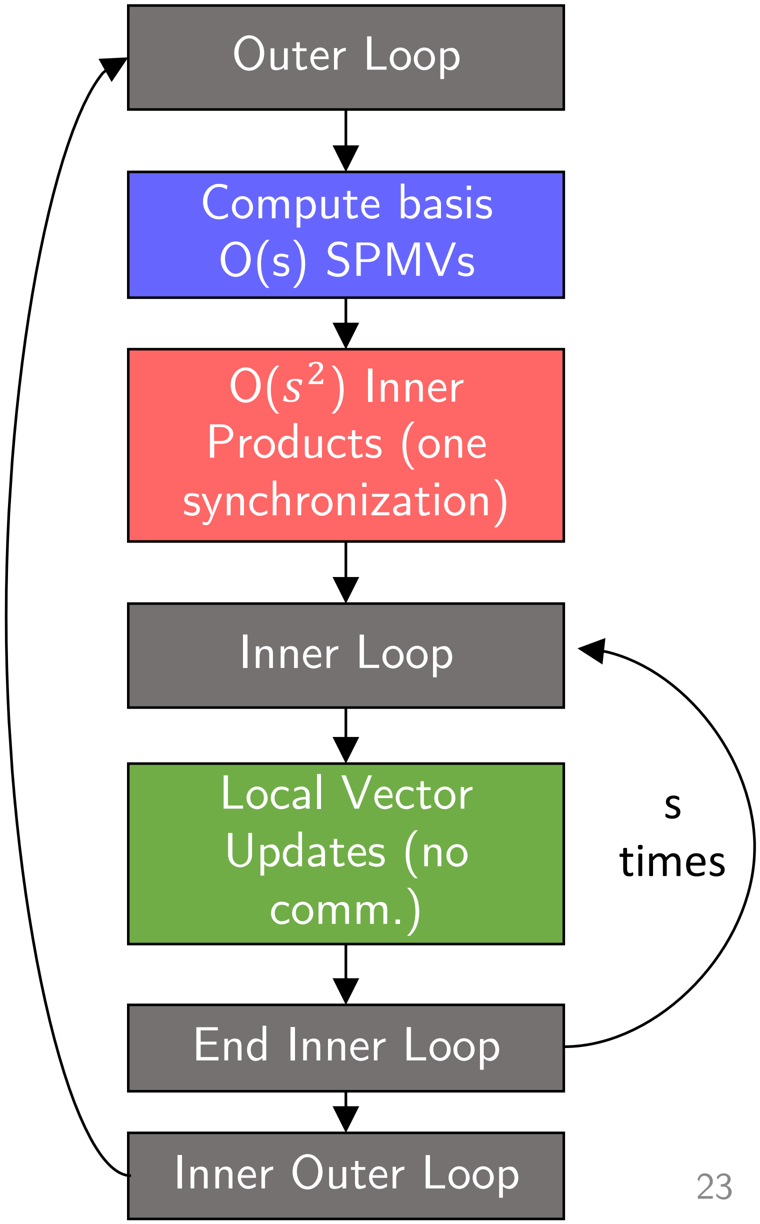
s times

End Inner Loop

Inner Outer Loop

s-step CG

```
$r_0 = b - Ax_0, p_0 = r_0$
for $k$ = 0:nmax/$s$
    Compute $\mathcal{Y}_k$ and $\mathcal{B}_k$ such that $A\underline{\mathcal{Y}}_k = \mathcal{Y}_k\mathcal{B}_k$ and
        span$(\mathcal{Y}_k) = \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$
    $\mathcal{G}_k = \mathcal{Y}_k^T\mathcal{Y}_k$
    $x_0' = 0, r_0' = e_{s+2}, p_0' = e_1$
    for $j = 1: s$
        $\alpha_{sk+j-1} = \frac{r_{j-1}'^T\mathcal{G}_k r_{j-1}'}{p_{j-1}'^T\mathcal{G}_k\mathcal{B}_k p_{j-1}'}$
        $x_j' = x_{j-1}' + \alpha_{sk+j-1}p_{j-1}'$
        $r_j' = r_{j-1}' - \alpha_{sk+j-1}\mathcal{B}_k p_{j-1}'$
        $\beta_{sk+j} = \frac{r_j'^T\mathcal{G}_k r_j'}{r_{j-1}'^T\mathcal{G}_k r_{j-1}'}$
        $p_j' = r_j' + \beta_{sk+j}p_{j-1}'$
    end

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



Outer Loop

Compute basis O(s) SPMVs

O(s^2) Inner Products (one synchronization)

Inner Loop

Local Vector Updates (no comm.)

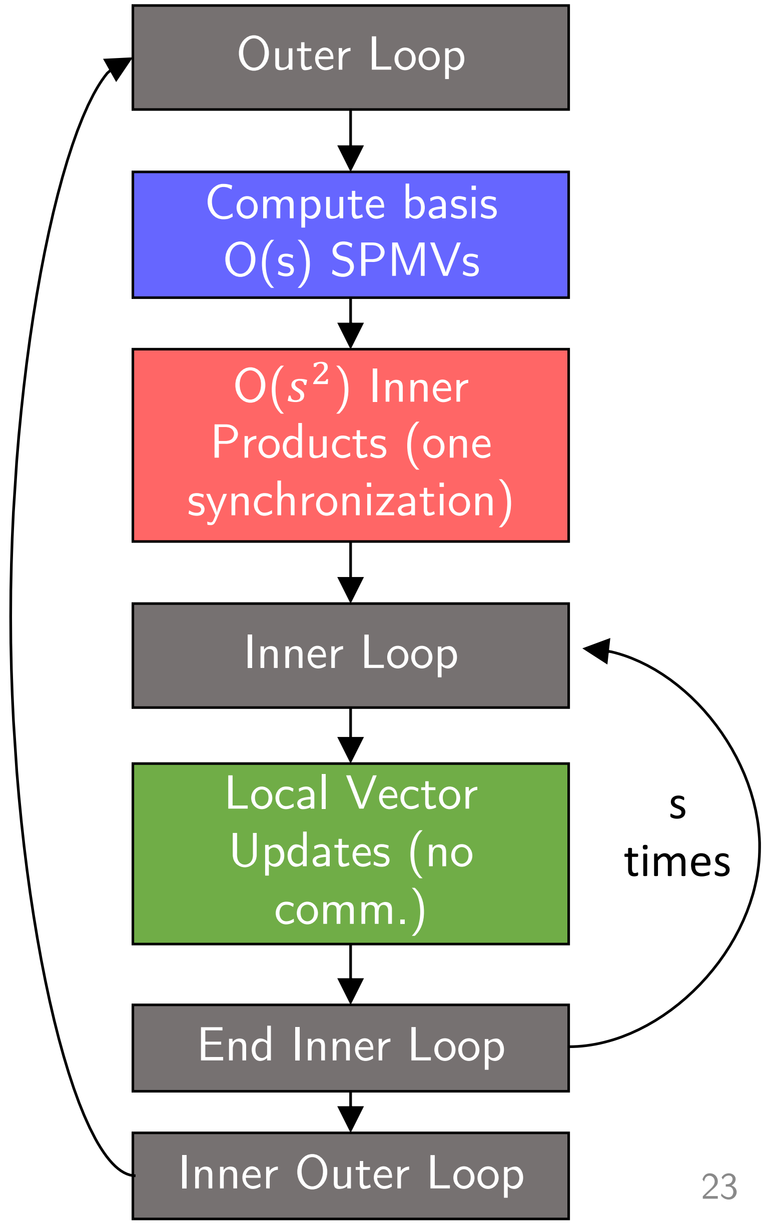
End Inner Loop

Inner Outer Loop

s times

s-step CG

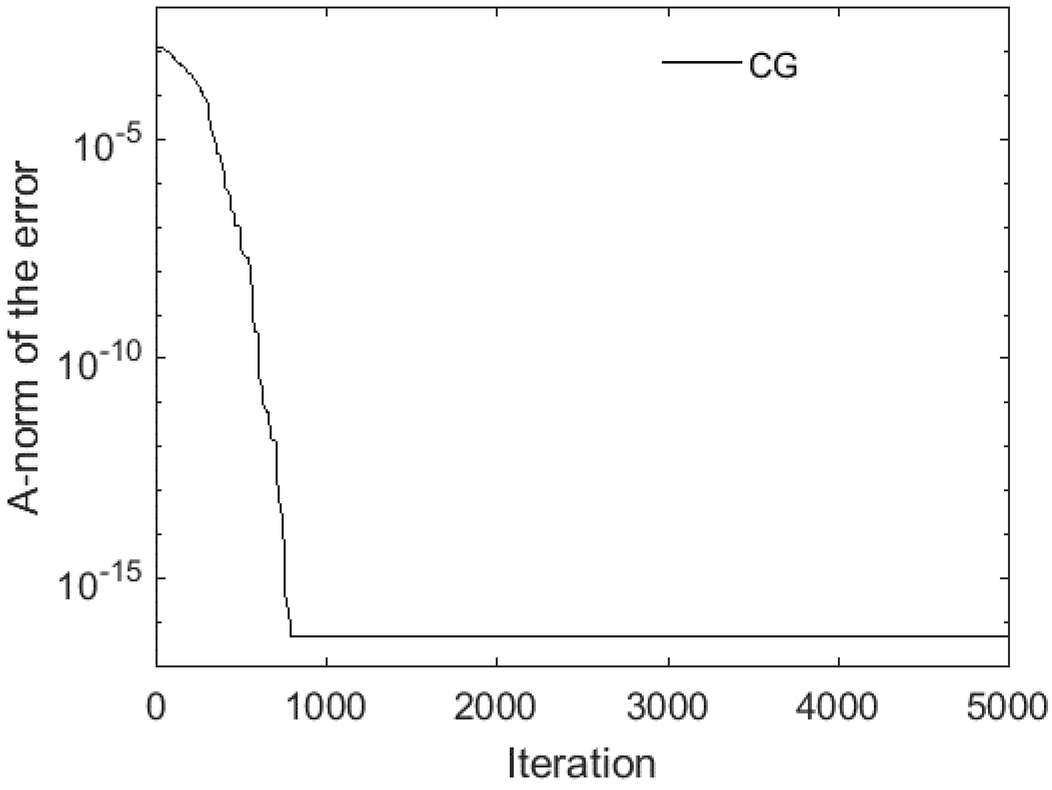
```
$r_0 = b - Ax_0, p_0 = r_0$
for $k$ = 0:nmax/$s$
    Compute $\mathcal{Y}_k$ and $\mathcal{B}_k$ such that $A\underline{\mathcal{Y}}_k = \mathcal{Y}_k\mathcal{B}_k$ and
        span($\mathcal{Y}_k$) = $\mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$
    $\mathcal{G}_k = \mathcal{Y}_k^T\mathcal{Y}_k$
    $x_0' = 0, r_0' = e_{s+2}, p_0' = e_1$
    for $j = 1{:}\, s$
        $\alpha_{sk+j-1} = \frac{r_{j-1}'^T \mathcal{G}_k r_{j-1}'}{p_{j-1}'^T \mathcal{G}_k \mathcal{B}_k p_{j-1}'}$
        $x_j' = x_{j-1}' + \alpha_{sk+j-1} p_{j-1}'$
        $r_j' = r_{j-1}' - \alpha_{sk+j-1} \mathcal{B}_k p_{j-1}'$
        $\beta_{sk+j} = \frac{r_j'^T \mathcal{G}_k r_j'}{r_{j-1}'^T \mathcal{G}_k r_{j-1}'}$
        $p_j' = r_j' + \beta_{sk+j} p_{j-1}'$
    end

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



Numerical Behavior of s-step CG

A: bcsstk03 from UFSMC
b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

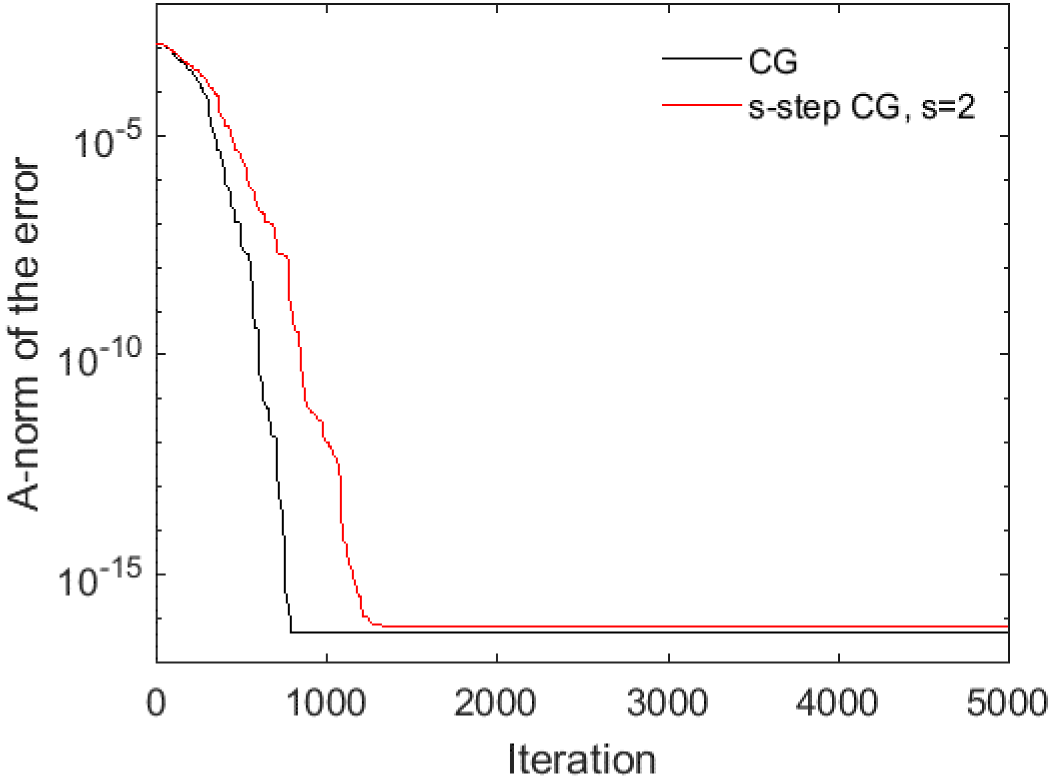
s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \dots, A^s p_i, r_i, Ar_i, \dots A^{s-1} r_i]$)



Numerical Behavior of s-step CG

A: bcsstk03 from UFSMC
b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

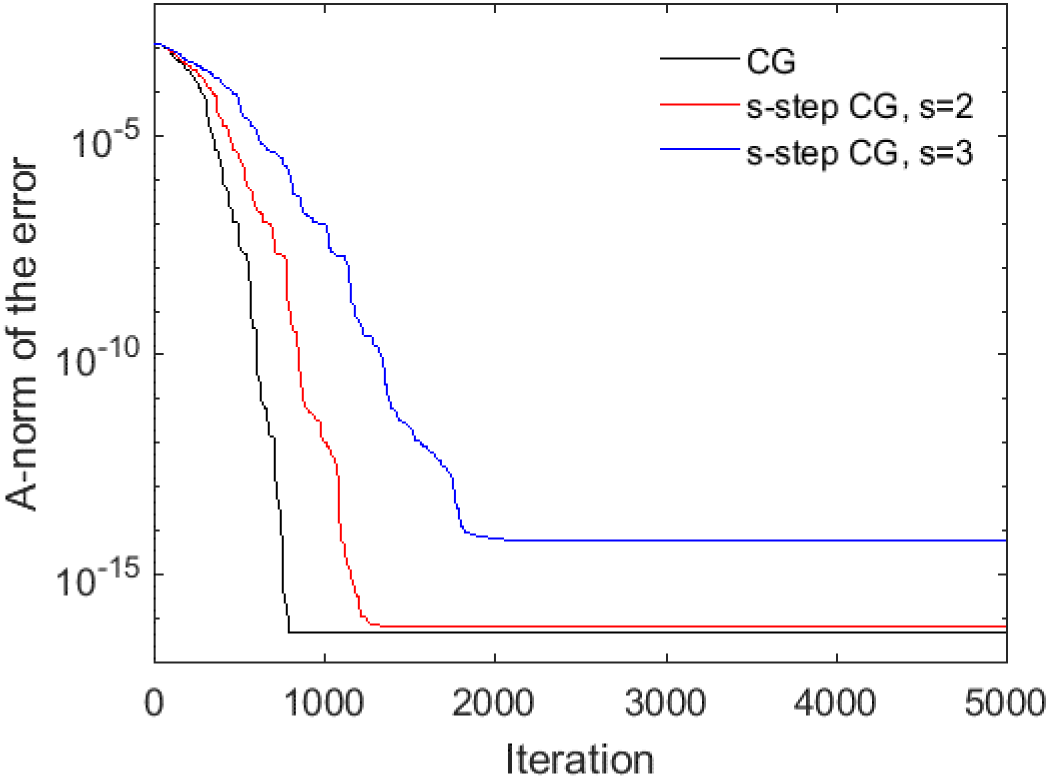
s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \dots, A^s p_i, r_i, Ar_i, \dots A^{s-1} r_i]$)



Numerical Behavior of s-step CG

A: bcsstk03 from UFSMC
b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

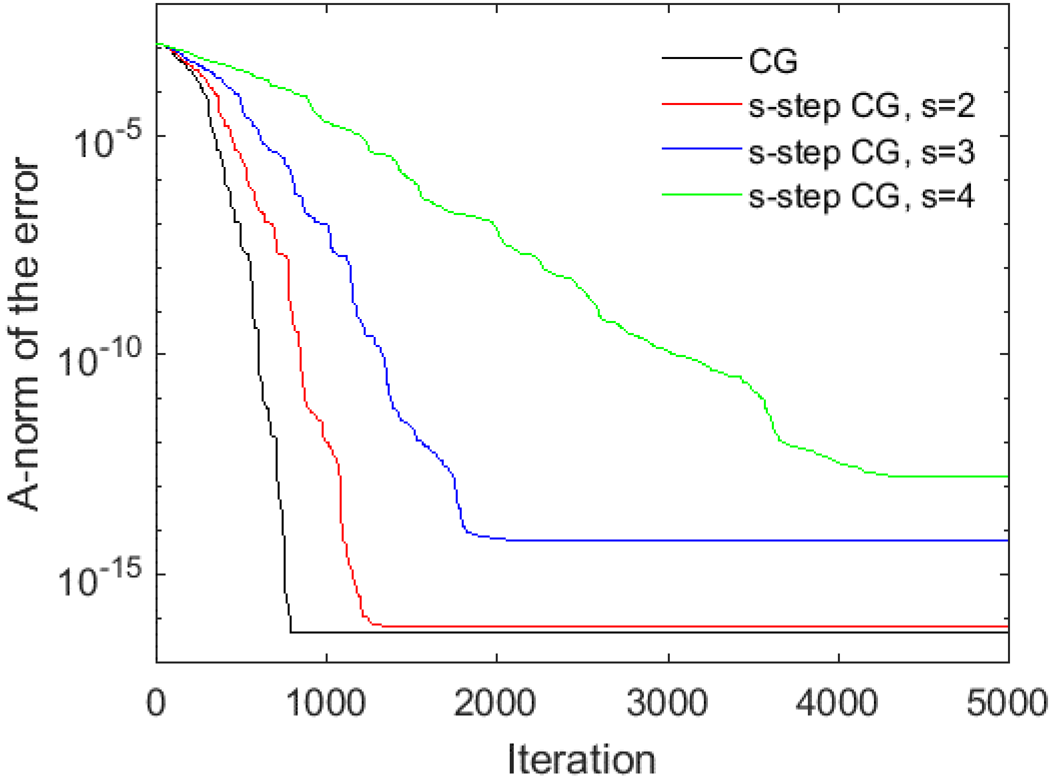
s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \dots, A^s p_i, r_i, Ar_i, \dots A^{s-1} r_i]$)



Numerical Behavior of s-step CG

A: bcsstk03 from UFSMC
b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

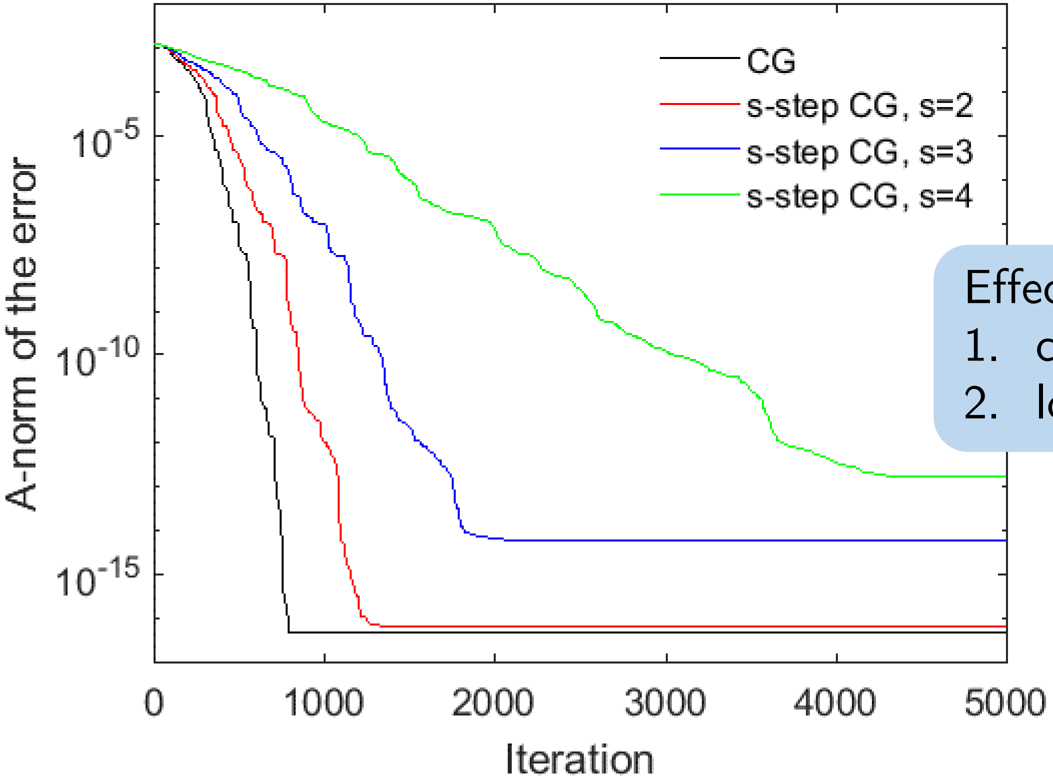
s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \dots, A^s p_i, r_i, Ar_i, \dots A^{s-1} r_i]$)



Numerical Behavior of s-step CG

A: bcsstk03 from UFSMC
b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \dots, A^s p_i, r_i, Ar_i, \dots A^{s-1} r_i]$)

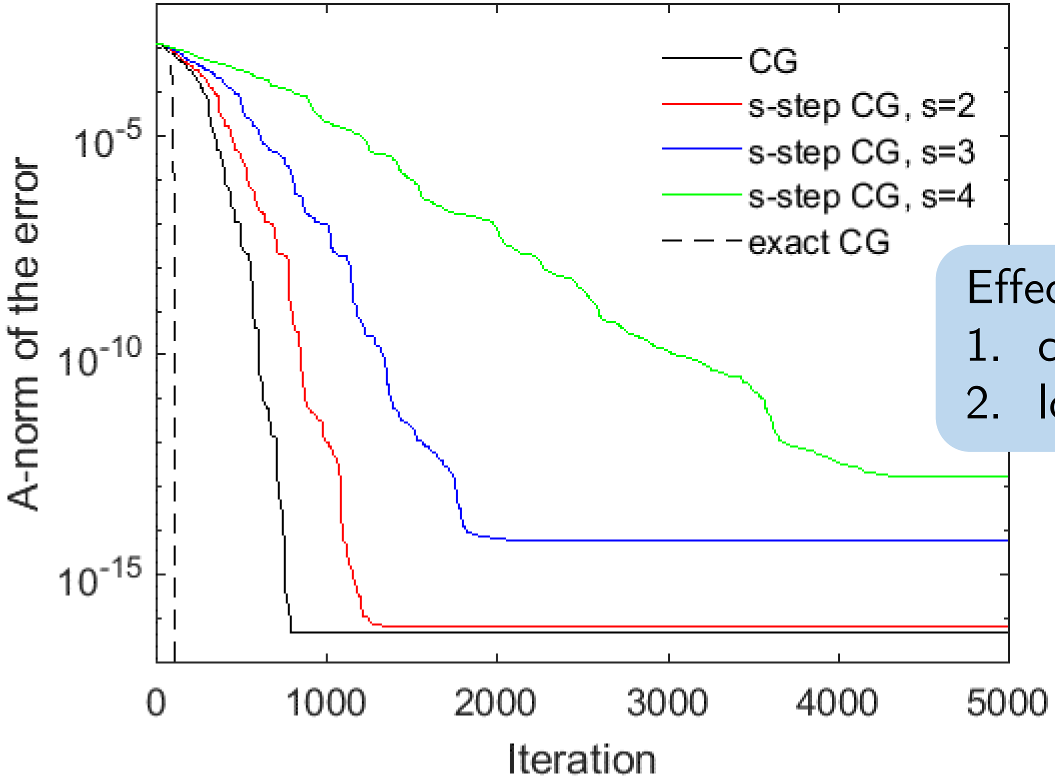


Effects of roundoff error:
1. convergence delay
2. loss of accuracy

Numerical Behavior of s-step CG

A: bcsstk03 from UFSMC
b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \ldots, A^s p_i, r_i, Ar_i, \ldots A^{s-1} r_i]$)



Effects of roundoff error:

1. convergence delay
2. loss of accuracy

Sources of Roundoff Error in s-step CG

Error in outer iteration k:

Computing the s-step Krylov subspace basis:

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

Updating coordinate vectors in the inner loop, $j = 1\colon s$:

$$\hat{x}'_j = \hat{x}'_{j-1} + \hat{q}'_{j-1} + \xi_j$$

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

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

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+s} = \hat{\mathcal{Y}}_k \hat{x}'_j + \hat{x}_{sk} + \phi_{sk+s}$$

$$\hat{r}_{sk+s} = \hat{\mathcal{Y}}_k \hat{r}'_j + \psi_{sk+s}$$

Sources of Roundoff Error in s-step CG

Error in outer iteration k:

Computing the s-step Krylov subspace basis:

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

Error in computing s-step basis

Updating coordinate vectors in the inner loop, $j = 1\colon s$:

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

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+s} = \hat{\mathcal{Y}}_k \hat{x}'_j + \hat{x}_{sk} + \phi_{sk+s}$$
$$\hat{r}_{sk+s} = \hat{\mathcal{Y}}_k \hat{r}'_j + \psi_{sk+s}$$

Sources of Roundoff Error in s-step CG

Error in outer iteration k:

Computing the s-step Krylov subspace basis:

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

Error in computing s-step basis

Updating coordinate vectors in the inner loop, $j = 1{:}\,s$:

$$\hat{x}'_j = \hat{x}'_{j-1} + \hat{q}'_{j-1} + \xi_j$$

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

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

Error in updating coefficient vectors

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+s} = \hat{\mathcal{Y}}_k \hat{x}'_j + \hat{x}_{sk} + \phi_{sk+s}$$

$$\hat{r}_{sk+s} = \hat{\mathcal{Y}}_k \hat{r}'_j + \psi_{sk+s}$$

Sources of Roundoff Error in s-step CG

Error in outer iteration k:

Computing the s-step Krylov subspace basis:

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

Error in computing s-step basis

Updating coordinate vectors in the inner loop, $j = 1{:}\,s$:

$$\hat{x}'_j = \hat{x}'_{j-1} + \hat{q}'_{j-1} + \xi_j$$

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

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

Error in updating coefficient vectors

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+s} = \hat{\mathcal{Y}}_k \hat{x}'_j + \hat{x}_{sk} + \phi_{sk+s}$$

$$\hat{r}_{sk+s} = \hat{\mathcal{Y}}_k \hat{r}'_j + \psi_{sk+s}$$

Error in basis change

Attainable Accuracy of s-step CG

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

For CG:

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

e.g., [van der Vorst and Ye, 2000], [Greenbaum, 1997]

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

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

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

[C., 2015]

where c is a low-degree polynomial in s

Roundoff Error in Lanczos vs. s-step Lanczos

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

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

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

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

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

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

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

Lanczos [Paige, 1976]

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

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

Roundoff Error in Lanczos vs. s-step Lanczos

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

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

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

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

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

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

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

Lanczos [Paige, 1976]

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

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

s-step Lanczos [C., Demmel, 2015]:

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

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

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

Convergence of Ritz Values in s-step Lanczos

- All results of Paige [1980], e.g., loss of orthogonality → eigenvalue convergence, hold for s-step Lanczos as long as

$$\left(\Gamma = c \cdot \max_{\ell \le k} \|\hat{\mathcal{Y}}_\ell^+\| \, \||\hat{\mathcal{Y}}_\ell|\|\right)$$

$$\Gamma \le \left(24\varepsilon(N + 11s + 15)\right)^{-1/2} \approx \frac{1}{\sqrt{N\varepsilon}}$$

Convergence of Ritz Values in s-step Lanczos

- All results of Paige [1980], e.g., loss of orthogonality → eigenvalue convergence, hold for s-step Lanczos as long as

$$\left(\Gamma = c \cdot \max_{\ell \le k} \|\hat{\mathcal{Y}}_\ell^+\| \, \||\hat{\mathcal{Y}}_\ell|\|\right)$$

$$\Gamma \le \left(24\varepsilon(N + 11s + 15)\right)^{-1/2} \approx \frac{1}{\sqrt{N\varepsilon}}$$

- Bounds on accuracy of Ritz values depend on Γ^2

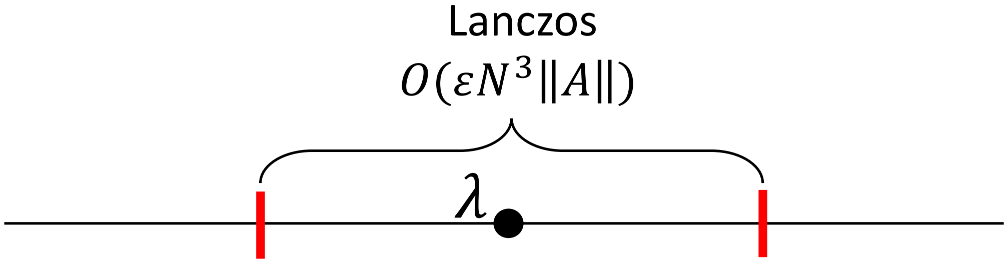
Convergence of Ritz Values in s-step Lanczos

- All results of Paige [1980], e.g., loss of orthogonality → eigenvalue convergence, hold for s-step Lanczos as long as

$\left(\Gamma = c \cdot \max_{\ell \le k} \|\hat{\mathcal{Y}}_\ell^+\| \, \||\hat{\mathcal{Y}}_\ell|\|\right)$

$$\Gamma \le \left(24\varepsilon(N + 11s + 15)\right)^{-1/2} \approx \frac{1}{\sqrt{N\varepsilon}}$$

- Bounds on accuracy of Ritz values depend on Γ^2



Convergence of Ritz Values in s-step Lanczos

- All results of Paige [1980], e.g., loss of orthogonality → eigenvalue convergence, hold for s-step Lanczos as long as

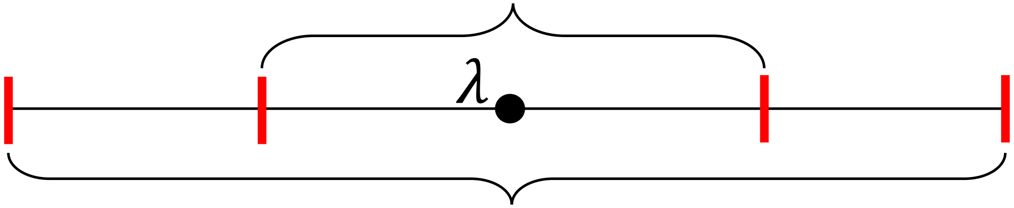
$$\left(\Gamma = c \cdot \max_{\ell \le k} \|\hat{\mathcal{Y}}_\ell^+\| \, \||\hat{\mathcal{Y}}_\ell|\|\right)$$

$$\Gamma \le \left(24\varepsilon(N + 11s + 15)\right)^{-1/2} \approx \frac{1}{\sqrt{N\varepsilon}}$$

- Bounds on accuracy of Ritz values depend on Γ^2

Lanczos

$O(\varepsilon N^3 \|A\|)$



$O(\varepsilon N^3 \|A\| \boldsymbol{\Gamma^2})$

s-step Lanczos

Convergence of Ritz Values in s-step Lanczos

- All results of Paige [1980], e.g., loss of orthogonality → eigenvalue convergence, hold for s-step Lanczos as long as

$$\left(\Gamma = c \cdot \max_{\ell \le k} \|\hat{\mathcal{Y}}_\ell^+\| \, \||\hat{\mathcal{Y}}_\ell|\|\right)$$

$$\Gamma \le \left(24\varepsilon(N + 11s + 15)\right)^{-1/2} \approx \frac{1}{\sqrt{N\varepsilon}}$$

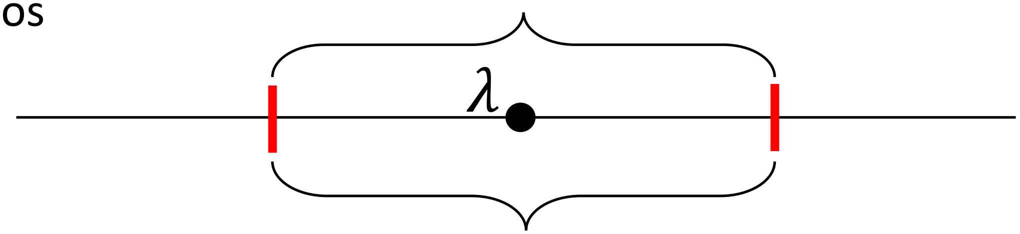
- Bounds on accuracy of Ritz values depend on Γ^2

If $\mathbf{\Gamma \approx 1}$:

s-step Lanczos behaves the same numerically as classical Lanczos

Lanczos

$O(\varepsilon N^3 \|A\|)$



$O(\varepsilon N^3 \|A\|)$

s-step Lanczos

A different problem...

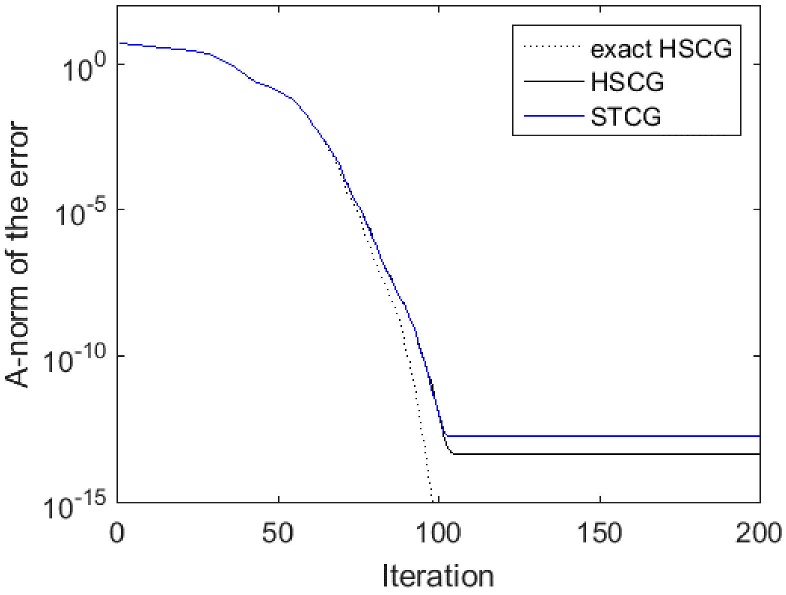
A: **nos4** from SuiteSparse

b: equal components in the eigenbasis of A and $\|b\| = 1$

$N = 100, \kappa(A) \approx 2\mathrm{e}3$

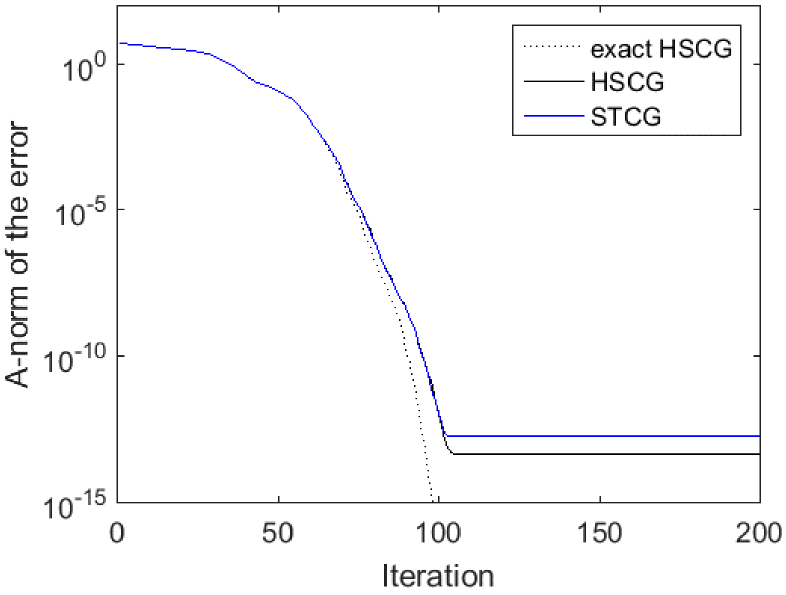
A different problem...

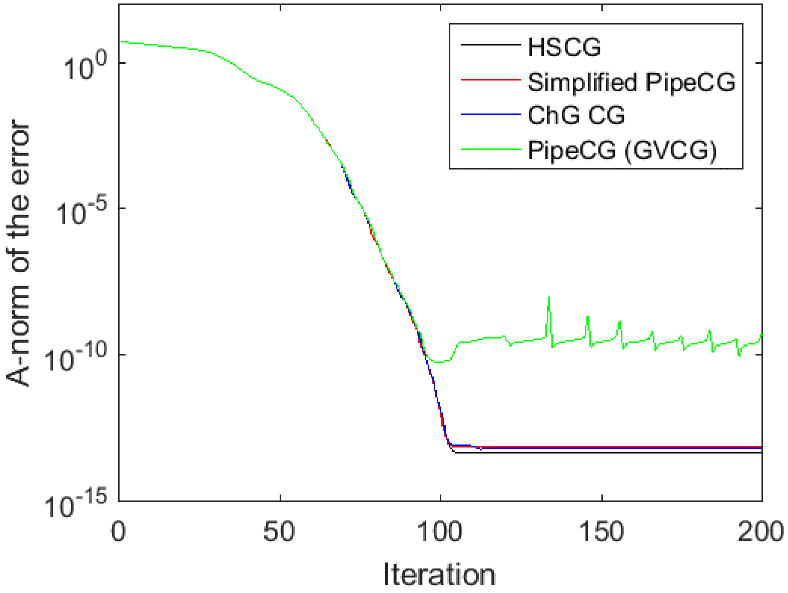
A: **nos4** from SuiteSparse
b: equal components in the eigenbasis
of A and $\|b\| = 1$
$N = 100, \kappa(A) \approx 2\text{e}3$



A different problem...

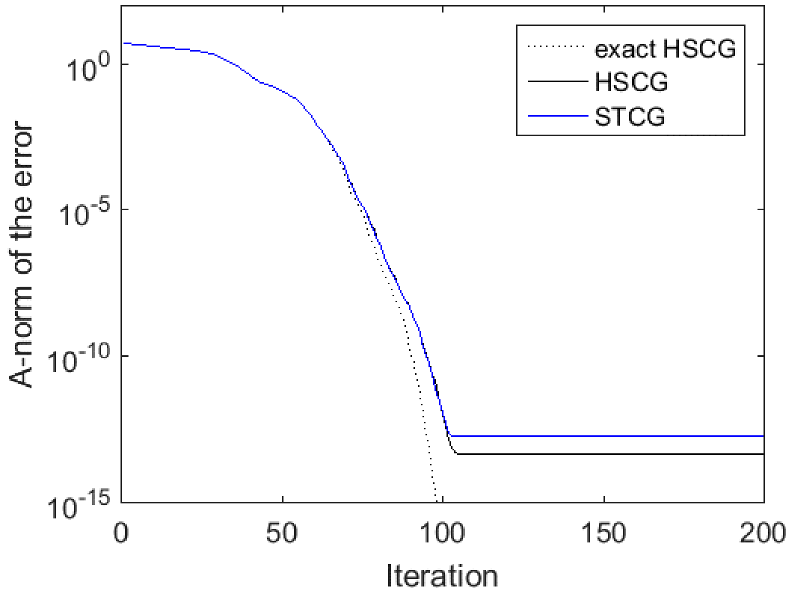
A: **nos4** from SuiteSparse
b: equal components in the eigenbasis
of A and $\|b\| = 1$
$N = 100, \kappa(A) \approx 2e3$

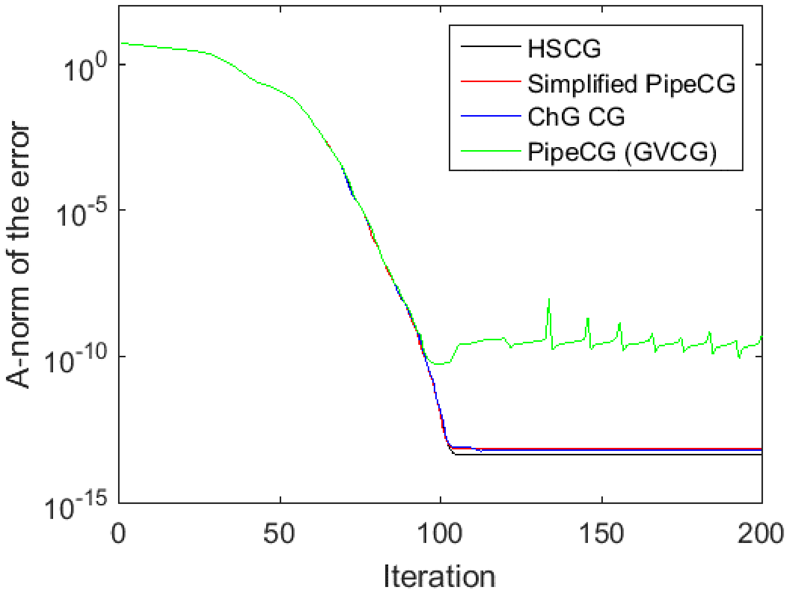


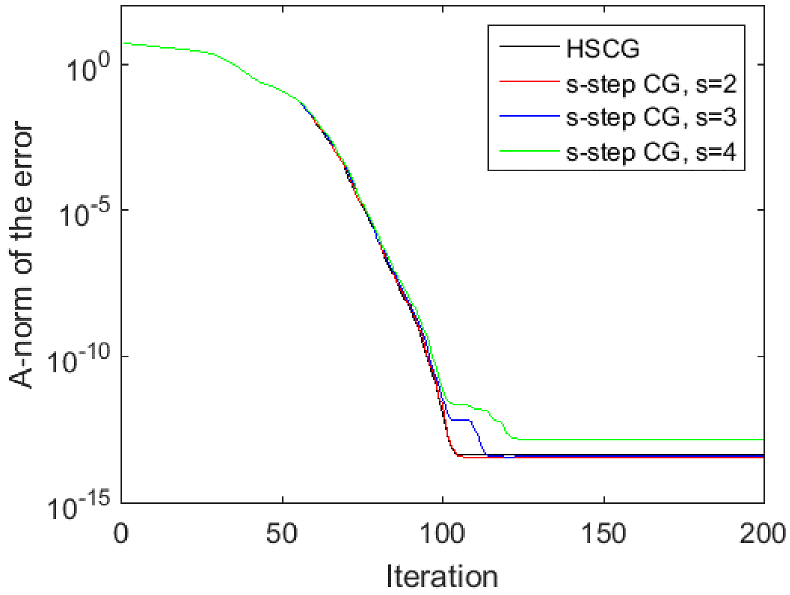


A different problem...

A: **nos4** from SuiteSparse
b: equal components in the eigenbasis
of A and $\|b\| = 1$
$N = 100, \kappa(A) \approx 2e3$







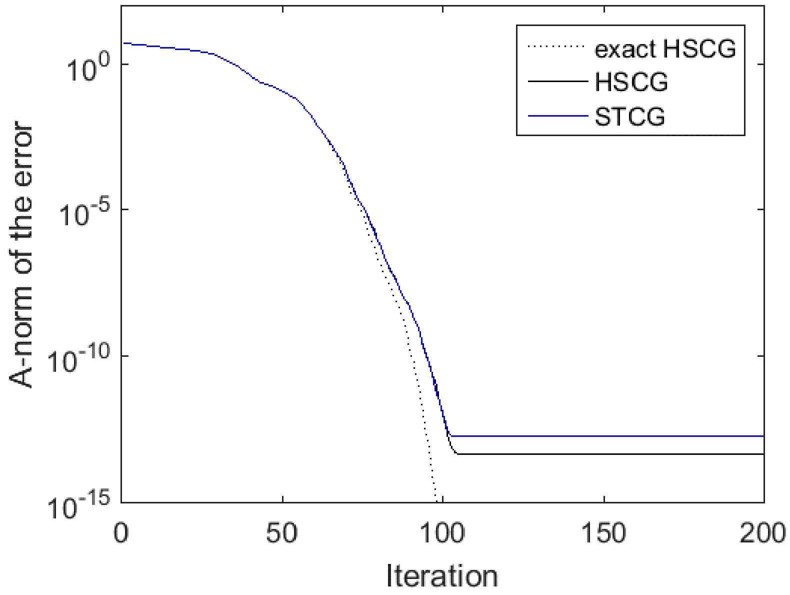
A different problem...

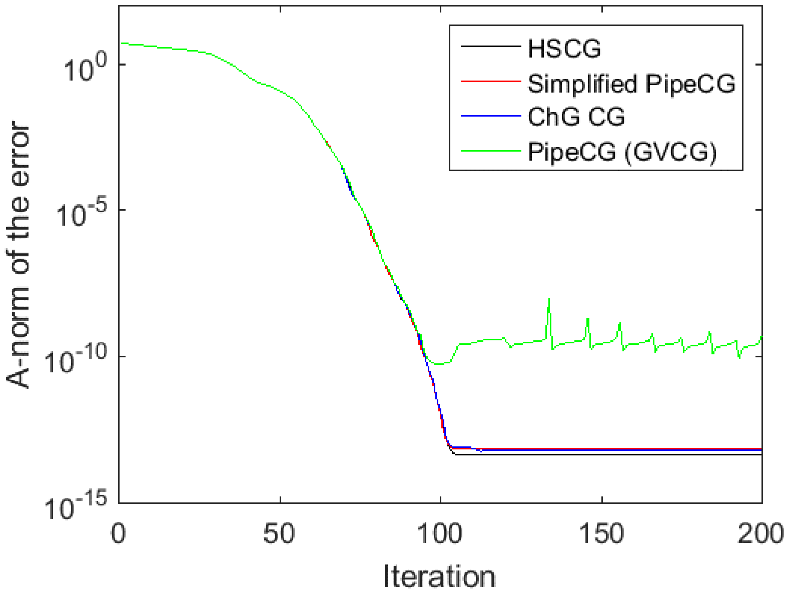
A: **nos4** from SuiteSparse

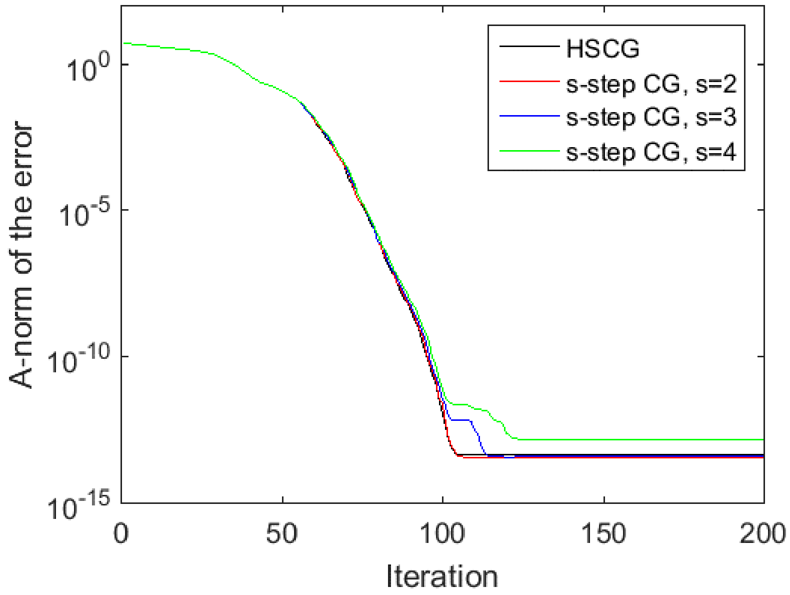
b: equal components in the eigenbasis of A and $\|b\| = 1$

$N = 100, \kappa(A) \approx 2\text{e}3$

If application only requires $\|x - x_i\|_A \leq 10^{-10}$, any of these methods will work!



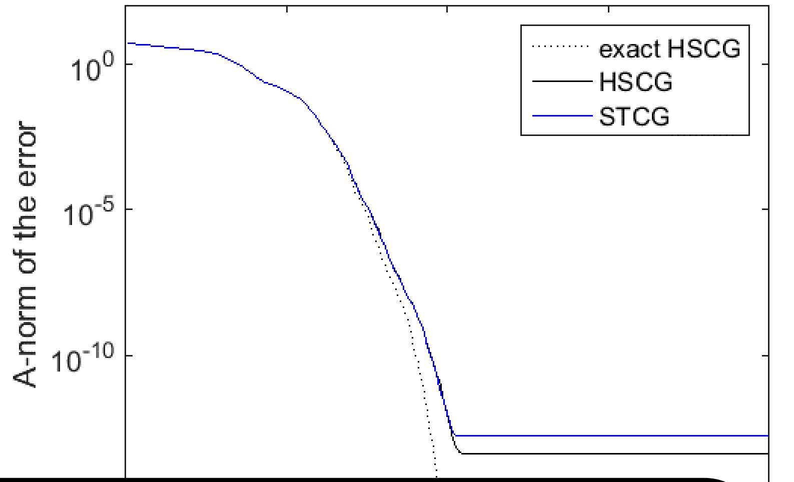




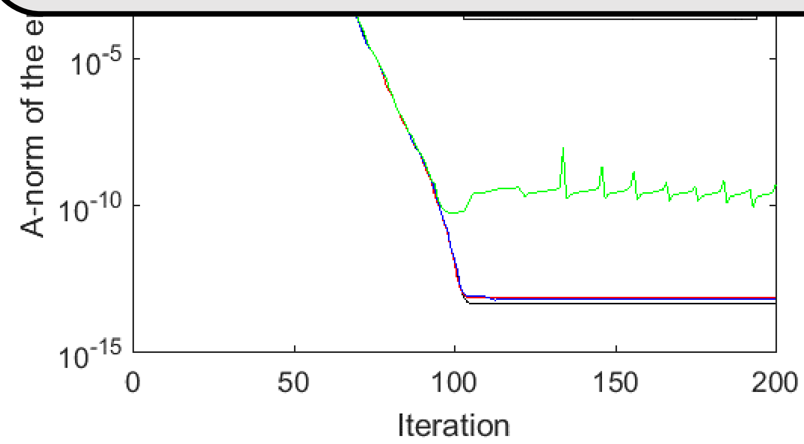
A different problem...

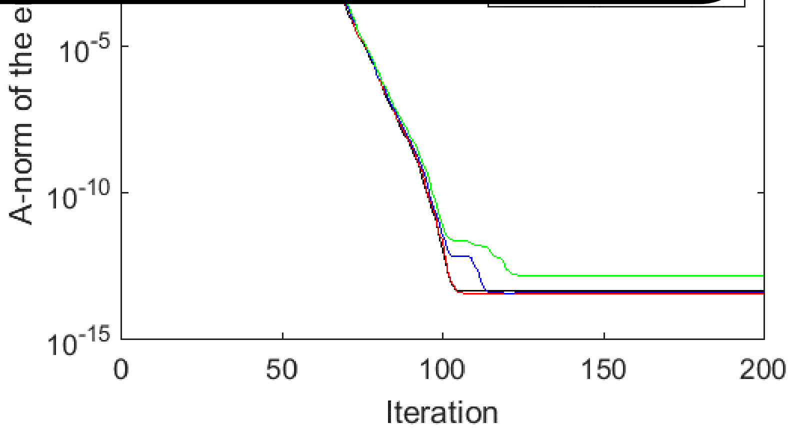
A: **nos4** from SuiteSparse
b: equal components in the eigenbasis
of A and $\|b\| = 1$
$N = 100, \kappa(A) \approx 2\text{e}3$

If application only requires
$\|x - x_i\|_A \leq 10^{-10}$,
any of these methods will work!



Need adaptive, problem-dependent approach based on understanding of finite precision behavior!





Adaptive s-step CG

- Consider the growth of the relative residual gap caused by errors in outer loop k, which begins with global iteration number m

Adaptive s-step CG

- Consider the growth of the relative residual gap caused by errors in outer loop k, which begins with global iteration number m
- We can approximate an upper bound on this quantity by

$$\frac{\|f_{m+s} - f_m\|}{\|A\| \|x\|} \lesssim \varepsilon \left(1 + \kappa(A) \Gamma_k \frac{\max_{j \in \{0,\ldots,s\}} \|\hat{r}_{m+j}\|}{\|A\| \|x\|} \right)$$

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

Adaptive s-step CG

- Consider the growth of the relative residual gap caused by errors in outer loop k, which begins with global iteration number m
- We can approximate an upper bound on this quantity by

$$\frac{\|f_{m+s} - f_m\|}{\|A\|\|x\|} \lesssim \varepsilon \left(1 + \kappa(A)\Gamma_k \frac{\max_{j\in\{0,\dots,s\}} \|\hat{r}_{m+j}\|}{\|A\|\|x\|} \right)$$

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

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

$$\Gamma_k \equiv c \cdot \|\hat{\mathcal{Y}}_k^+\| \, \||\hat{\mathcal{Y}}_k|\| \lesssim \frac{\varepsilon^*}{\varepsilon \max_{j\in\{0,\dots,s\}} \|\hat{r}_{m+j}\|}$$

Adaptive s-step CG

- Consider the growth of the relative residual gap caused by errors in outer loop k, which begins with global iteration number m
- We can approximate an upper bound on this quantity by

$$\frac{\|f_{m+s}-f_m\|}{\|A\|\|x\|} \lesssim \varepsilon\left(1+\kappa(A)\Gamma_k\frac{\max_{j\in\{0,\dots,s\}}\|\hat{r}_{m+j}\|}{\|A\|\|x\|}\right)$$

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

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

$$\Gamma_k \equiv c\cdot\|\hat{\mathcal{Y}}_k^+\|\,\||\hat{\mathcal{Y}}_k|\| \lesssim \frac{\varepsilon^*}{\varepsilon\max_{j\in\{0,\dots,s\}}\boxed{\|\hat{r}_{m+j}\|}}$$

- $\|\hat{r}_i\|$ large → Γ_k must be small; $\|\hat{r}_i\|$ small → Γ_k can grow

Adaptive s-step CG

- Consider the growth of the relative residual gap caused by errors in outer loop k, which begins with global iteration number m
- We can approximate an upper bound on this quantity by

$$\frac{\|f_{m+s} - f_m\|}{\|A\|\|x\|} \lesssim \varepsilon \left(1 + \kappa(A)\Gamma_k \frac{\max_{j\in\{0,\dots,s\}} \|\hat{r}_{m+j}\|}{\|A\|\|x\|} \right)$$

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

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

$$\Gamma_k \equiv c \cdot \|\hat{\mathcal{Y}}_k^+\| \, \||\hat{\mathcal{Y}}_k|\| \lesssim \frac{\varepsilon^*}{\varepsilon \max_{j\in\{0,\dots,s\}} \|\hat{r}_{m+j}\|}$$

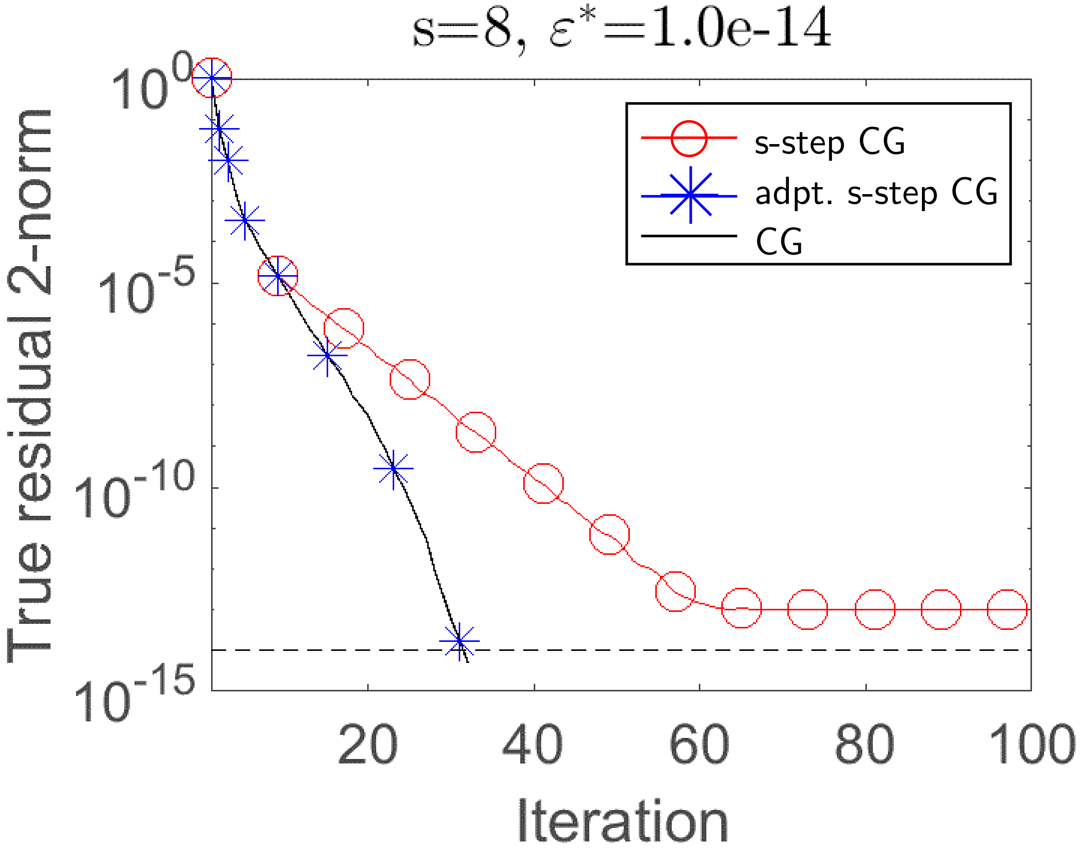
- $\|\hat{r}_i\|$ large → Γ_k must be small; $\|\hat{r}_i\|$ small → Γ_k can grow

$\Rightarrow$ adaptive s-step approach [C., 2018]

- s starts off small, increases at rate depending on $\|\hat{r}_i\|$ and ε^*

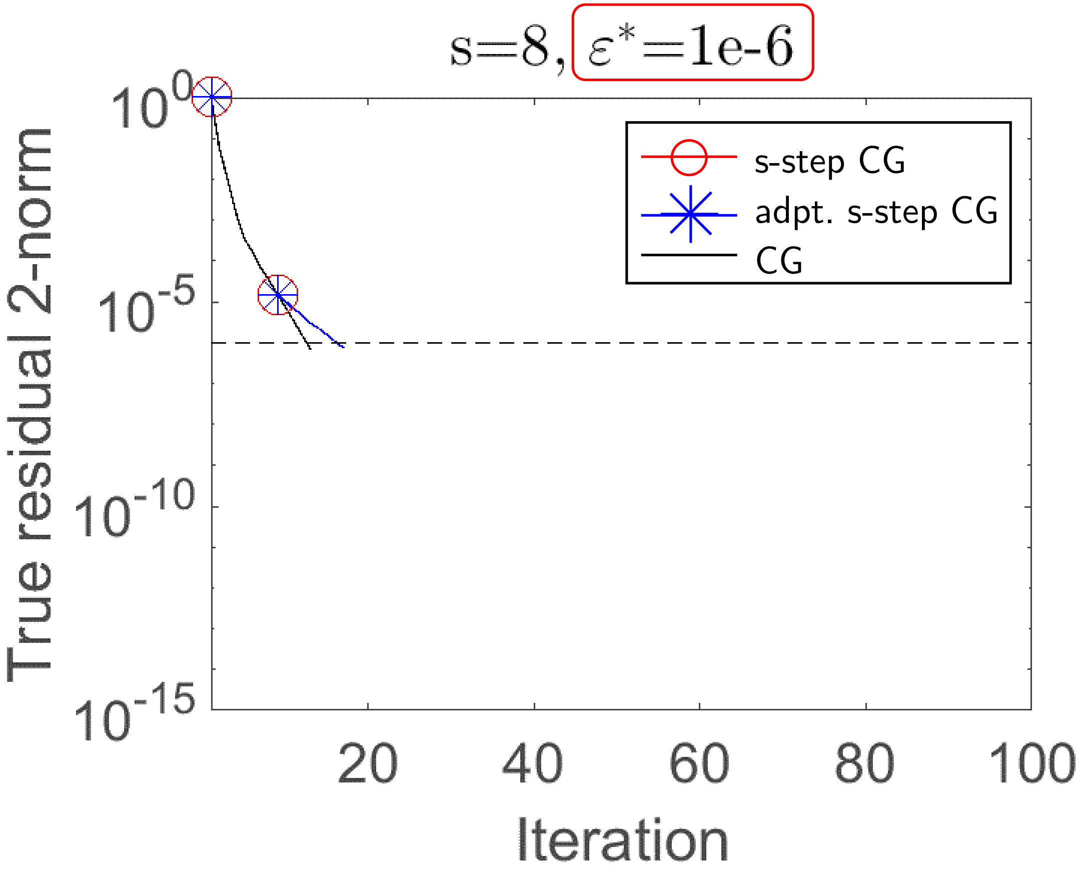
Adaptive s-step CG

mesh3e1 (UFSMC)
$n = 289$
$\kappa(A) \approx 10$
$b_i = 1/\sqrt{N}$



Adaptive s-step CG

mesh3e1 (UFSMC)
$n = 289$
$\kappa(A) \approx 10$
$b_i = 1/\sqrt{N}$



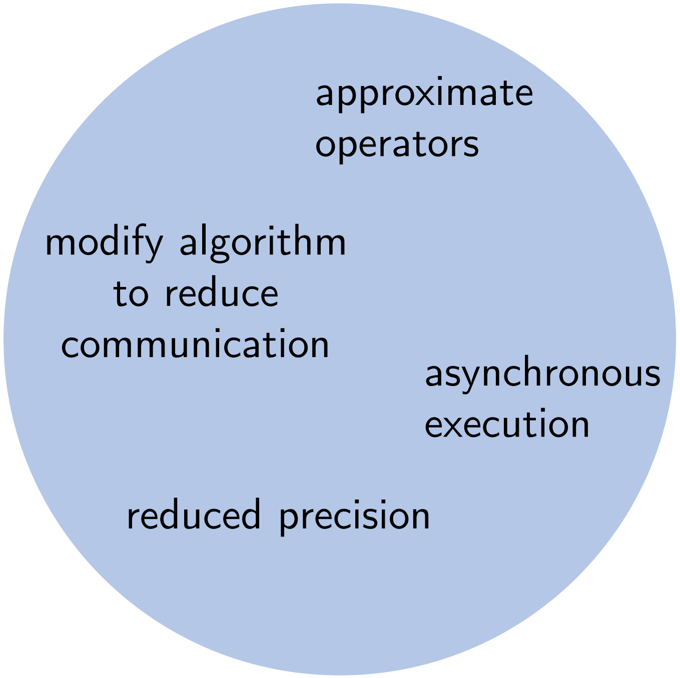
Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix} \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}$$

Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix} \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}$$

Reduce time per iteration



approximate operators

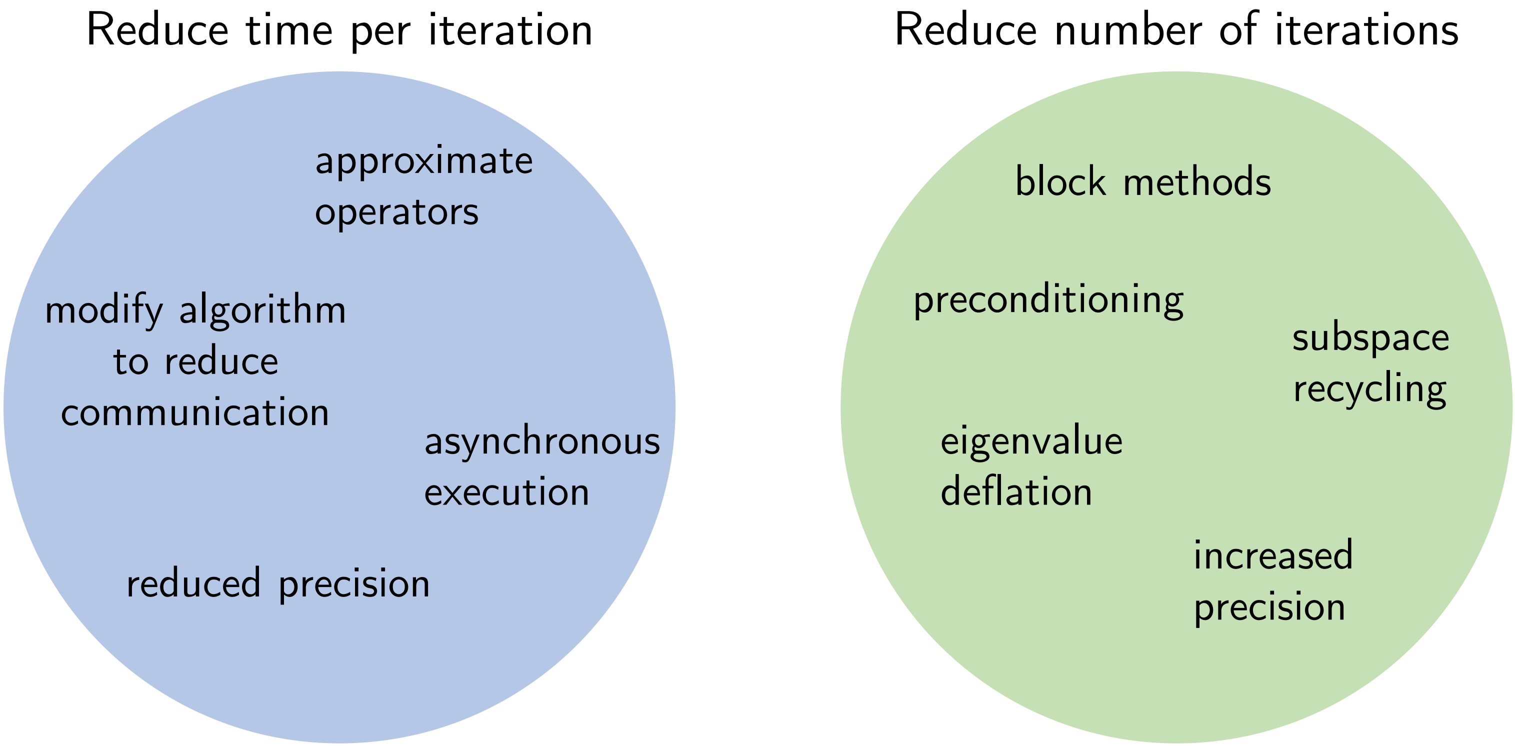
modify algorithm to reduce communication

asynchronous execution

reduced precision

Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix} \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}$$



Reduce time per iteration

approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

Reduce number of iterations

block methods

preconditioning

subspace recycling

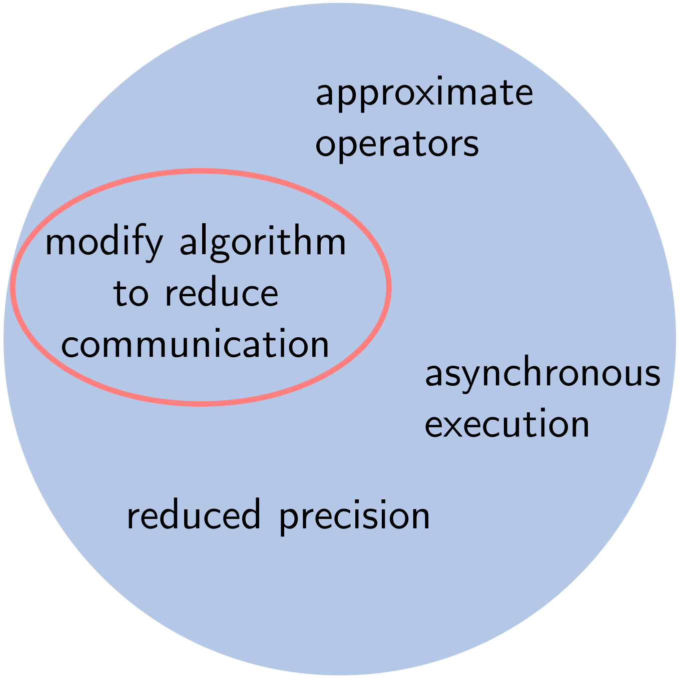
eigenvalue deflation

increased precision

Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix} \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}$$

Reduce time per iteration



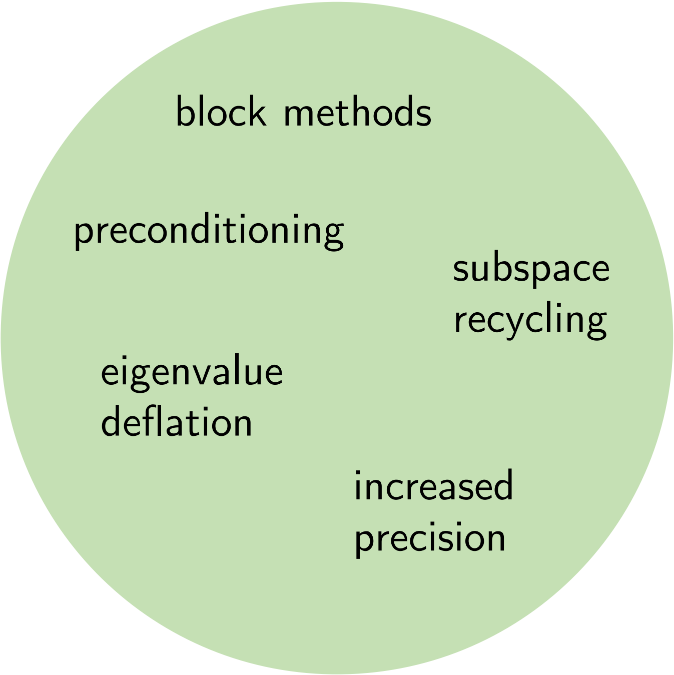
approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

Reduce number of iterations



block methods

preconditioning

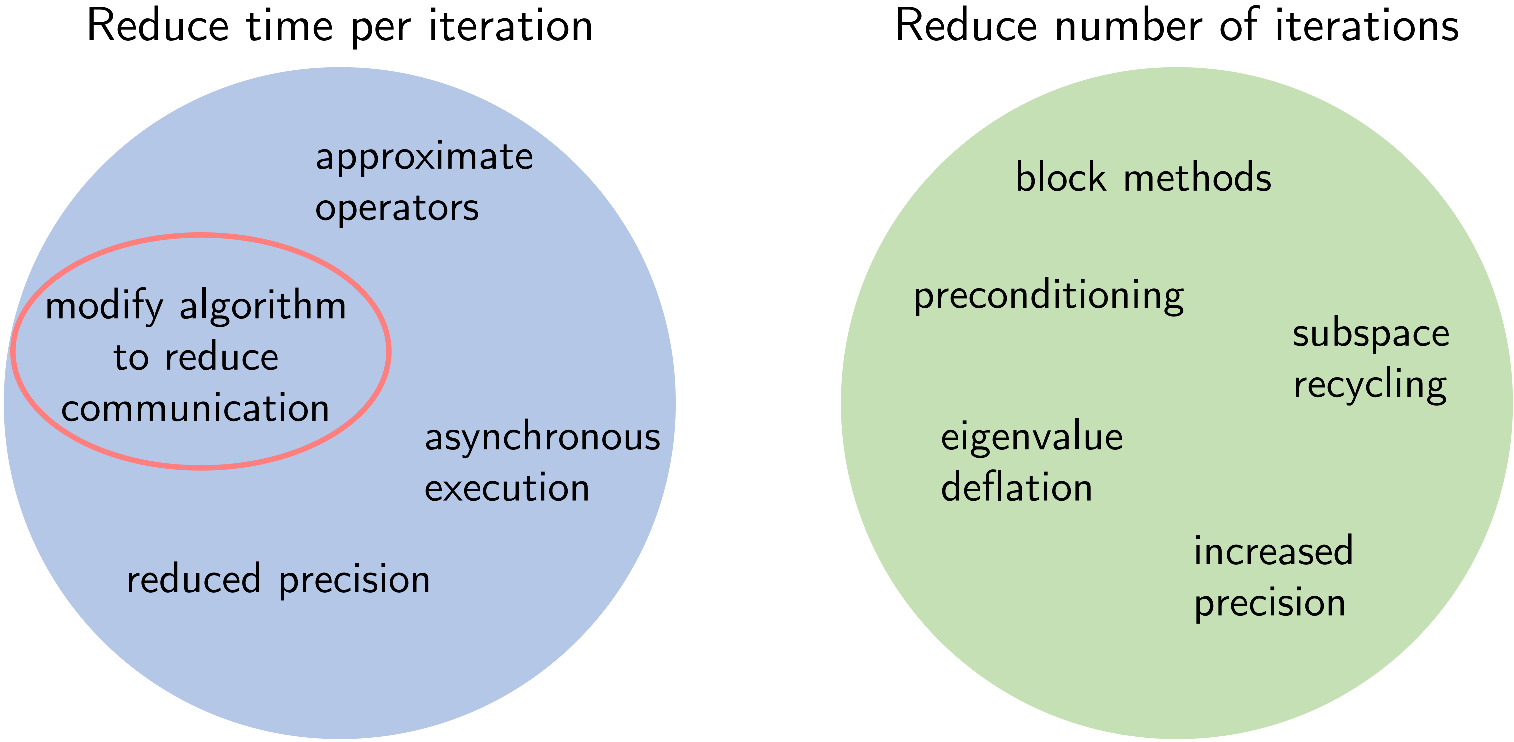
subspace recycling

eigenvalue deflation

increased precision

Takeaway

$$\text{runtime} = \left(\begin{matrix}\text{time per}\\ \text{iteration}\end{matrix}\right)\Downarrow \times \left(\begin{matrix}\text{number of iterations}\\ \text{until convergence}\end{matrix}\right)\Uparrow$$



Reduce time per iteration

approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

Reduce number of iterations

block methods

preconditioning

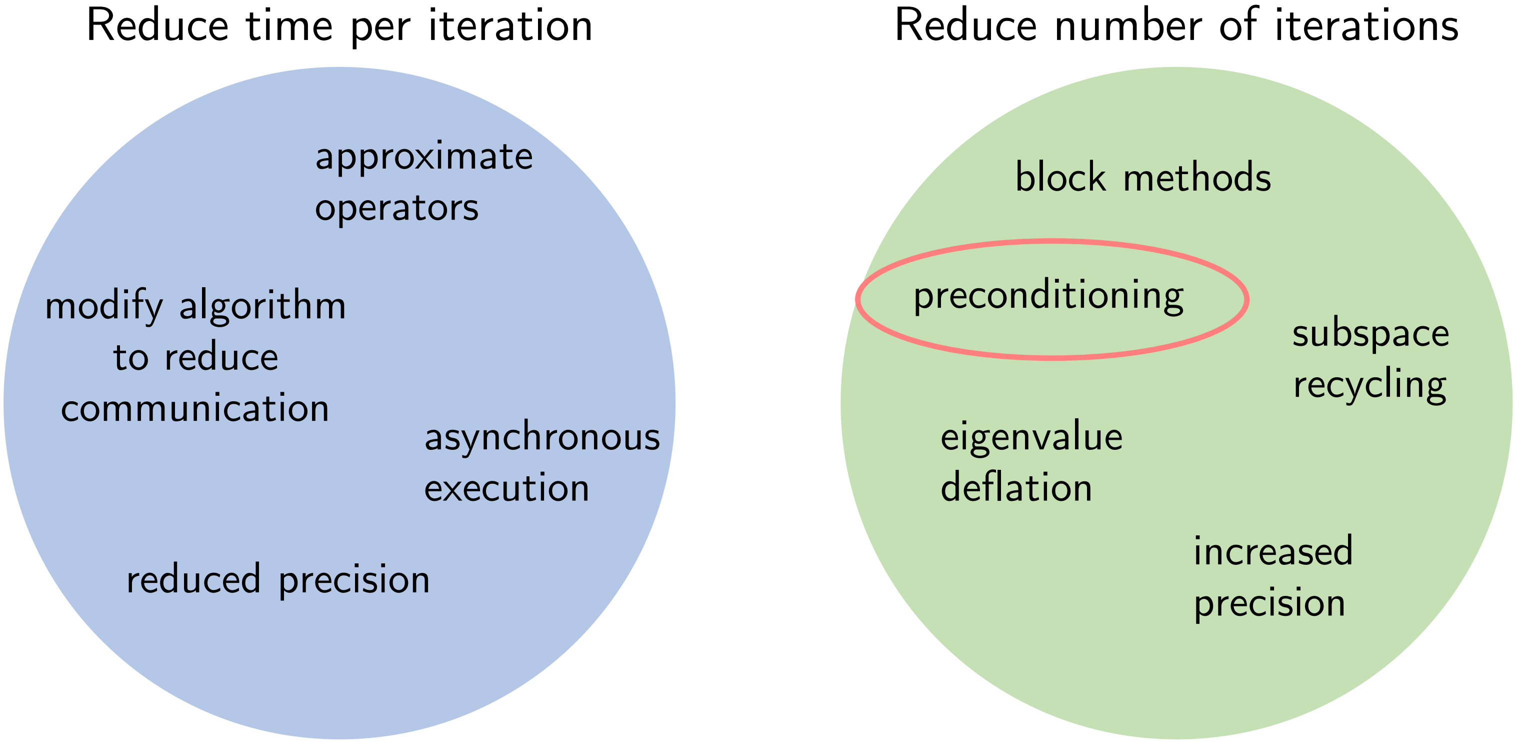
subspace recycling

eigenvalue deflation

increased precision

Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix} \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}$$



Reduce time per iteration

approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

Reduce number of iterations

block methods

preconditioning

subspace recycling

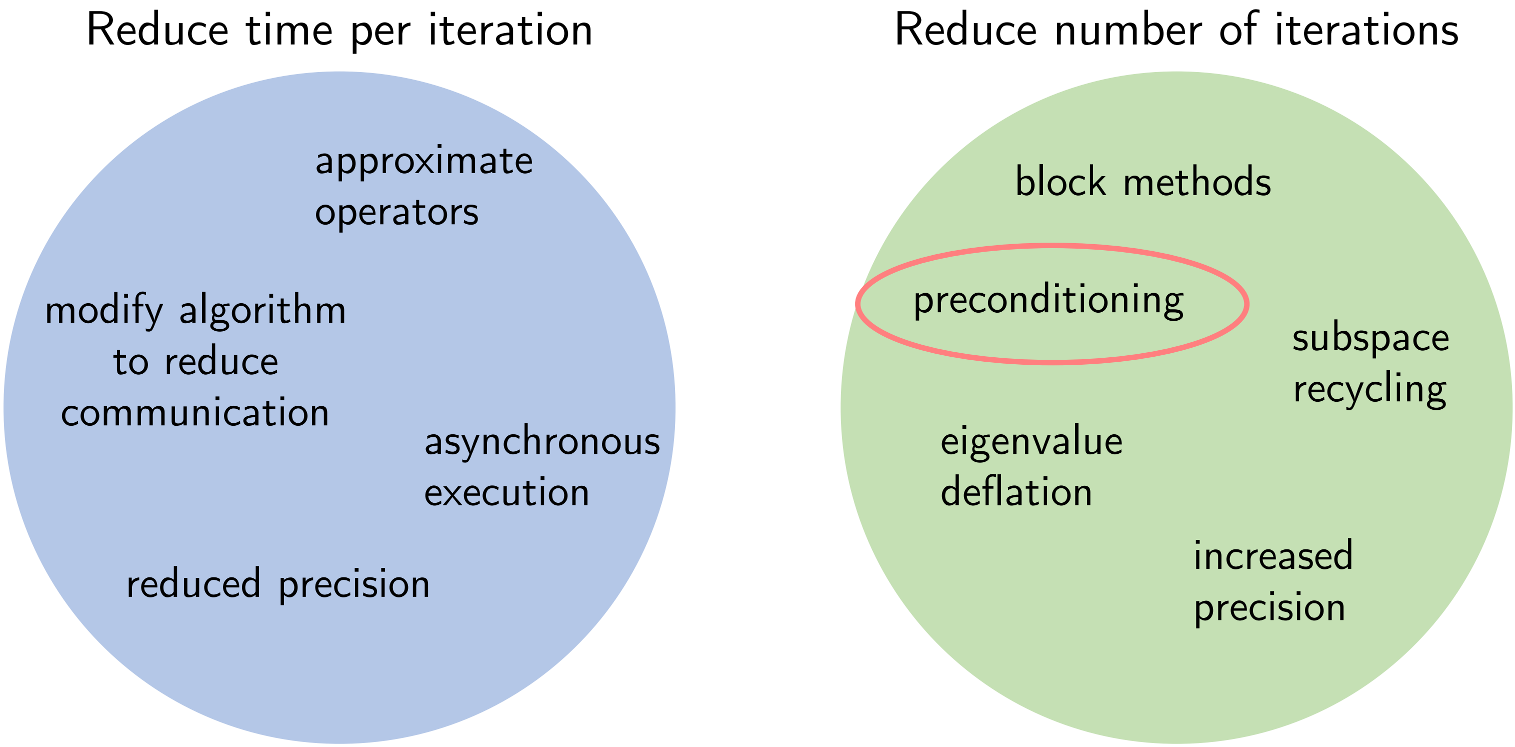
eigenvalue deflation

increased precision

$$\begin{aligned} Ax = b \quad \Rightarrow \quad M_L^{-1} A M_R^{-1} u &= M_L^{-1} b \\ x &= M_R^{-1} u \end{aligned}$$

Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix}\Uparrow \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}\Downarrow$$



Reduce time per iteration

approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

Reduce number of iterations

block methods

preconditioning

subspace recycling

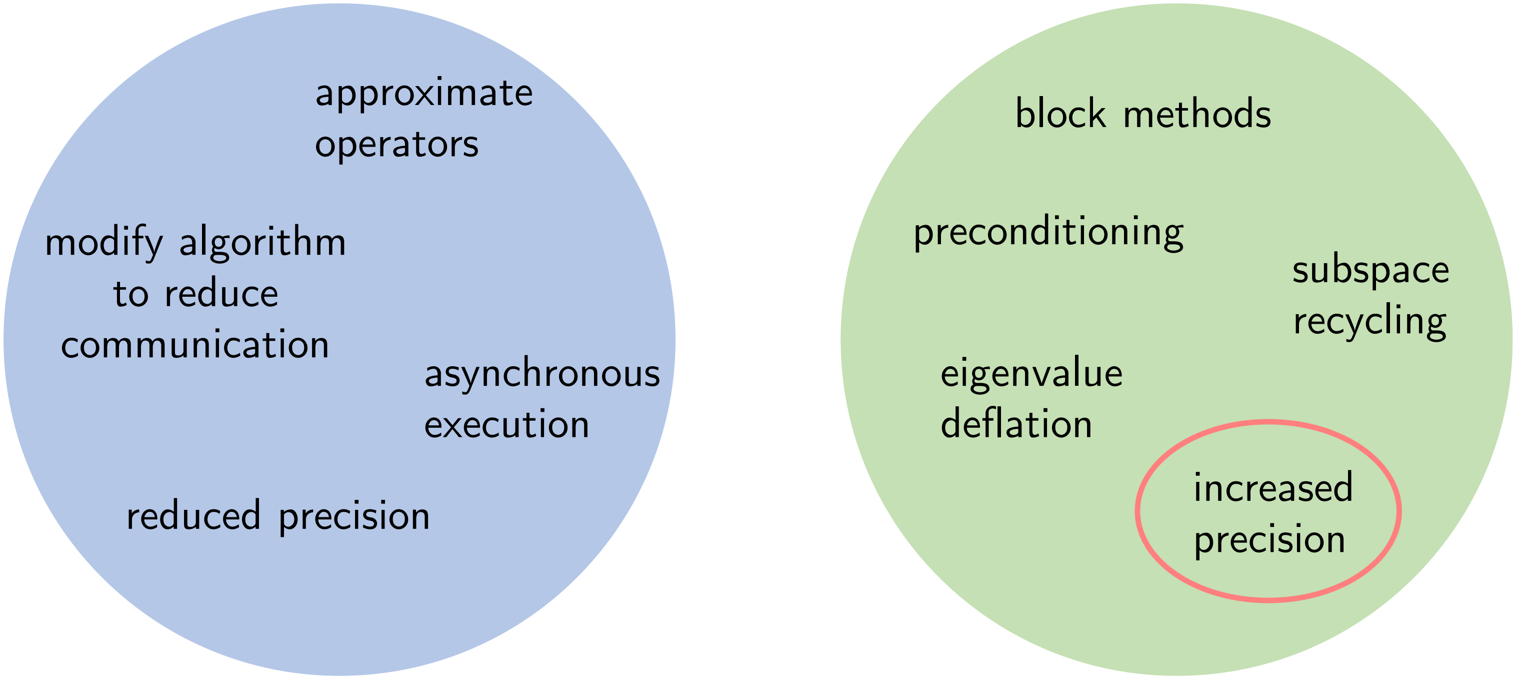
eigenvalue deflation

increased precision

$$\begin{aligned} Ax = b \quad \Rightarrow \quad M_L^{-1} A M_R^{-1} u &= M_L^{-1} b \\ x &= M_R^{-1} u \end{aligned}$$

Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix}\uparrow \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}\downarrow$$



Reduce time per iteration

approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

Reduce number of iterations

block methods

preconditioning

subspace recycling

eigenvalue deflation

increased precision

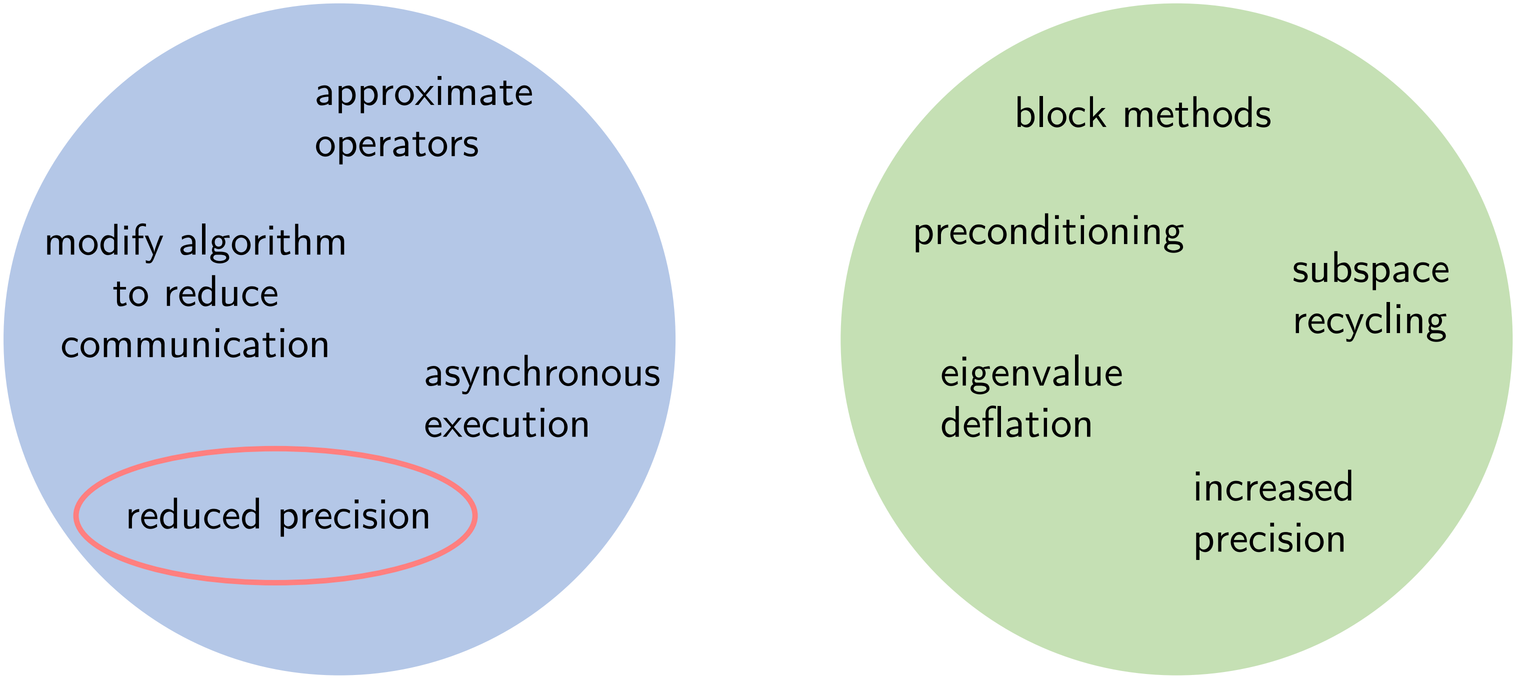
doubled precision → twice as many bits moved

Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix}\downarrow \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}\uparrow$$

Reduce time per iteration

Reduce number of iterations



approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

block methods

preconditioning

subspace recycling

eigenvalue deflation

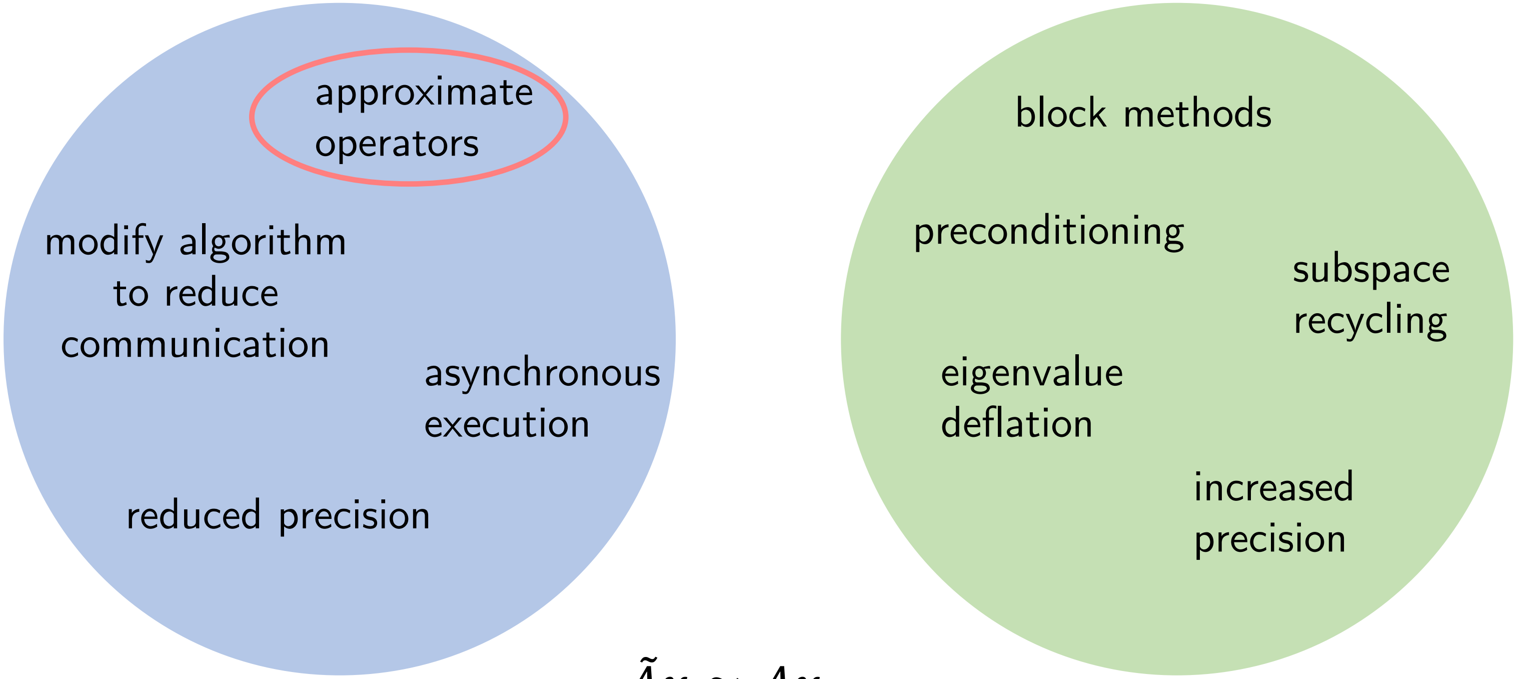
increased precision

Takeaway

$$\text{runtime} = \left(\begin{matrix}\text{time per}\\ \text{iteration}\end{matrix}\right)\downarrow \times \left(\begin{matrix}\text{number of iterations}\\ \text{until convergence}\end{matrix}\right)\uparrow$$

Reduce time per iteration

Reduce number of iterations



approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

block methods

preconditioning

subspace recycling

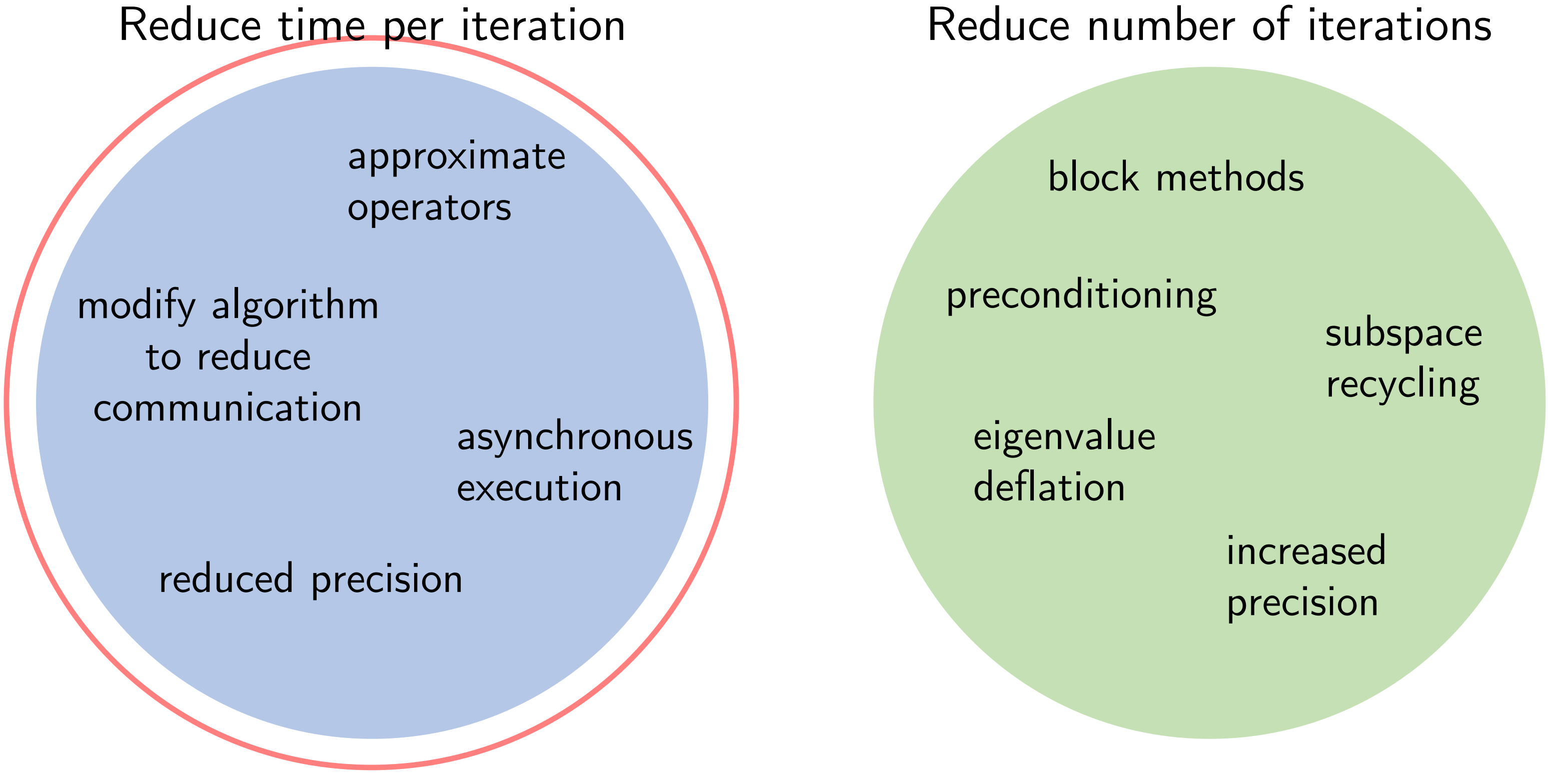
eigenvalue deflation

increased precision

$$\tilde{A}x \approx Ax$$

Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix} \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}$$



Reduce time per iteration

approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

Reduce number of iterations

block methods

preconditioning

subspace recycling

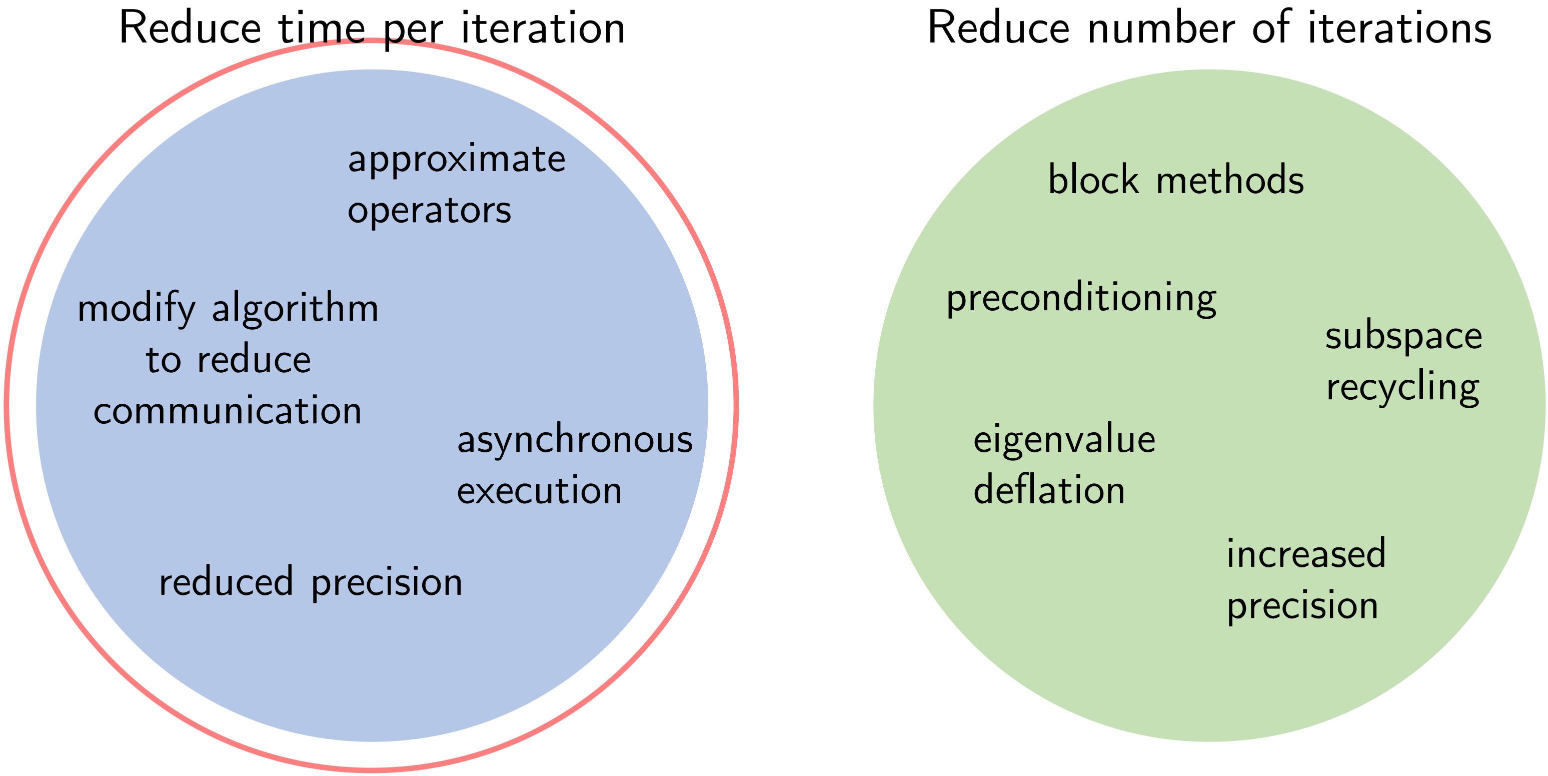
eigenvalue deflation

increased precision

convergence criteria never met: divergence, or convergence to inaccurate solution

Takeaway

$$\text{runtime} = \left(\begin{matrix}\text{time per}\\ \text{iteration}\end{matrix}\right) \times \overset{\infty}{\left(\begin{matrix}\text{number of iterations}\\ \text{until convergence}\end{matrix}\right)}$$



Reduce time per iteration

approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

Reduce number of iterations

block methods

preconditioning

subspace recycling

eigenvalue deflation

increased precision

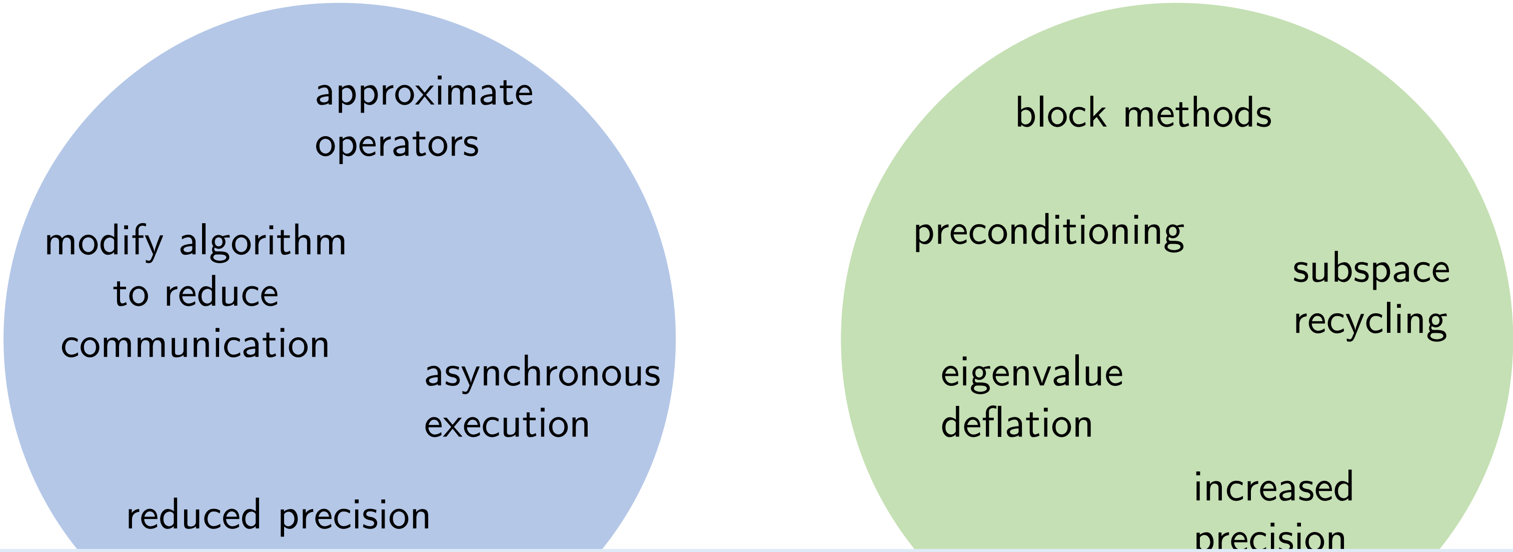
convergence criteria never met: divergence, or convergence to inaccurate solution

Takeaway

$$\text{runtime} = \begin{pmatrix}\text{time per}\\ \text{iteration}\end{pmatrix} \times \begin{pmatrix}\text{number of iterations}\\ \text{until convergence}\end{pmatrix}$$

Reduce time per iteration

Reduce number of iterations



approximate operators

modify algorithm to reduce communication

asynchronous execution

reduced precision

block methods

preconditioning

subspace recycling

eigenvalue deflation

increased precision

To minimize runtime, must understand how modifications affect:

1) attainable accuracy 2) convergence rate 3) time per iteration

Future Work: Finite Precision Krylov Subspace Methods

- Convergence delay in high-performance CG variants
  - Extending results of Greenbaum [1989] to s-step and pipelined versions

Future Work: Finite Precision Krylov Subspace Methods

- Convergence delay in high-performance CG variants
  - Extending results of Greenbaum [1989] to s-step and pipelined versions

- Deviation from exact Krylov subspaces in Lanczos
  - Can the space spanned by the computed $\hat{V}_i$ be related to some exactly Krylov subspace?

Future Work: Finite Precision Krylov Subspace Methods

- Convergence delay in high-performance CG variants
  - Extending results of Greenbaum [1989] to s-step and pipelined versions

- Deviation from exact Krylov subspaces in Lanczos
  - Can the space spanned by the computed $\hat{V}_i$ be related to some exactly Krylov subspace?

- Loss of orthogonality vs. backward error in finite precision GMRES

$$\frac{\|\hat{r}_i\|}{\|b\|+\|A\|\|\hat{x}_i\|} \cdot \left\|I - \hat{V}_i^T \hat{V}_i\right\| \approx O(\varepsilon) \text{ ?}$$

Future Work: Finite Precision Krylov Subspace Methods

- Convergence delay in high-performance CG variants
  - Extending results of Greenbaum [1989] to s-step and pipelined versions

- Deviation from exact Krylov subspaces in Lanczos
  - Can the space spanned by the computed $\hat{V}_i$ be related to some exactly Krylov subspace?

- Loss of orthogonality vs. backward error in finite precision GMRES

$$\frac{\|\hat{r}_i\|}{\|b\|+\|A\|\|\hat{x}_i\|} \cdot \left\|I - \hat{V}_i^T \hat{V}_i\right\| \approx O(\varepsilon) \text{ ?}$$

- Rigorous analysis of accuracy and convergence for various commonly-used techniques
  - Deflation, incomplete preconditioning, matrix equilibration, look-ahead, etc.

Simulation + Data + Learning

- Data analytics and machine learning increasingly important in scientific discovery
  - Event identification, correlation in high-energy physics
  - Climate simulation validation using sensor data
  - Determine patterns and trends from astronomical data
  - Genetic sequencing

Empirical
Theoretical
Computational
Data-driven

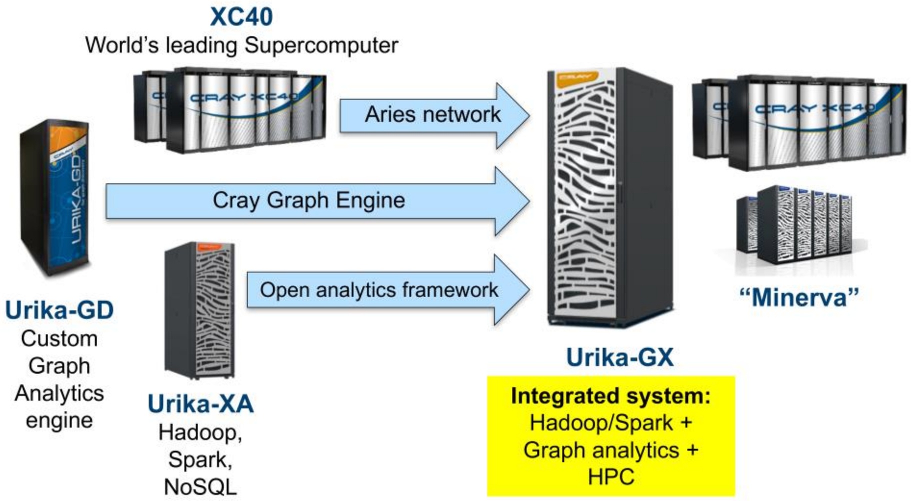
- **The convergence of simulation, data, and learning**
  - current hot topic: workshops, conferences, research initiatives, funding calls

Simulation + Data + Learning

- Data analytics and machine learning increasingly important in scientific discovery
  - Event identification, correlation in high-energy physics
  - Climate simulation validation using sensor data
  - Determine patterns and trends from astronomical data
  - Genetic sequencing

- **The convergence of simulation, data, and learning**
  - current hot topic: workshops, conferences, research initiatives, funding calls

- Driving changes in supercomputer architecture
  - Multiprecision hardware
  - Specialized accelerators
  - Memory at node


XC40
World's leading Supercomputer
Aries network
Cray Graph Engine
Open analytics framework
Urika-GD
Custom Graph Analytics engine
Urika-XA
Hadoop, Spark, NoSQL
Urika-GX
Integrated system: Hadoop/Spark + Graph analytics + HPC
"Minerva"

Numerical Linear Algebra for Data Analytics + ML

- Numerical linear algebra routines are the core computational kernels in many data science and machine learning applications

Numerical Linear Algebra for Data Analytics + ML

- Numerical linear algebra routines are the core computational kernels in many data science and machine learning applications
    - Growing problem sizes, growing datasets → need scalable performance

Numerical Linear Algebra for Data Analytics + ML

- Numerical linear algebra routines are the core computational kernels in many data science and machine learning applications
  - Growing problem sizes, growing datasets → need scalable performance

Challenges:

- Optimizing performance in different space: different/new architectures, matrix structures, accuracy requirements, etc.
- Translation between

$$(\%\text{ accuracy on test dataset}) \leftrightarrow (\text{number of FP digits})$$

- Designing efficient and effective preconditioners
- More general error analyses: How do approximations (e.g., sparsification, low-rank representation) affect convergence and accuracy of numerical algorithms?

Thank you!

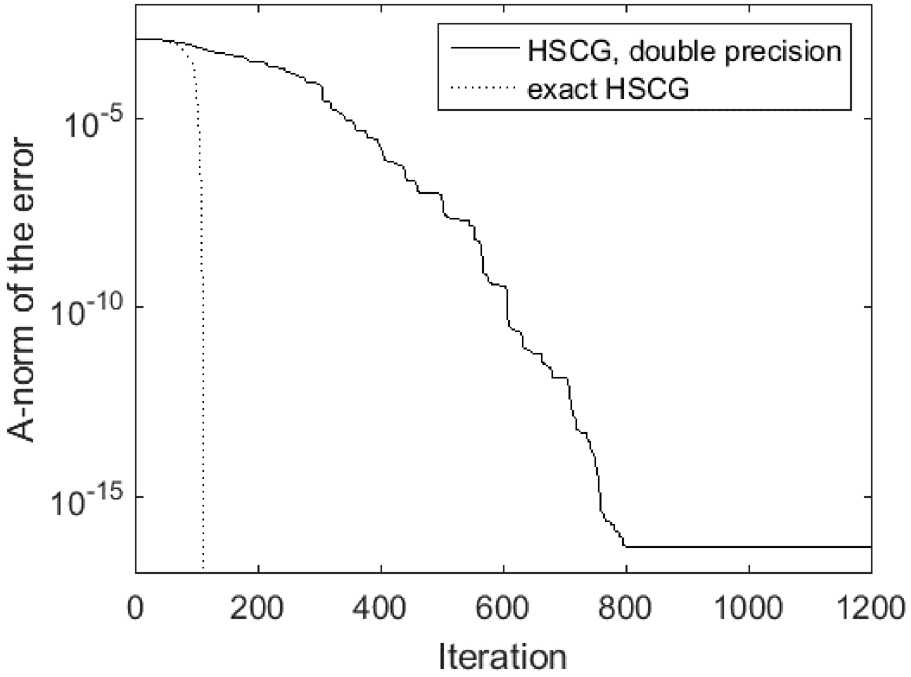
carson@karlin.mff.cuni.cz
www.karlin.mff.cuni.cz/~carson

The effects of finite precision

Errors have two effects:

1. Delay of convergence
   - No longer have exact Krylov subspace
   - Can lose numerical rank deficiency
   - Residuals no longer orthogonal
     - Minimization no longer exact!

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



A: bcsstk03 from UFSMC, b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

Many existing results for CG; See Meurant and Strakoš (2006) for a thorough summary of early developments in finite precision analysis of Lanczos and CG

Attainable accuracy of pipelined CG

- Both ChG CG and GVCG use the same update formulas for x_i and r_i:

$$x_i = x_{i-1} + \alpha_{i-1} p_{i-1}, \qquad r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$$

- In finite precision:

$$\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} + \boldsymbol{\delta x_i} \qquad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}\,\hat{s}_{i-1} + \boldsymbol{\delta r_i}$$

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

where

$$G_i = \hat{S}_i - A\hat{P}_i, \quad d_i = [\hat{\alpha}_0, \dots, \hat{\alpha}_{i-1}]^T$$

- Bound on $\|G_i\|$ will differ depending on the method (other recurrences or auxiliary vectors used)

Preconditioning for s-step KSMs

- Much recent/ongoing work in developing communication-avoiding preconditioned methods

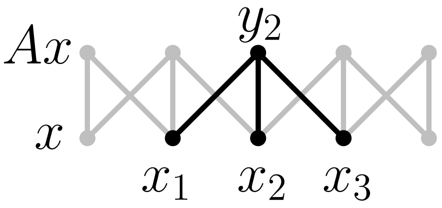
- Many approaches shown to be compatible
  - Diagonal
  - Sparse Approx. Inverse (SAI) – for s-step BICGSTAB by Mehri (2014)
  - HSS preconditioning (Hoemmen, 2010); for banded matrices (Knight, C., Demmel, 2014); same general technique for any system that can be written as sparse + low-rank
  - CA-ILU(0) – Moufawad and Grigori (2013)
  - Deflation for s-step CG (C., Knight, Demmel, 2014), for s-step GMRES (Yamazaki et al., 2014)
  - Domain decomposition – avoid introducing additional communication by "underlapping" subdomains (Yamazaki et al., 2014)

SpMV Dependency Graph

$G = (V, E)$ where $V = \{y_0, \dots, y_{n-1}\} \cup \{x_0, \dots, x_{n-1}\}$ and $\left(y_i, x_j\right) \in E$ if $A_{ij} \neq 0$

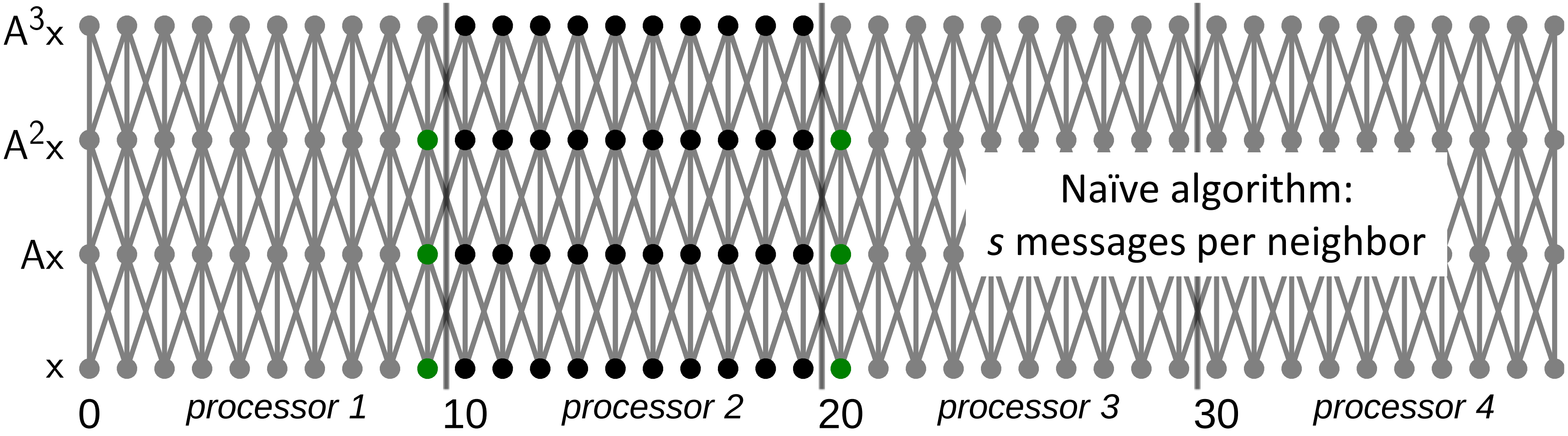
Example: Tridiagonal matrix

$$\underset{y}{\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}} = \underset{A}{\begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}} \underset{x}{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}} \Rightarrow$$

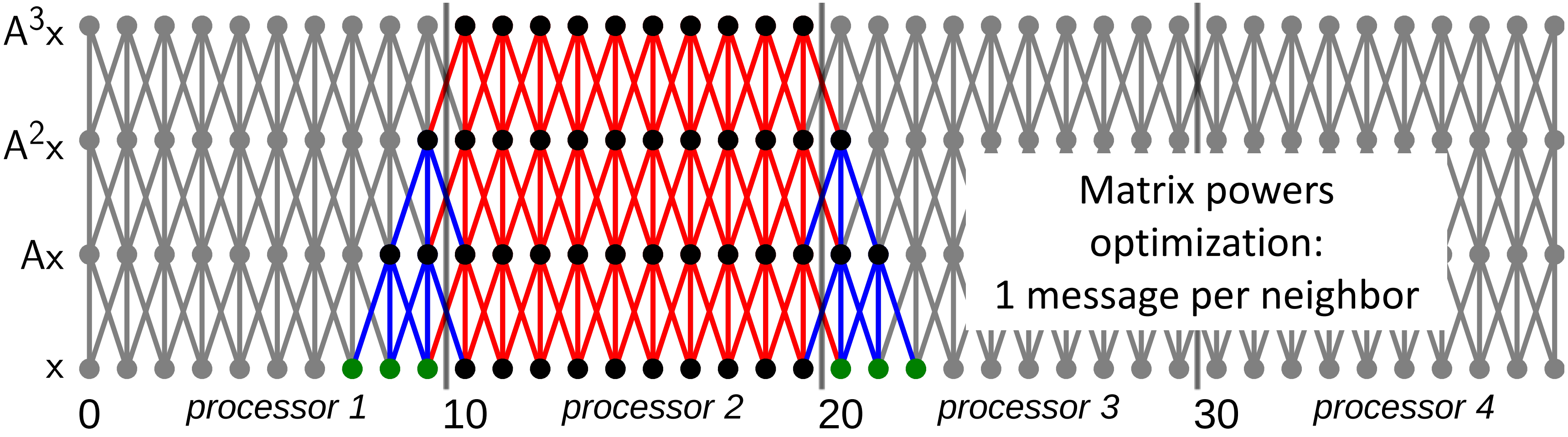


Parallel Matrix Powers

Example: tridiagonal matrix, *s* = 3, *n* = 40, *p* = 4



Naïve algorithm:
s messages per neighbor

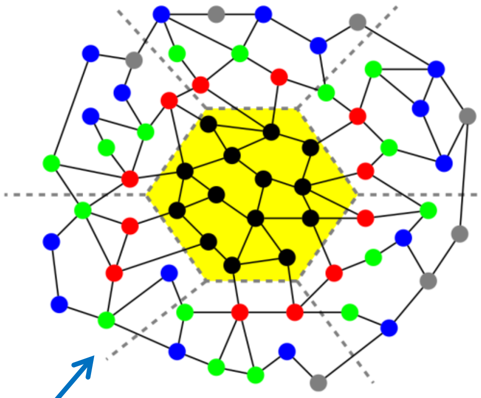


Matrix powers
optimization:
1 message per neighbor

The Matrix Powers Kernel (Demmel et al., 2007)

Avoids communication:

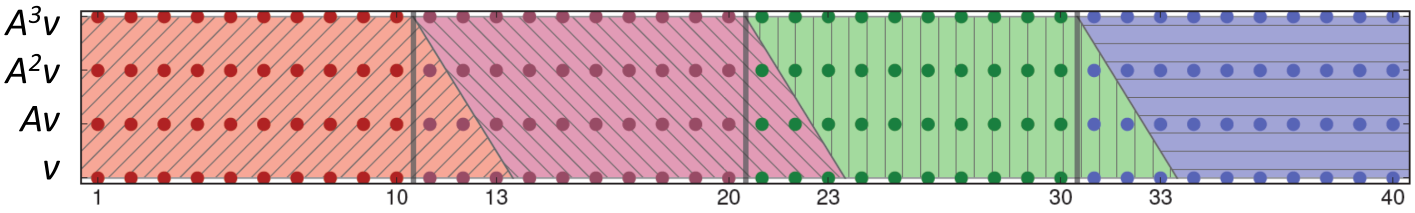
- In serial, by exploiting temporal locality:
  - Reading A, reading vectors
- In parallel, by doing only 1 ‘expand’ phase (instead of s).
- Requires sufficiently low ‘surface-to-volume’ ratio



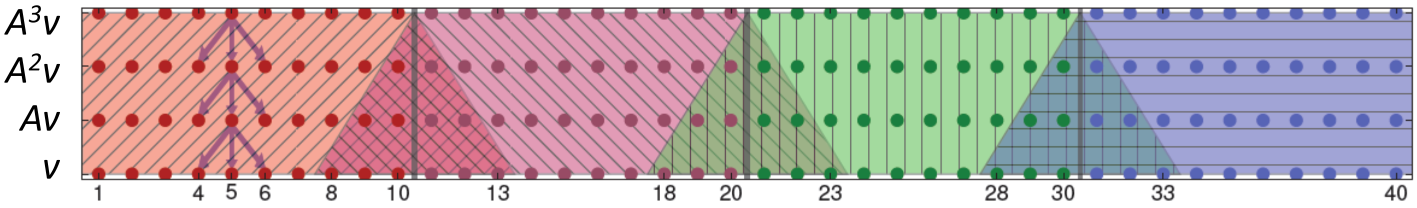
Also works for general graphs!

black = local elements
red = 1-level dependencies
green = 2-level dependencies
blue = 3-level dependencies

Tridiagonal Example:



Sequential



Parallel

Complexity comparison

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

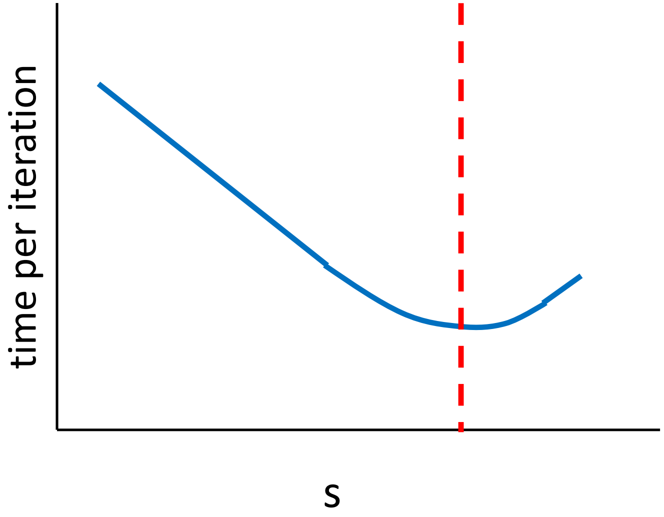
(Assuming each of p processors owns n/p rows of the matrix and $s \leq \sqrt{n/p}$)

	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	$\frac{sn}{p}$	$\frac{s^2n}{p}$	$s\sqrt{n/p}$	$s^2\log_2 p$	1	$\log_2 p$

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

Choosing the Block Size s

- Parameter s is limited by machine parameters, matrix sparsity structure, and machine properties
  - As we increase s, at some point the lower-order terms in flops and words moved will dominate runtime
  - This point depends on relative costs of, e.g., a flop versus sending a message on the machine



- 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 ...

Choosing a Polynomial Basis

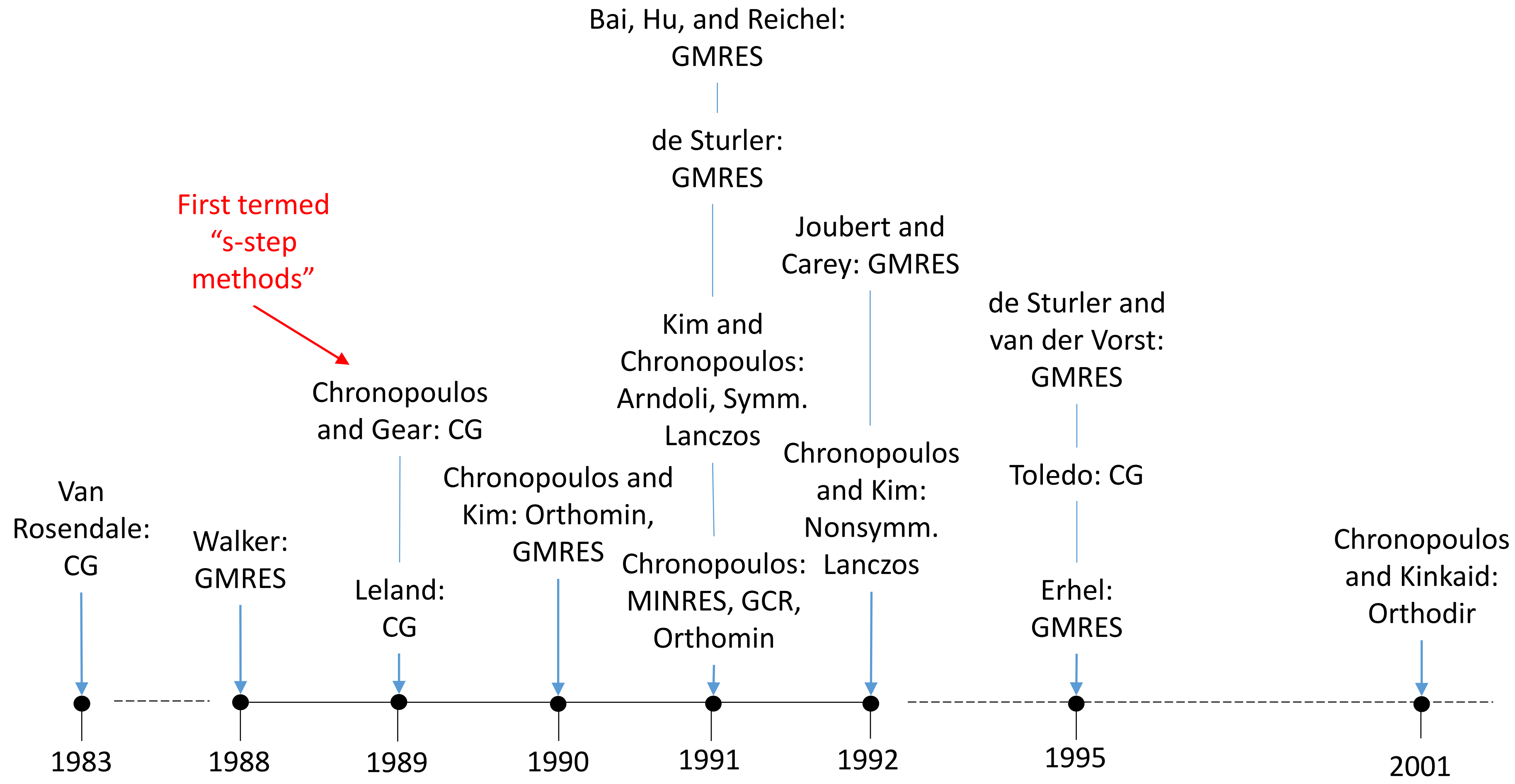
- Recall: in each outer loop of CA-CG, we compute bases for some Krylov subspaces, $\mathcal{K}_m(A, v) = \text{span}\{v, Av, \dots, A^{m-1}v\}$

- Simple loop unrolling gives monomial basis $Y = [p, Ap, A^2p, A^3p, \dots]$
  - Condition number can grow exponentially with s
    - Condition number = ratio of largest to smallest eigenvalues, $\lambda_{\max}/\lambda_{\min}$
  - Recognized early on that this negatively affects convergence (Leland, 1989)

- **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**

History of *s*-step Krylov Methods



Van Rosendale: CG

1983

Walker: GMRES

1988

First termed "s-step 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

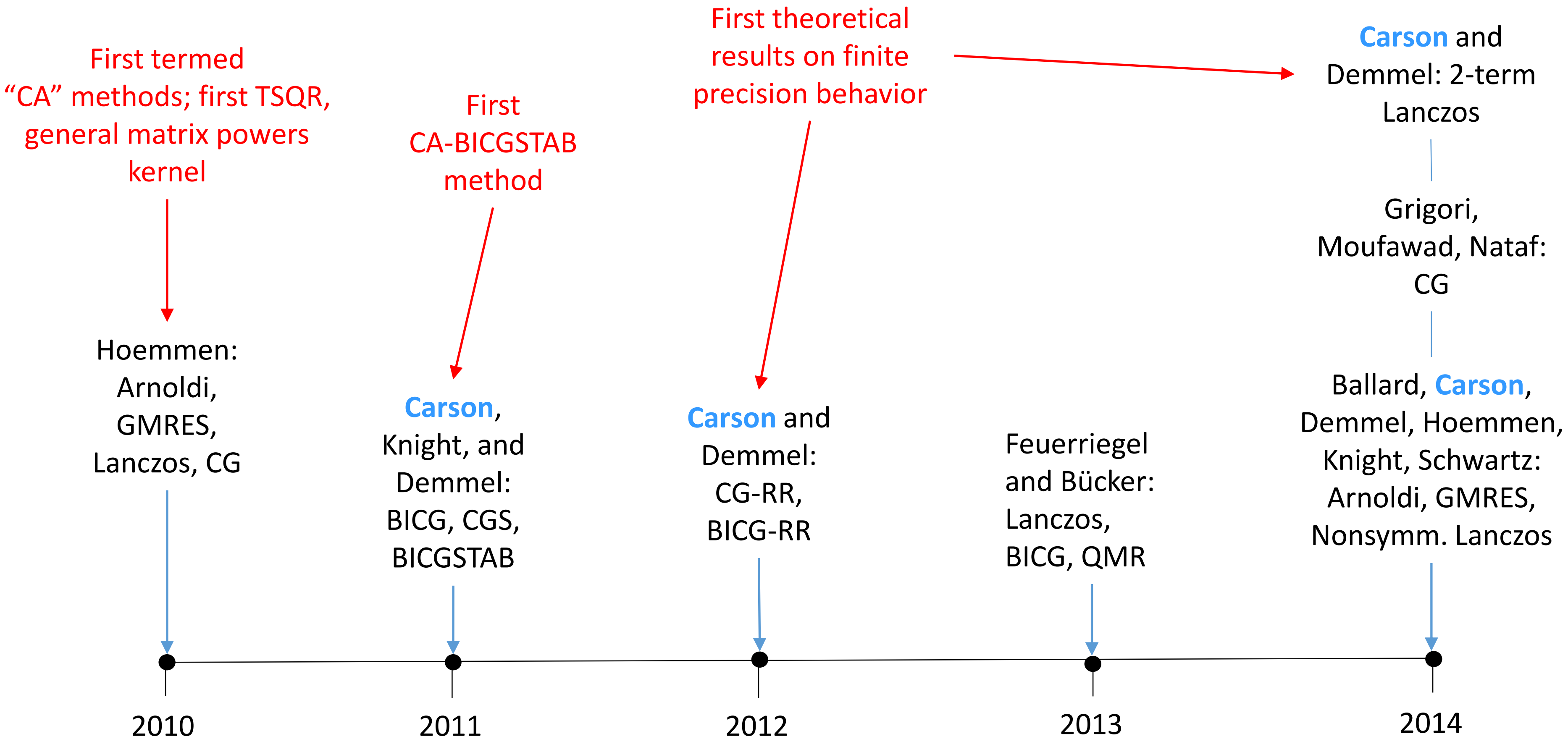
Erhel: GMRES

1995

Chronopoulos and Kinkaid: Orthodir

2001

Recent Years…



First termed "CA" methods; first TSQR, general matrix powers kernel

Hoemmen: Arnoldi, GMRES, Lanczos, CG

2010

First CA-BICGSTAB method

Carson, Knight, and Demmel: BICG, CGS, BICGSTAB

2011

First theoretical results on finite precision behavior

Carson and Demmel: CG-RR, BICG-RR

2012

Feuerriegel and Bücker: Lanczos, BICG, QMR

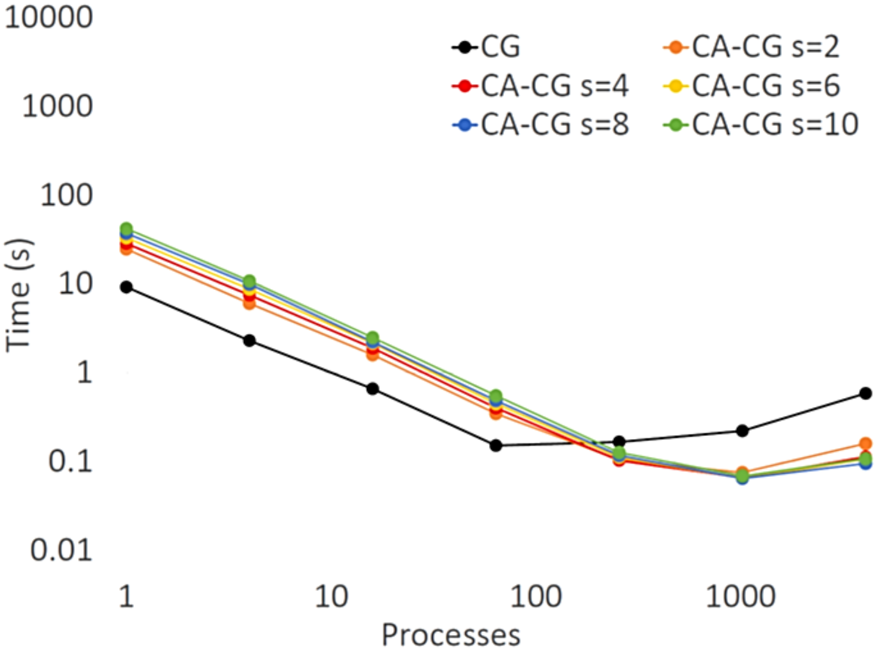
2013

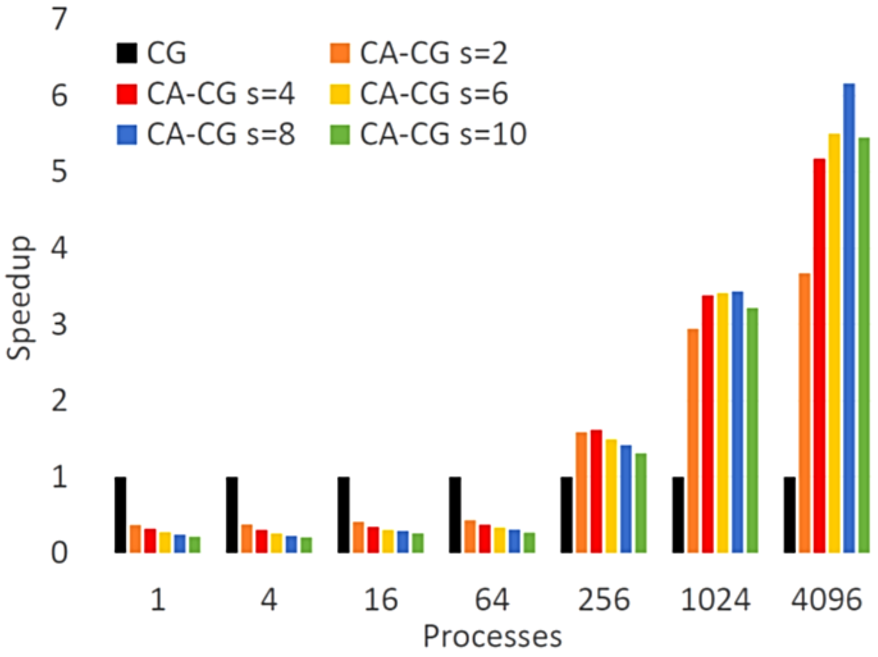
Carson and Demmel: 2-term Lanczos

Grigori, Moufawad, Nataf: CG

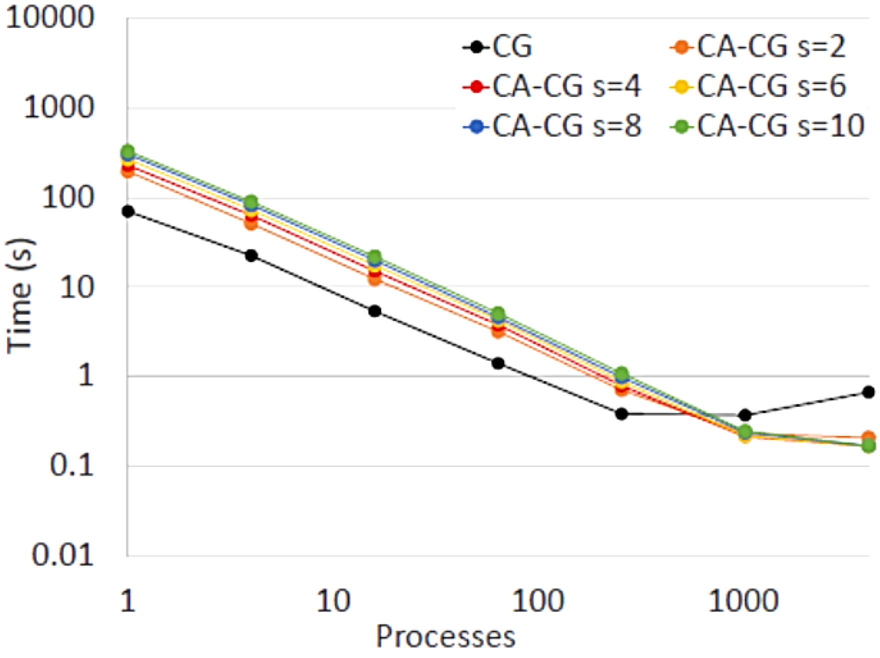
Ballard, Carson, Demmel, Hoemmen, Knight, Schwartz: Arnoldi, GMRES, Nonsymm. Lanczos

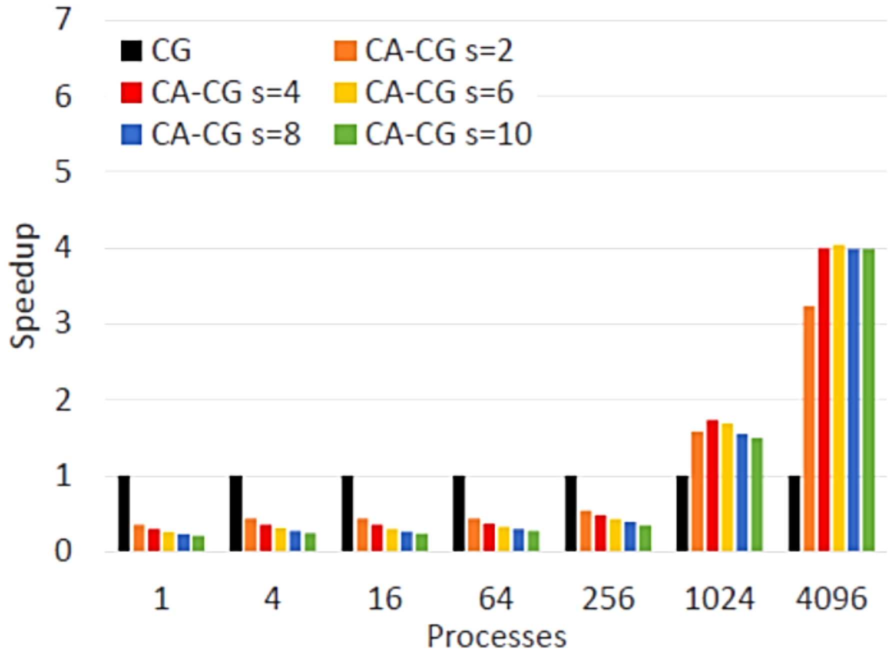
2014



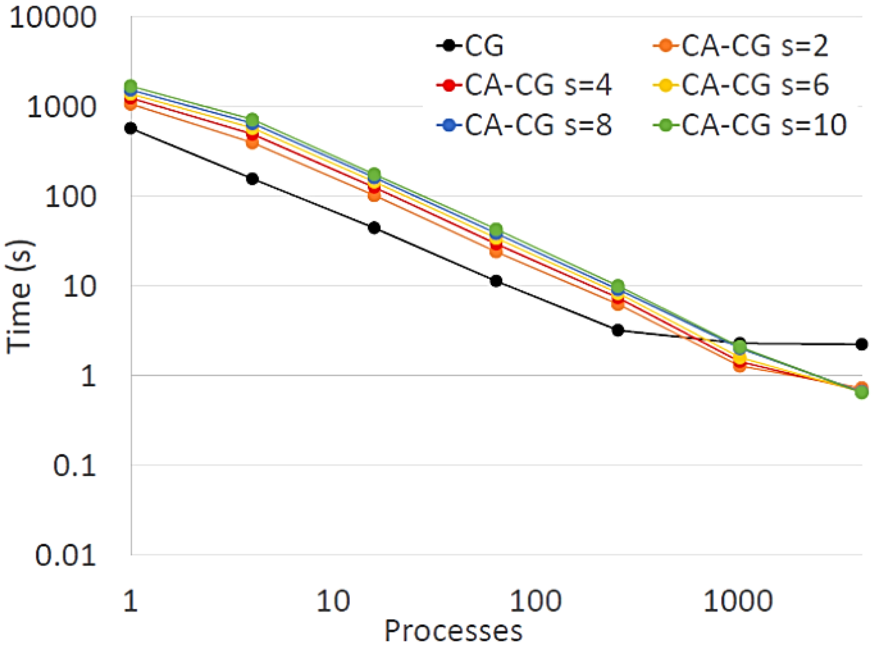


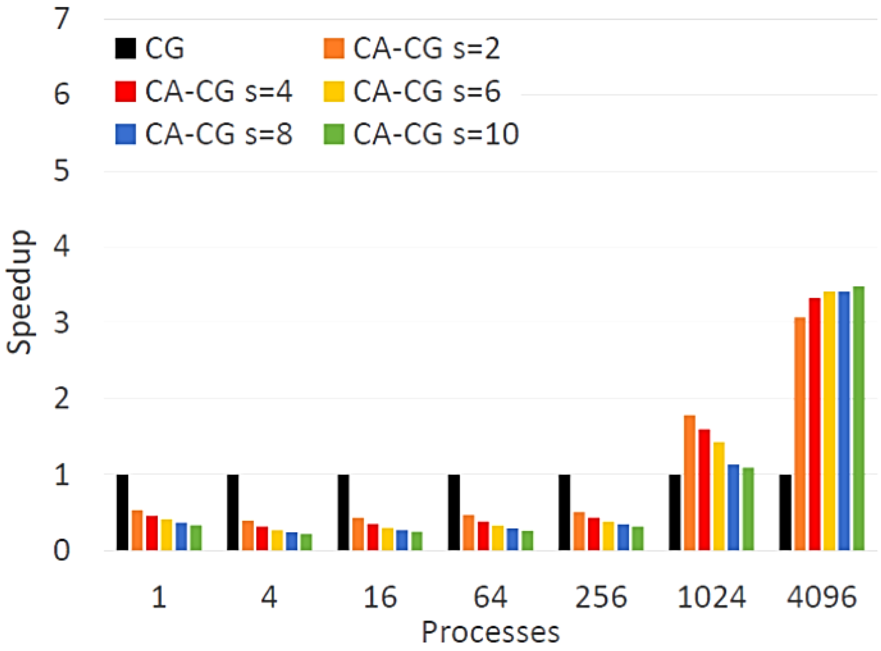
Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 512^2 grid



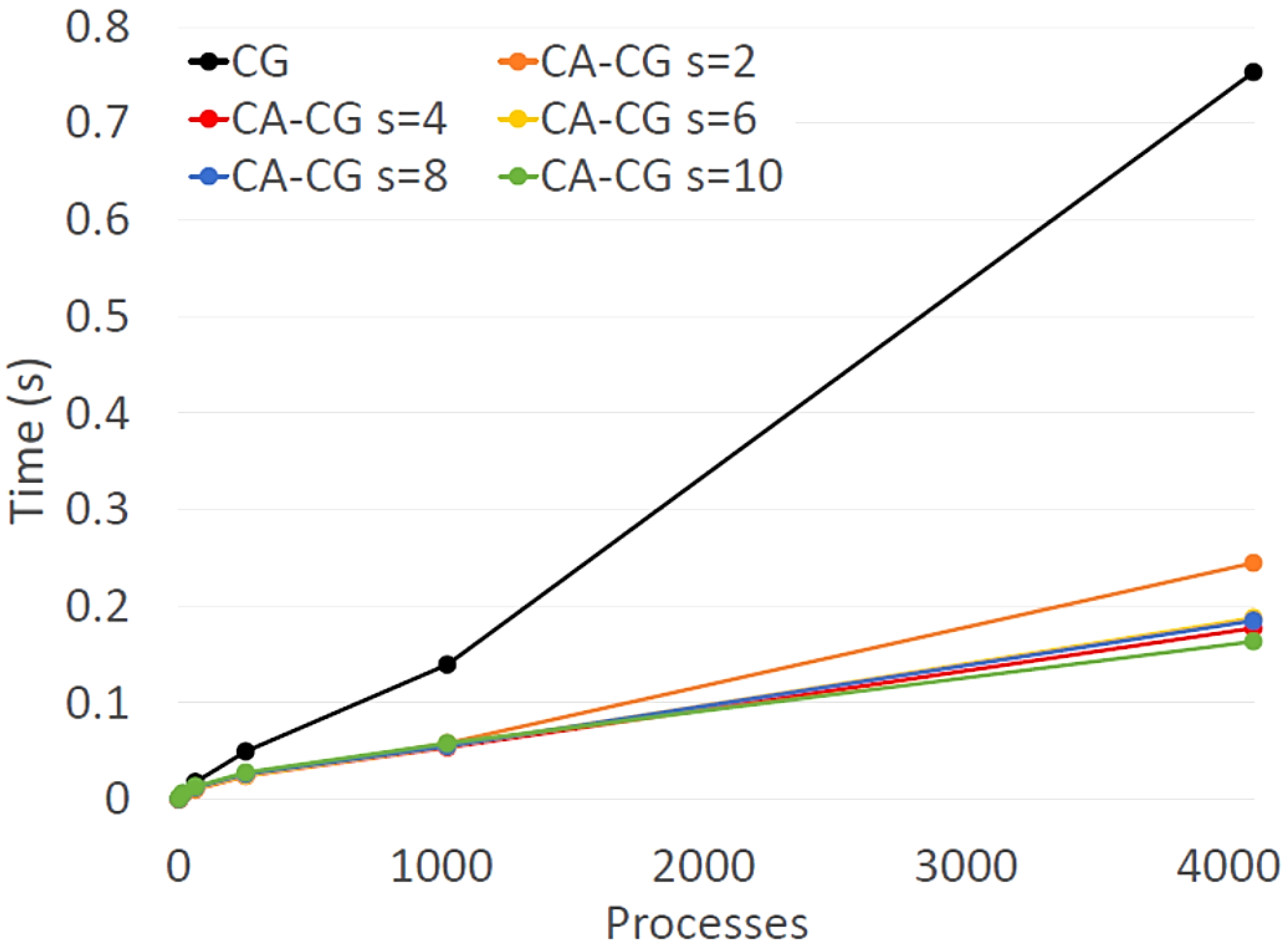


Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 1024^2 grid

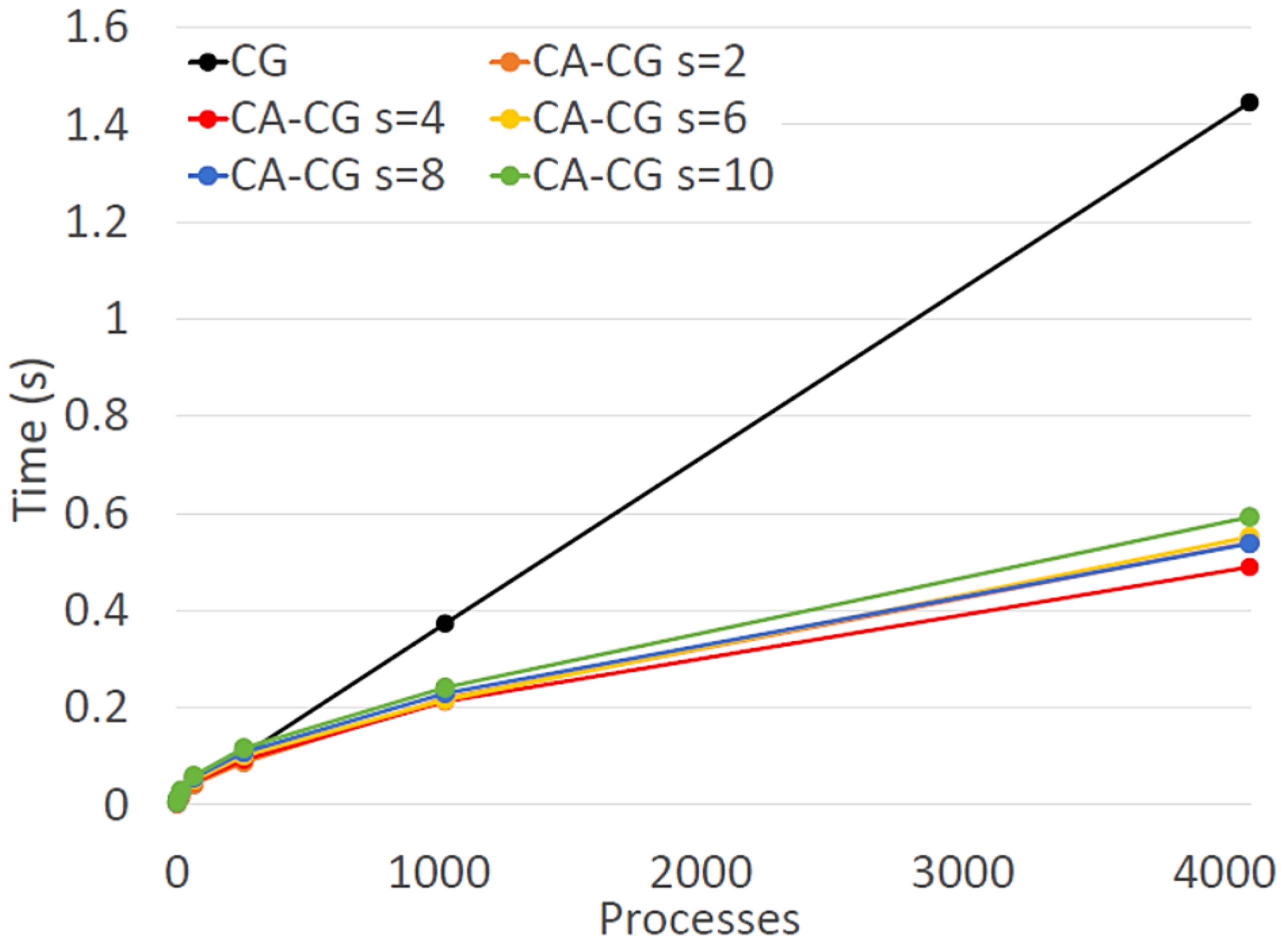




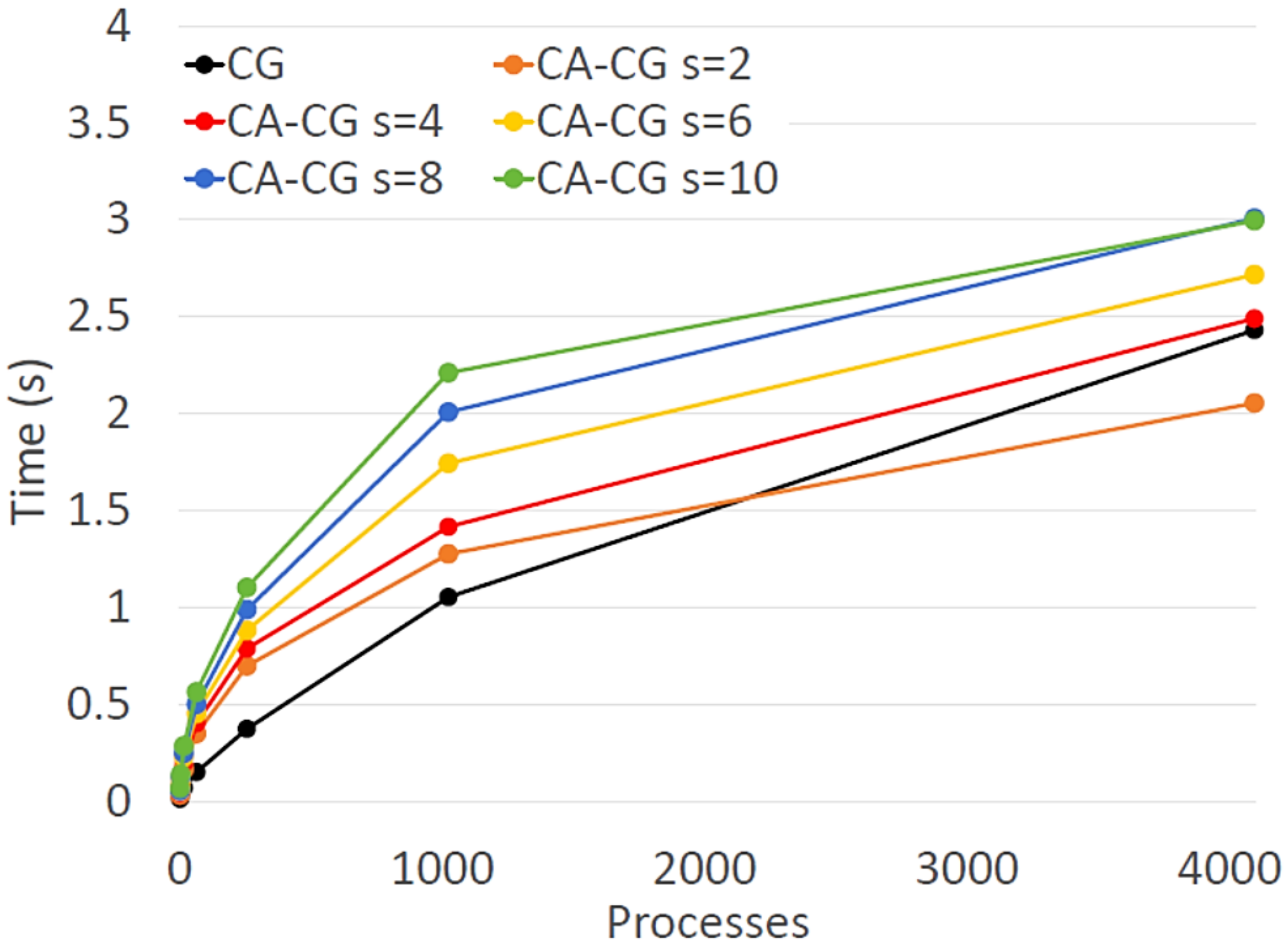
Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 2048^2 grid



Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 16^2 grid per process



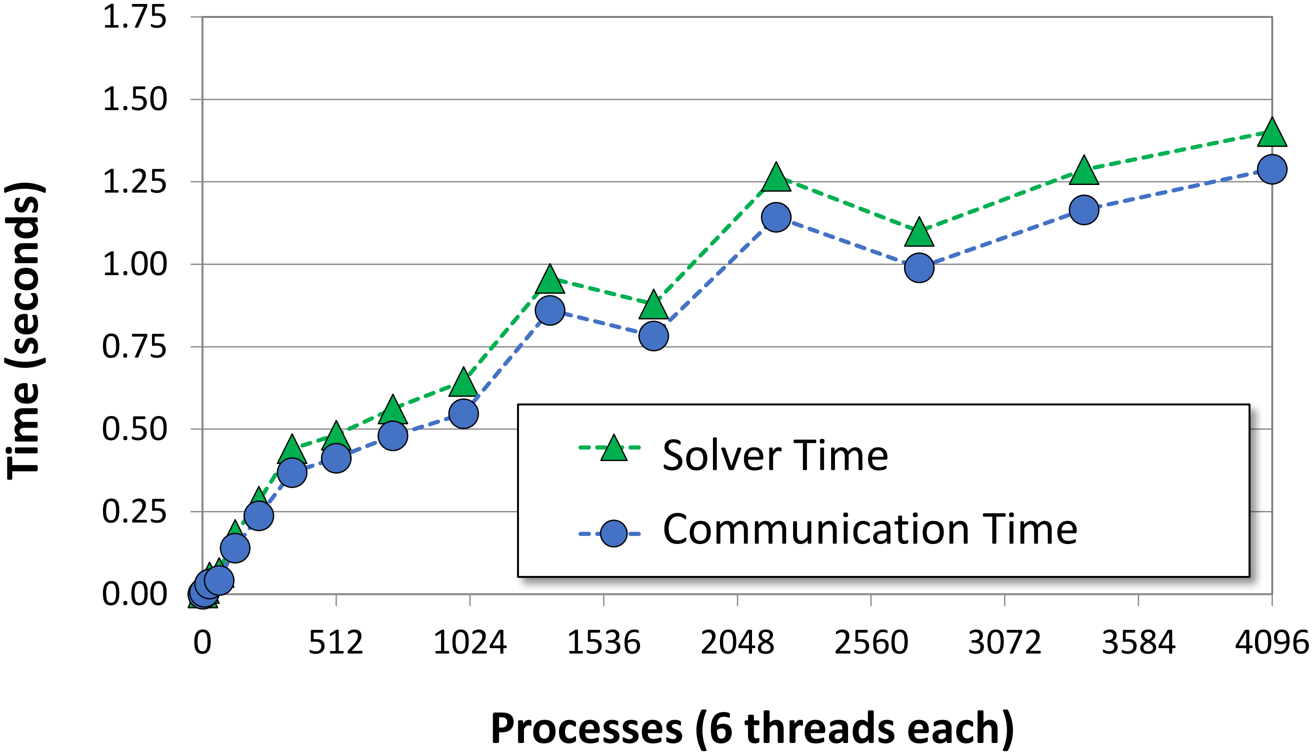
Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 32^2 grid per process



Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 64^2 grid per process

Coarse-grid Krylov Solver on NERSC’s Hopper (Cray XE6)

Weak Scaling: 4^3 points per process (0 slope ideal)



Solver performance and scalability limited by communication!

Communication-Avoiding Krylov Method Speedups

- Recent results: CA-BICGSTAB used as geometric multigrid (GMG) bottom-solve (Williams, Carson, et al., IPDPS '14)
- Plot: Net time spent on different operations over one GMG bottom solve using 24,576 cores, 64^3 points/core on fine grid, 4^3 points/core on coarse grid
- Hopper at NERSC (Cray XE6), 4 6-core Opteron chips per node, Gemini network, 3D torus

- 3D Helmholtz equation

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

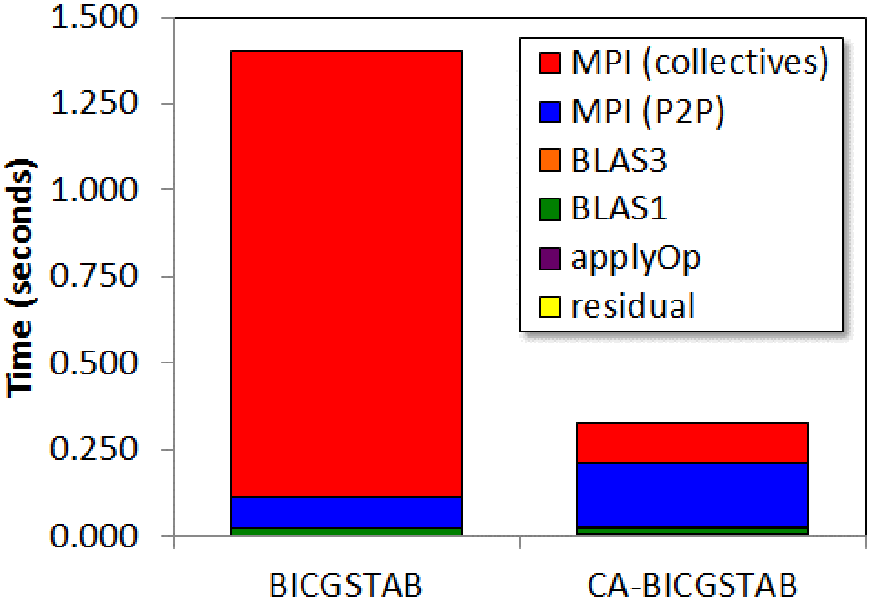
$$\alpha = \beta = 1.0,\ a = b = 0.9$$

- **CA-BICGSTAB with $s = 4$**

  4.2x speedup in Krylov solve;

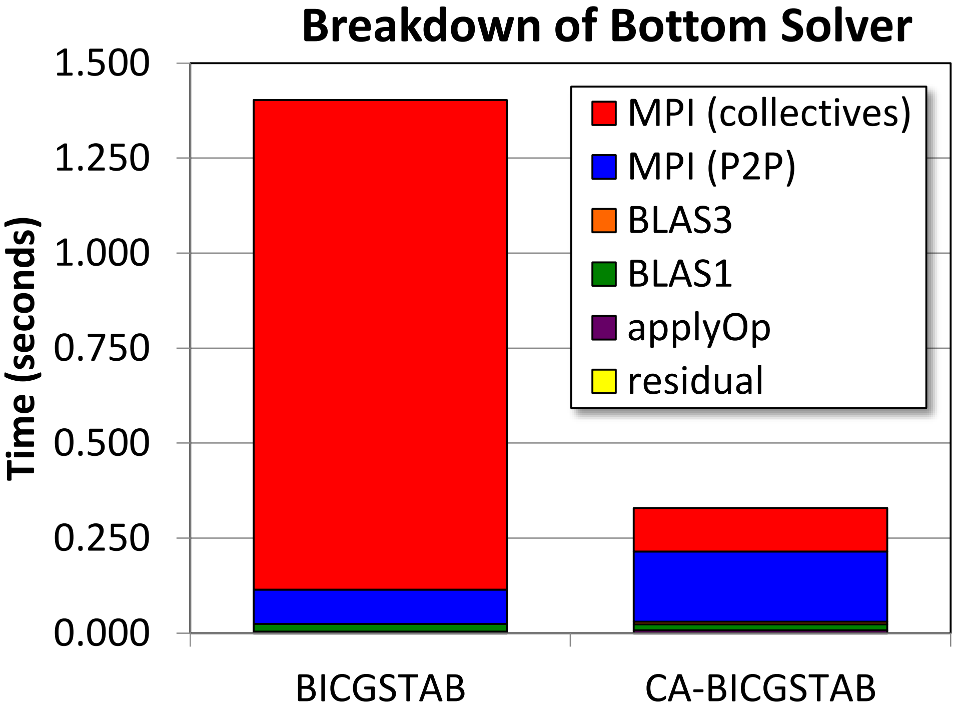
  2.5x in overall GMG solve

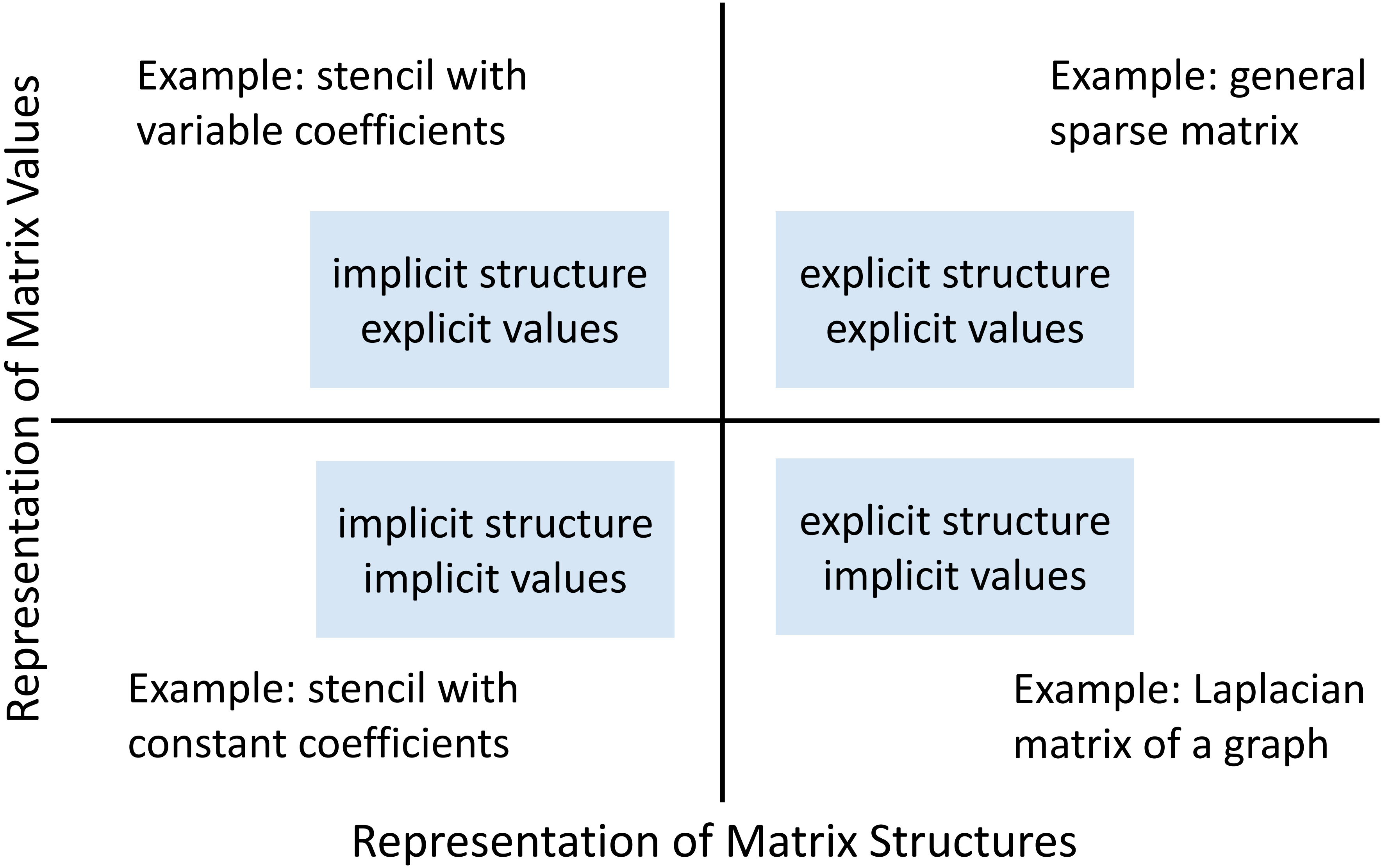
- Implemented in BoxLib: applied to low-Mach number combustion and 3D N-body dark matter simulation apps



Benchmark timing breakdown

- Plot: Net time spent across all bottom solves at 24,576 cores, for BICGSTAB and CA-BICGSTAB with $s = 4$
- **11.2x reduction in MPI_AllReduce time (red)**
  - BICGSTAB requires $6s$ more MPI_AllReduce's than CA-BICGSTAB
  - Less than theoretical 24x since messages in CA-BICGSTAB are larger, not always latency-limited
- **P2P (blue) communication doubles** for CA-BICGSTAB
  - Basis computation requires twice as many SpMVs (P2P) per iteration as BICGSTAB





Example: stencil with variable coefficients

implicit structure
explicit values

Example: general sparse matrix

explicit structure
explicit values

implicit structure
implicit values

Example: stencil with constant coefficients

explicit structure
implicit values

Example: Laplacian matrix of a graph

Representation of Matrix Values

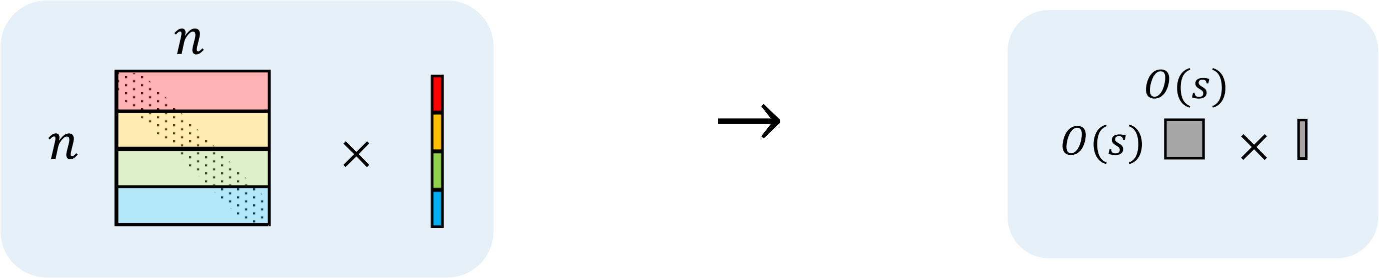
Representation of Matrix Structures

Hoemmen (2010), Fig 2.5

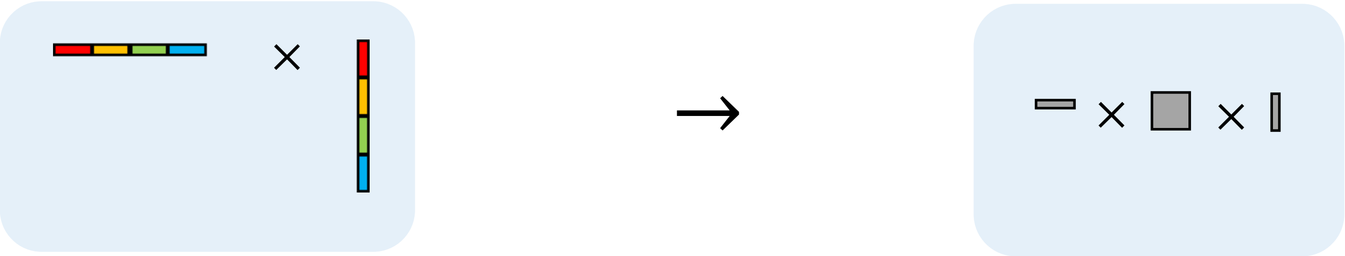
s-step (communication-avoiding) CG

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

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



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



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

group update of approximate solution

set residual to true residual

$$\|r_i\|_2 = \mu_i^{(2)} \|A\|_2 \|x - \widehat{x}_i\|_2. \tag{2.10}$$

We have

$$x - \widehat{x}_i = V\Sigma^{-1}U^T r_i = \sum_{j=1}^{n} \frac{(u_j^T r_i)v_j}{\sigma_j},$$

and so

$$\|x - \widehat{x}_i\|_2^2 \geq \sum_{j=n+1-k}^{n} \frac{(u_j^T r_i)^2}{\sigma_j^2} \geq \frac{1}{\sigma_{n+1-k}^2} \sum_{j=n+1-k}^{n} (u_j^T r_i)^2 = \frac{\|P_k r_i\|_2^2}{\sigma_{n+1-k}^2},$$

where $P_k = U_k U_k^T$ with $U_k = [u_{n+1-k}, \ldots, u_n]$. Hence from (2.10) we have

$$\mu_i^{(2)} \leq \frac{\|r_i\|_2}{\|P_k r_i\|_2} \frac{\sigma_{n+1-k}}{\sigma_1}.$$

The bound tells us that $\mu_i^{(2)}$ will be much less than 1 if r_i contains a significant component in the subspace $\operatorname{span}(U_k)$ for any k such that $\sigma_{n+1-k} \approx \sigma_n$.

This argument says that we can expect $\mu_i^{(2)} \ll 1$ when r_i is a “typical” vector—one having sizeable components in the direction of every left singular vector of A—in which case $x - \widehat{x}_i$ is not typical, in that it has large components in the direction of the right singular vectors of A corresponding to small singular values.