MS 28: Scalable Communication-Avoiding and -Hiding Krylov Subspace Methods I

Organizers: Siegfried Cools, University of Antwerp, Belgium

Erin C. Carson, New York University, USA

10:50-11:10 **High Performance Variants of Krylov Subspace Methods**

Erin C. Carson, New York University, USA

11:15-11:35 **About Parallel Variants of GMRES Algorithm**

Jocelyne Erhel, Inria-Rennes, France

11:40-12:00 **Enlarged GMRES for Reducing Communication**

OlivierTissot, Inria, France

MS 40: Scalable Communication-Avoiding and -Hiding Krylov Subspace Methods II

Organizers: Siegfried Cools, University of Antwerp, Belgium
Erin C. Carson, New York University, USA

2:40-3:00 **Impact of Noise Models on Pipelined Krylov Methods**
Hannah Morgan, University of Chicago, USA

3:05-3:25 **Scalable Krylov Methods for Spectral Graph Partitioning**
Pieter Ghysels, Lawrence Berkeley National Laboratory, USA

3:30-3:50 **Using Non-Blocking Communication to Achieve Scalability for Preconditioned Conjugate Gradient Methods**
William D. Gropp, University of Illinois at Urbana-Champaign, USA

3:55-4:15 **Performance of S-Step and Pipelined Krylov Methods**
Piotr Luszczek, University of Tennessee, Knoxville, USA

High Performance Variants of Krylov Subspace Methods

Erin Carson
New York University

SIAM PP18, Tokyo, Japan
March 8, 2018

Collaborators

Emmanuel Agullo, Inria, France

Siegfried Cools, University of Antwerp, Belgium

James Demmel, University of California, Berkeley, USA

Pieter Ghysels, Lawrence Berkeley National Laboratory, USA

Luc Giraud, Inria, France

Miro Rozložník, Czech Academy of Sciences, Czech Republic

Zdeněk Strakoš, Charles University, Czech Republic

Petr Tichý, Czech Academy of Sciences, Czech Republic

Miroslav Tůma, Czech Academy of Sciences, Czech Republic

Wim Vanroose, Antwerp University, Belgium

Emrullah Fatih Yetkin, Inria, France

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)

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

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
- Reducing time spent moving data/waiting for data will be essential for applications at exascale!

$\Rightarrow$ *communication avoiding & communication hiding*

Krylov Subspace Methods

Krylov Subspace Method: projection process onto the Krylov subspace

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

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

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

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

In each iteration:

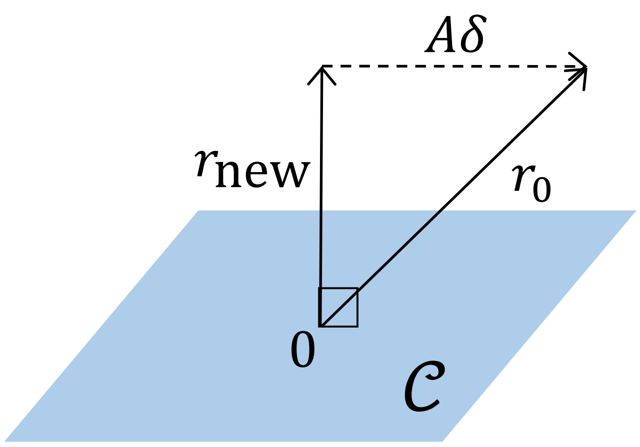
- Add a dimension to the Krylov subspace
  - Forms nested sequence of Krylov subspaces

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

- 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

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

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

In each iteration:

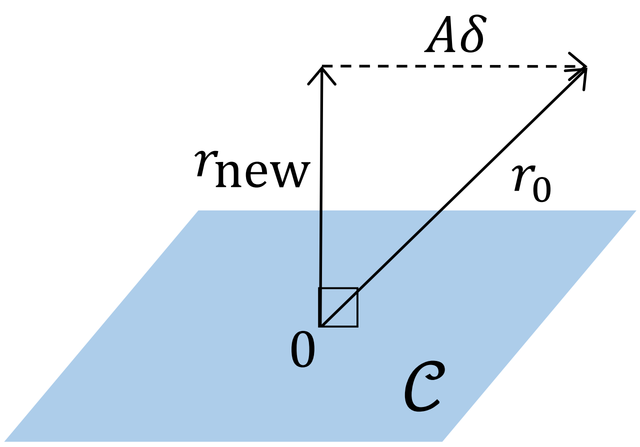
- Add a dimension to the Krylov subspace
  - Forms nested sequence of Krylov subspaces

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

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



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

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

Conjugate Gradient on the World's Fastest Computer

Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway

Site:	National Supercomputing Center in Wuxi
Manufacturer:	NRCPC
Cores:	10,649,600
Memory:	1,310,720 GB
Processor:	Sunway SW26010 260C 1.45GHz
Interconnect:	Sunway
Performance	
Linpack Performance (Rmax)	93,014.6 TFlop/s
Theoretical Peak (Rpeak)	125,436 TFlop/s
Nmax	12,288,000
HPCG [TFlop/s]	480.8

Conjugate Gradient on the World's Fastest Computer

Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway

Site:	National Supercomputing Center in Wuxi
Manufacturer:	NRCPC
Cores:	10,649,600
Memory:	1,310,720 GB
Processor:	Sunway SW26010 260C 1.45GHz
Interconnect:	Sunway
Performance	
Linpack Performance (Rmax)	93,014.6 TFlop/s
Theoretical Peak (Rpeak)	125,436 TFlop/s
Nmax	12,288,000
HPCG [TFlop/s]	480.8

current #1 on top500

Conjugate Gradient on the World's Fastest Computer

Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway

Site:	National Supercomputing Center in Wuxi
Manufacturer:	NRCPC
Cores:	10,649,600
Memory:	1,310,720 GB
Processor:	Sunway SW26010 260C 1.45GHz
Interconnect:	Sunway
Performance	
Linpack Performance (Rmax)	93,014.6 TFlop/s
Theoretical Peak (Rpeak)	125,436 TFlop/s
Nmax	12,288,000
HPCG [TFlop/s]	480.8

current #1 on top500

Linpack benchmark (dense $Ax = b$, direct) 74% efficiency

Conjugate Gradient on the World's Fastest Computer

Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway

Site:	National Supercomputing Center in Wuxi
Manufacturer:	NRCPC
Cores:	10,649,600
Memory:	1,310,720 GB
Processor:	Sunway SW26010 260C 1.45GHz
Interconnect:	Sunway
Performance	
Linpack Performance (Rmax)	93,014.6 TFlop/s
Theoretical Peak (Rpeak)	$\boxed{\text{125,436 TFlop/s}}$
Nmax	12,288,000
HPCG [TFlop/s]	480.8

current #1
on top500

Linpack benchmark
(dense $Ax = b$, direct)
74% efficiency

HPCG benchmark
(sparse $Ax = b$, iterative)
0.4% 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

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

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

Iteration Loop

↓

Sparse Matrix × Vector

The Conjugate Gradient (CG) Method

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

for i = 1:nmax

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

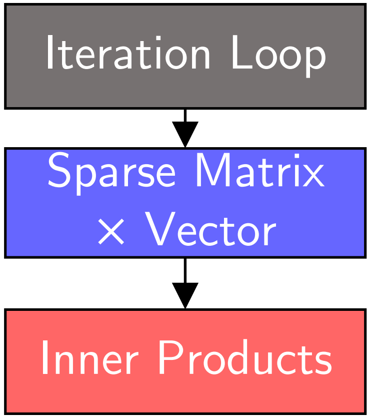
$$x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$$

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

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

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

end



The Conjugate Gradient (CG) Method

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

for i = 1:nmax

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

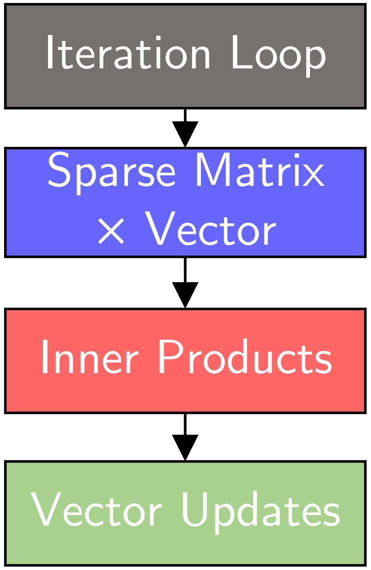
$$x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$$

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

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

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

end



The Conjugate Gradient (CG) Method

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

for i = 1:nmax

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

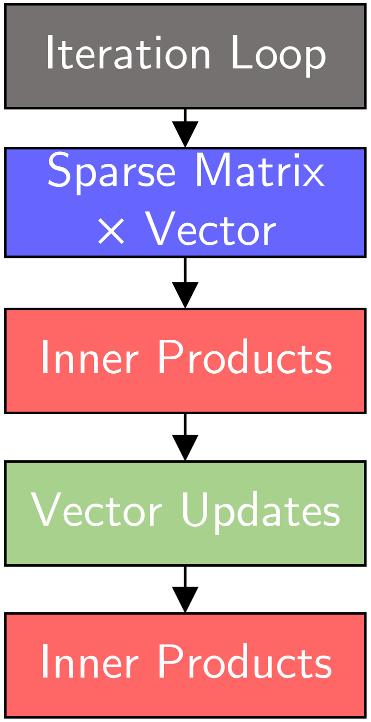
$$x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$$

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

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

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

end



The Conjugate Gradient (CG) Method

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

for i = 1:nmax

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

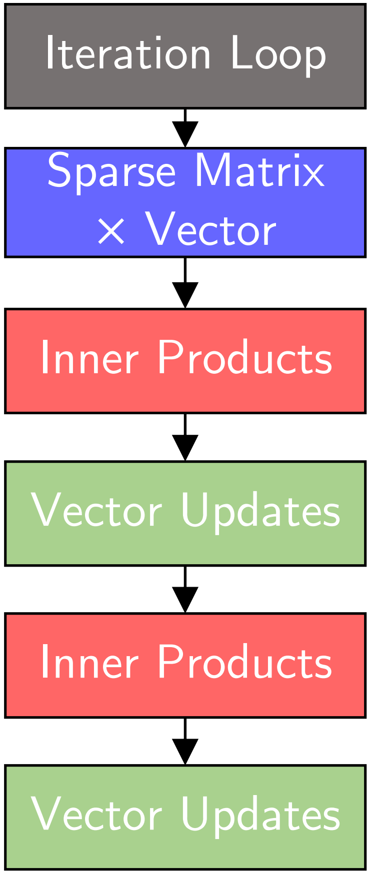
$$x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$$

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

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

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

end



The Conjugate Gradient (CG) Method

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

for i = 1:nmax

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

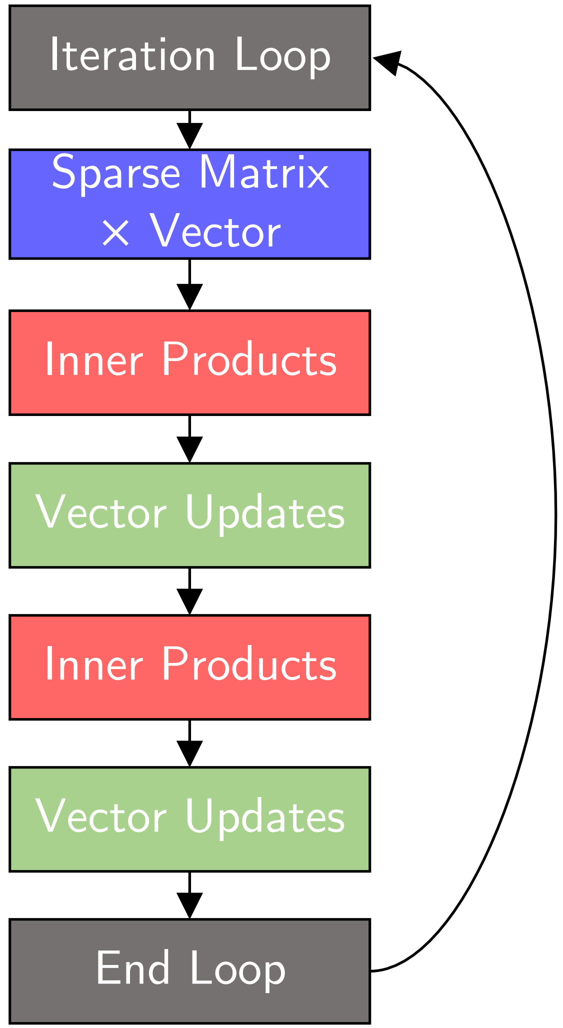
$$x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$$

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

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

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

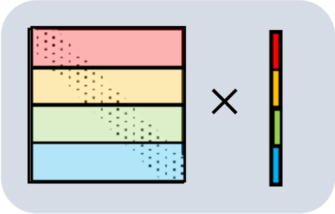
end



Cost Per Iteration

→ Sparse matrix-vector multiplication (SpMV)

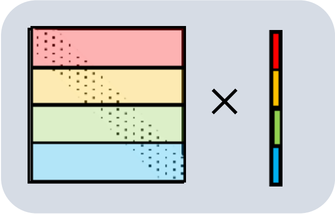
- O(nnz) flops
- Must communicate vector entries w/neighboring processors (nearest neighbor MPI collective)



Cost Per Iteration

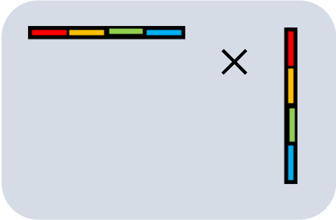
→ Sparse matrix-vector multiplication (SpMV)

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



→ Inner products

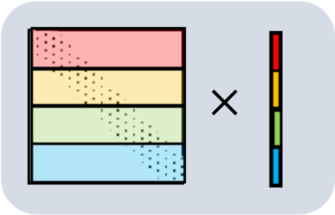
- $O(N)$ flops
- global synchronization (MPI_Allreduce)
- all processors must exchange data and wait for *all* communication to finish before proceeding



Cost Per Iteration

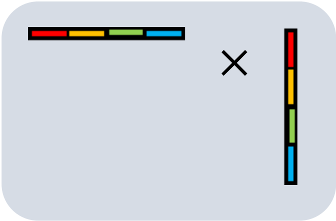
→ Sparse matrix-vector multiplication (SpMV)

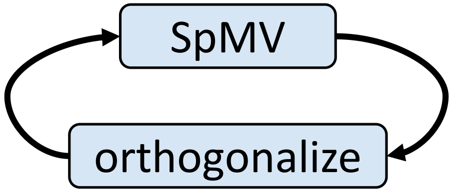
- $O(\text{nnz})$ flops
- Must communicate vector entries w/neighboring processors (nearest neighbor MPI collective)



→ Inner products

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





Low computation/communication ratio

⇒ Performance is communication-bound

Reducing Synchronization Cost

Communication cost has motivated many approaches to reducing synchronization cost in Krylov subspace methods:

Hiding communication: Pipelined Krylov subspace methods

- Introduce auxiliary vectors to decouple SpMV and inner products
- Enables overlapping of communication and computation

Avoiding communication: s-step Krylov subspace methods

- Compute iterations in blocks of s (using a different Krylov subspace basis)
- Reduces number of synchronizations per iteration by a factor of O(s)

* Both equivalent to classical CG in exact arithmetic

Pipelined CG (Ghysels and Vanroose 2013)

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

$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$

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

Pipelined CG (Ghysels and Vanroose 2013)

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

$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$

$\alpha_0 = r_0^T r_0 / p_0^T s_0$

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

- Uses auxiliary vectors

$$s_i \equiv Ap_i,\ w_i \equiv Ar_i,\ z_i \equiv A^2 r_i$$

Pipelined CG (Ghysels and Vanroose 2013)

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

$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$

$\alpha_0 = r_0^T r_0 / p_0^T s_0$

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

- Uses auxiliary vectors

$$s_i \equiv Ap_i,\ w_i \equiv Ar_i,\ z_i \equiv A^2 r_i$$

- Removes sequential dependency between SpMV and inner products

- Allows the use of nonblocking (asynchronous) MPI communication to *overlap* SpMV and inner products
  - See talk by W. Gropp in Part II: MS40

- Hides the latency of global communications

Pipelined CG (Ghysels and Vanroose 2013)

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

$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$

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

Pipelined CG (Ghysels and Vanroose 2013)

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

$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$

for i = 1:nmax

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

$$r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$$

$$w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$$

$$q_i = Aw_i$$

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

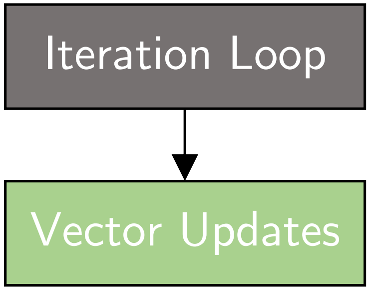
$$\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i/\alpha_{i-1}) r_i^T r_i}$$

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

$$s_i = w_i + \beta_i s_{i-1}$$

$$z_i = q_i + \beta_i z_{i-1}$$

end



Pipelined CG (Ghysels and Vanroose 2013)

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

$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$

for $i = 1$:nmax

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

$r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$

$w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$

$q_i = Aw_i$

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

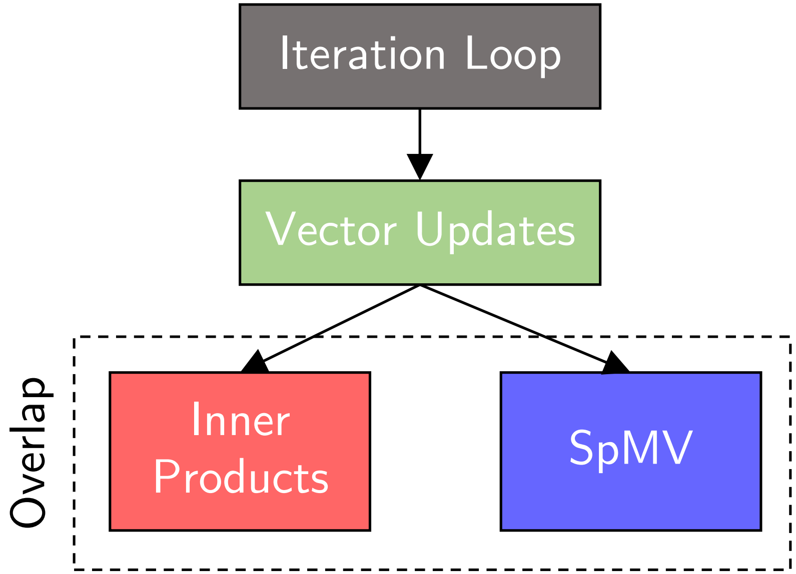
$\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i}$

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

$s_i = w_i + \beta_i s_{i-1}$

$z_i = q_i + \beta_i z_{i-1}$

end



Pipelined CG (Ghysels and Vanroose 2013)

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

$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$

for i = 1:nmax

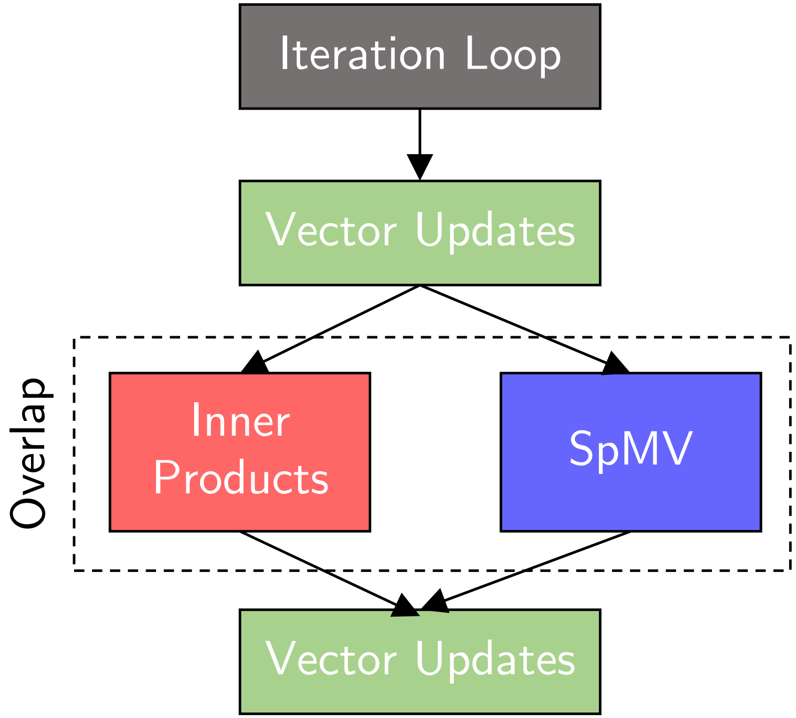
$$x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$$
$$r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$$
$$w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$$

$$q_i = Aw_i$$

$$\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}$$
$$\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i/\alpha_{i-1}) r_i^T r_i}$$

$$p_i = r_i + \beta_i p_{i-1}$$
$$s_i = w_i + \beta_i s_{i-1}$$
$$z_i = q_i + \beta_i z_{i-1}$$

end



Pipelined CG (Ghysels and Vanroose 2013)

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

$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$

for i = 1:nmax

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

$r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$

$w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$

$q_i = Aw_i$

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

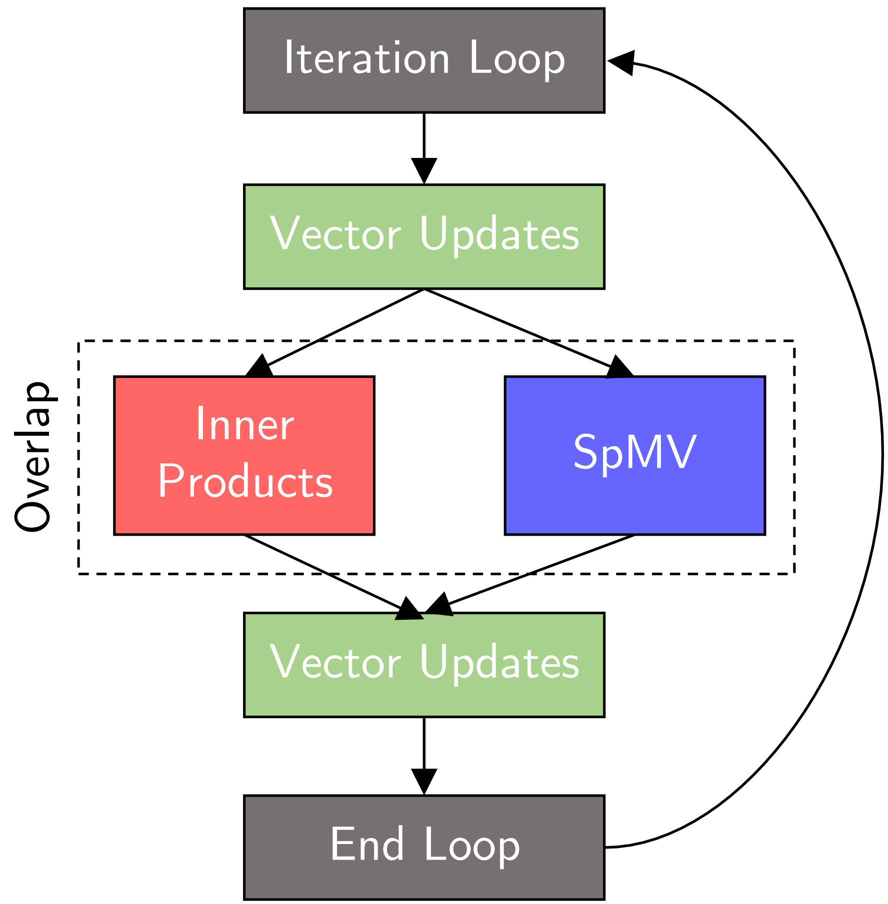
$$\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i/\alpha_{i-1}) r_i^T r_i}$$

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

$s_i = w_i + \beta_i s_{i-1}$

$z_i = q_i + \beta_i z_{i-1}$

end



Pipelined CG (Ghysels and Vanroose 2013)

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

$s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0,$
$\alpha_0 = r_0^T r_0 / p_0^T s_0$

for i = 1:nmax

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

$r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$

$w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$

$q_i = Aw_i$

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

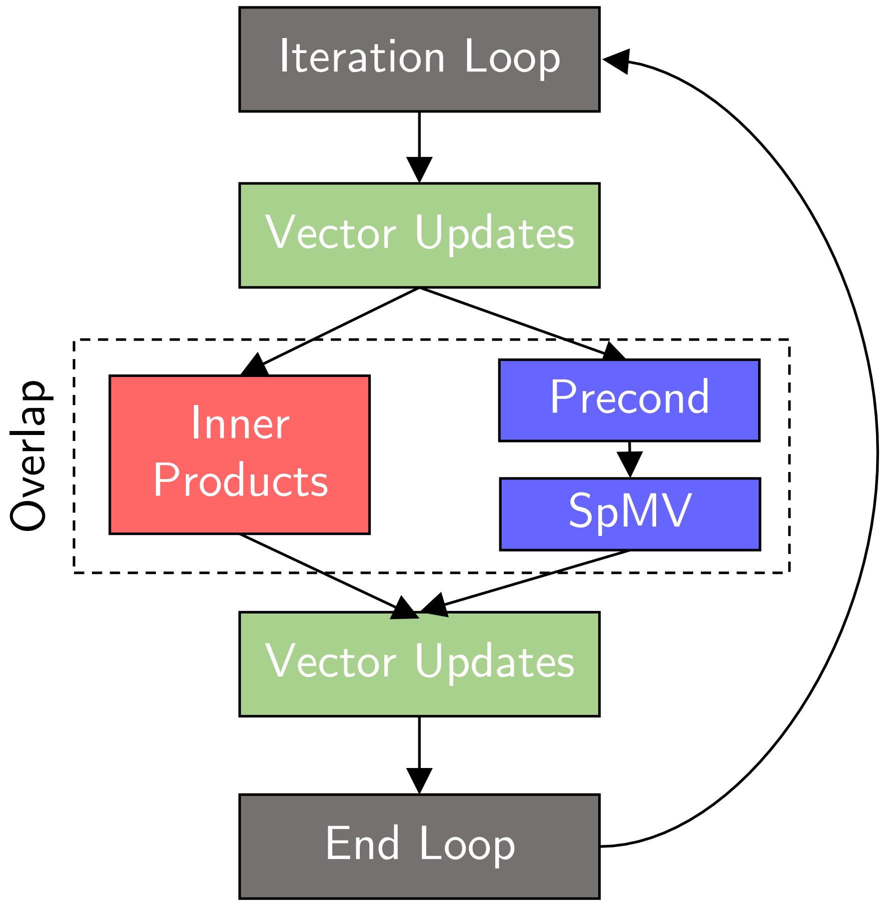
$\alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i}$

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

$s_i = w_i + \beta_i s_{i-1}$

$z_i = q_i + \beta_i z_{i-1}$

end



Overview of Pipelined KSMs

- Pipelined GMRES (Ghysels et al. 2013)
  - Deep pipelines - compute ℓ new Krylov basis vectors during global communication, orthogonalize after ℓ iterations
    - Talk by W. Vanroose, IP7 Sat March 10
- Pipelined CG (Ghysels et al. 2013)
  - With deep pipelines (Cornelis et al. 2018)
- Pipelined BiCGSTAB (Cools et al. 2017)
- Probabilistic performance modeling of pipelined KSMs
  - Talk by H. Morgan, Part II: MS40

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

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

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

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

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

- Block iterations into groups of s
- Construct basis matrix $\mathcal{Y}_k$ to expand Krylov subspace s dimensions at once
  - Same latency cost as 1 SpMV (under assumptions on sparsity)
- 1 global synchronization to compute inner products between basis vectors
- Update coordinates of iteration vectors in the constructed basis
  - requires no communication

s-step CG

Outer Loop

$r_0 = b - Ax_0, p_0 = r_0$

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

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

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

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

for $j = 1: s$

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

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

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

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

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

end

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

end

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

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

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

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

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

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

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

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

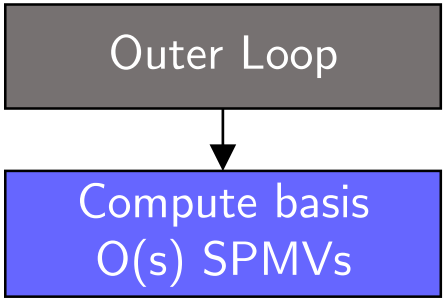
$$\beta_{sk+j} = \frac{r_j'^T \mathcal{G}_k r_j'}{r_{j-1}'^T \mathcal{G}_k r_{j-1}'}$$

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

end

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

end



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

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

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

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

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

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

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

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

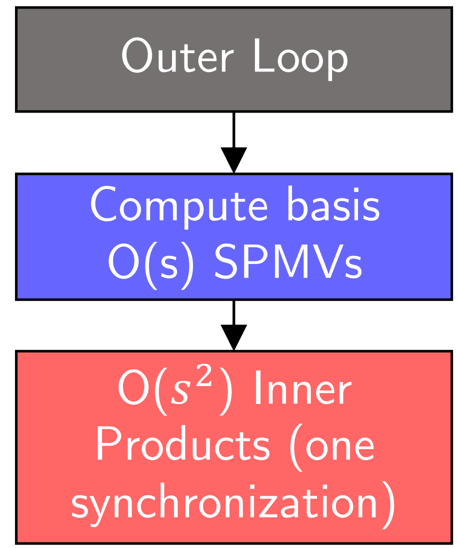
$$\beta_{sk+j} = \frac{r_j'^T\mathcal{G}_k r_j'}{r_{j-1}'^T\mathcal{G}_k r_{j-1}'}$$

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

end

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

end



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 $j = 1{:}\, s$

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

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

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

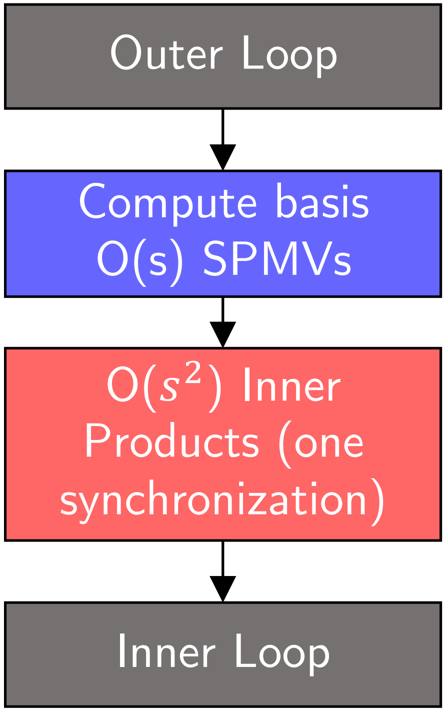
$$\beta_{sk+j} = \frac{r_j'^T \mathcal{G}_k r_j'}{r_{j-1}'^T \mathcal{G}_k r_{j-1}'}$$

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

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



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 $A\mathcal{Y}_k = \mathcal{Y}_k\mathcal{B}_k$ 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 $j = 1{:}\, s$

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

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

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

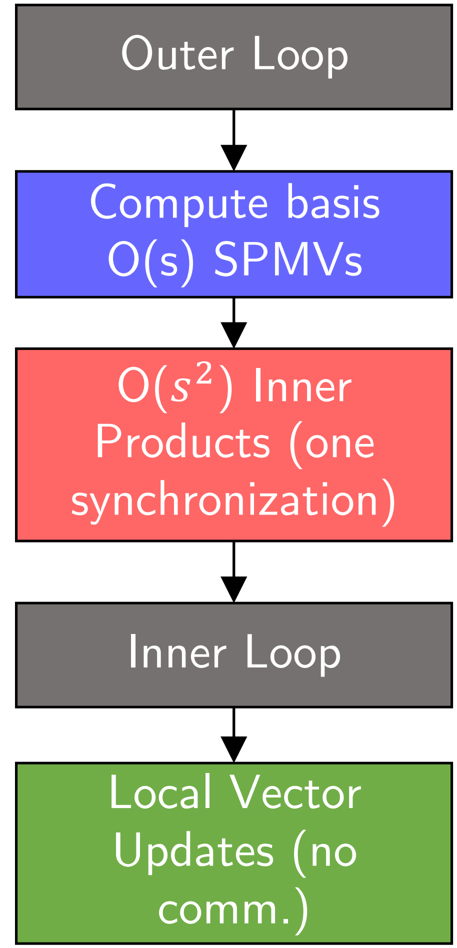
$$\beta_{sk+j} = \frac{r_j'^T \mathcal{G}_k r_j'}{r_{j-1}'^T \mathcal{G}_k r_{j-1}'}$$

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

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



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

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

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

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

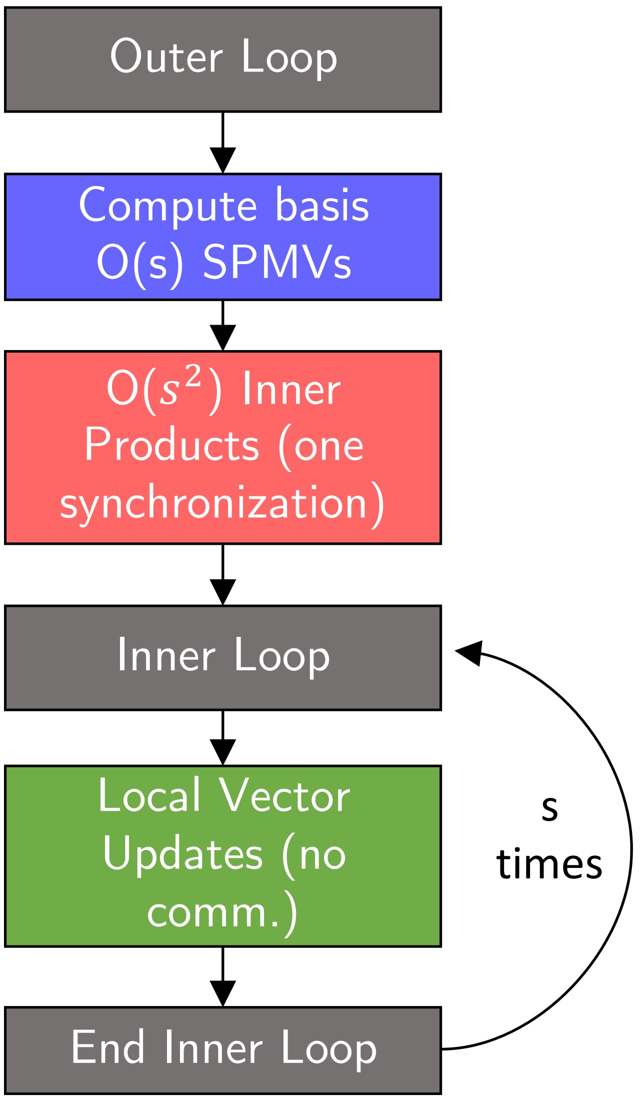
$$\beta_{sk+j} = \frac{r_j'^T \mathcal{G}_k r_j'}{r_{j-1}'^T \mathcal{G}_k r_{j-1}'}$$

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

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



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 $A\mathcal{Y}_k = \mathcal{Y}_k\mathcal{B}_k$ 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 $j = 1{:}\, s$

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

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

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

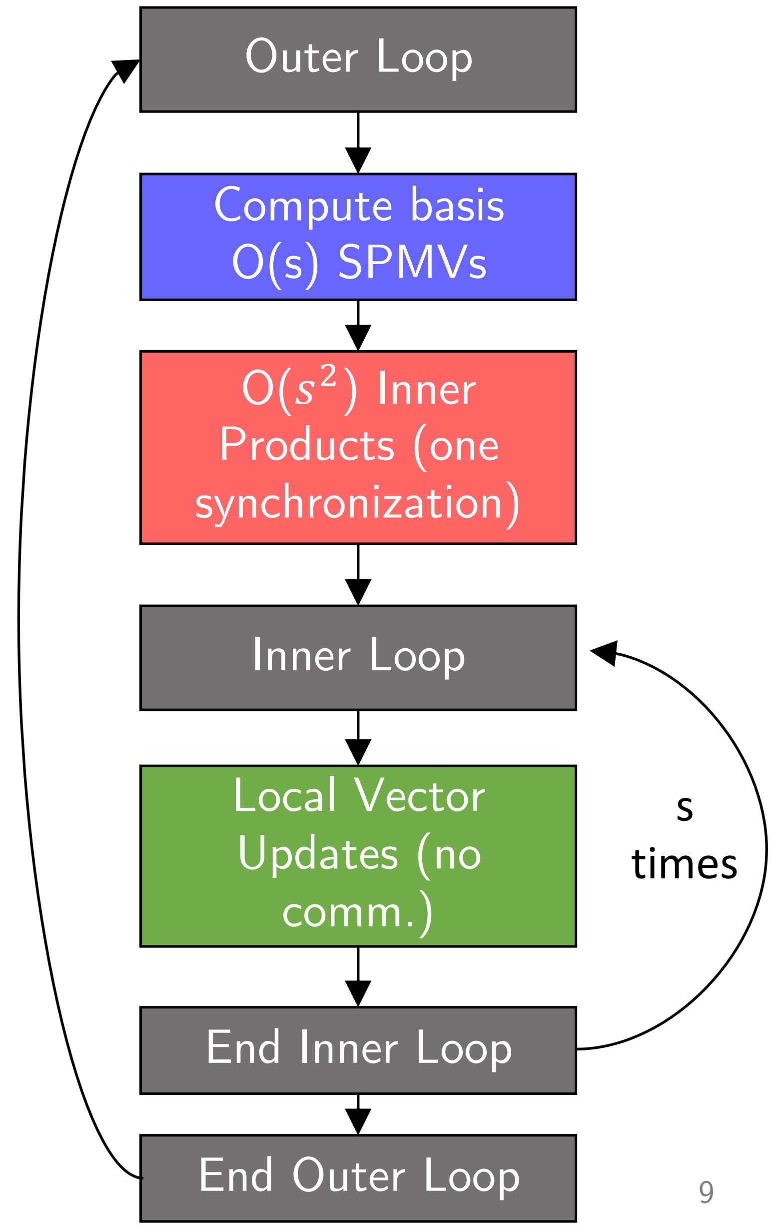
$$\beta_{sk+j} = \frac{r_j'^T \mathcal{G}_k r_j'}{r_{j-1}'^T \mathcal{G}_k r_{j-1}'}$$

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

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



Overview of s-step KSMs

- s-step CG/Lanczos: (Van Rosendale, 1983), (Chronopoulos and Gear, 1989), (Leland, 1989), (Toledo, 1995), (Hoemmen et al., 2010)
- s-step GMRES/Arnoldi: (Walker, 1988), (Chronopoulous and Kim, 1990), (Bai, Hu, Reichel, 1991), (de Sturler, 1991), (Joubert, Carey, 1992), (Erhel, 1995), (Hoemmen et al., 2010)
- s-step BICGSTAB (C. et al., 2012)
- s-step QMR (Feuerriegel, Bücker, 2013)
- s-step LSQR (C., 2015)
- Many others...

- Recent work:
  - Hybrid pipelined s-step methods (Yamazaki et al., 2017)
    - Talk by P. Luszczek in Part II, MS40
  - Improving convergence rate and scalability in preconditioned s-step GMRES methods
    - Talk by J. Erhel in MS28 (this session)

The effect 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 no longer exact!

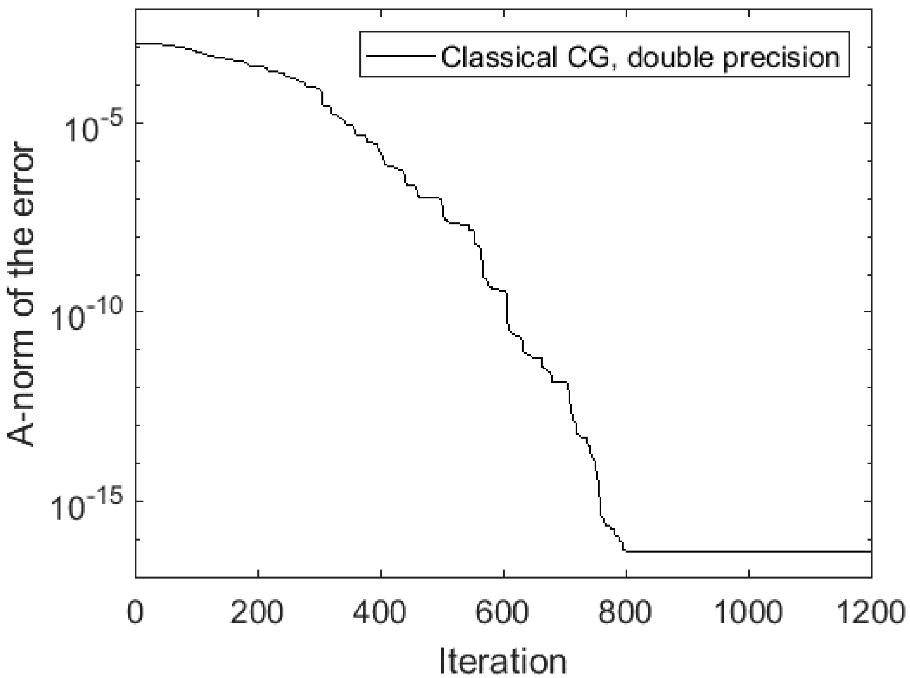
2. Loss of attainable accuracy
    - Rounding errors cause true residual $b - Ax_i$ and updated residual r_i deviate!

The effect 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 no longer exact!

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



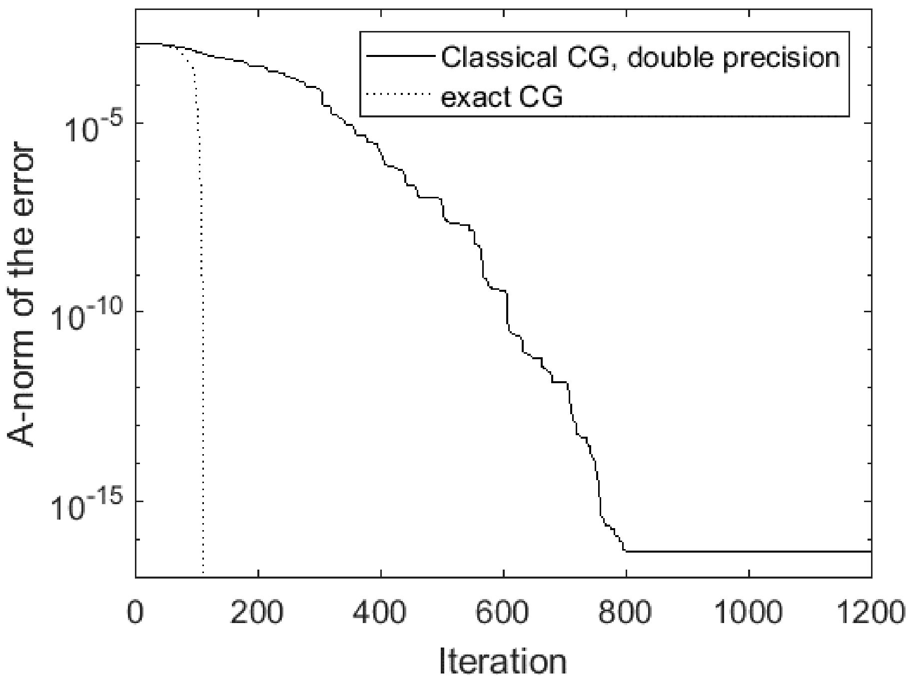
A: bcsstk03 from UFSMC, b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7e6$

The effect 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 no longer exact!

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



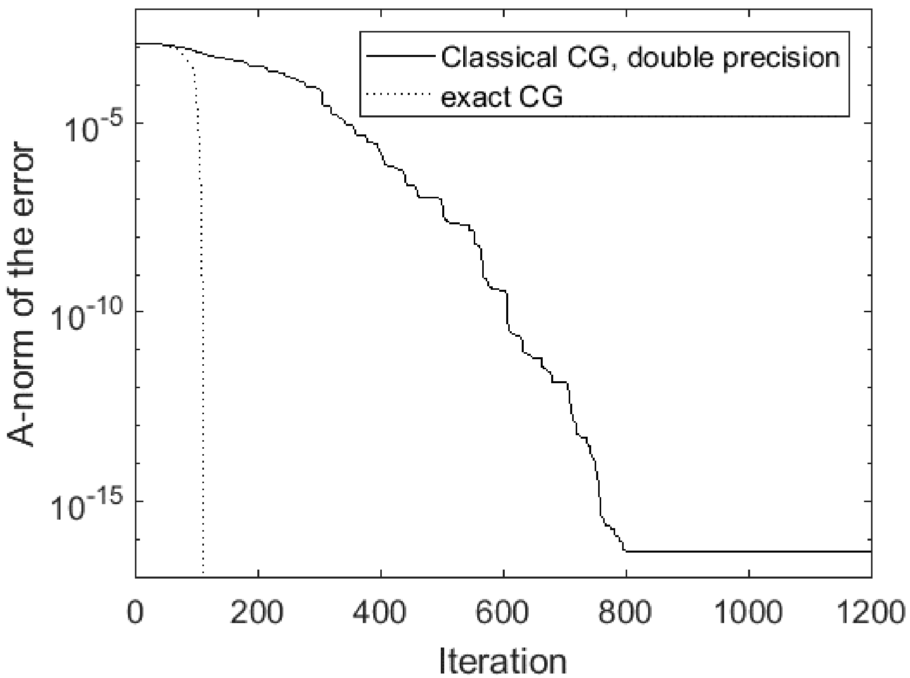
A: bcsstk03 from UFSMC, b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

The effect 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 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 $\|b\| = 1$
$N = 112, \kappa(A) \approx 7\text{e}6$

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

Optimizing high performance iterative solvers

- Synchronization-reducing variants are designed to reduce the time/iteration
- But this is not the whole story!

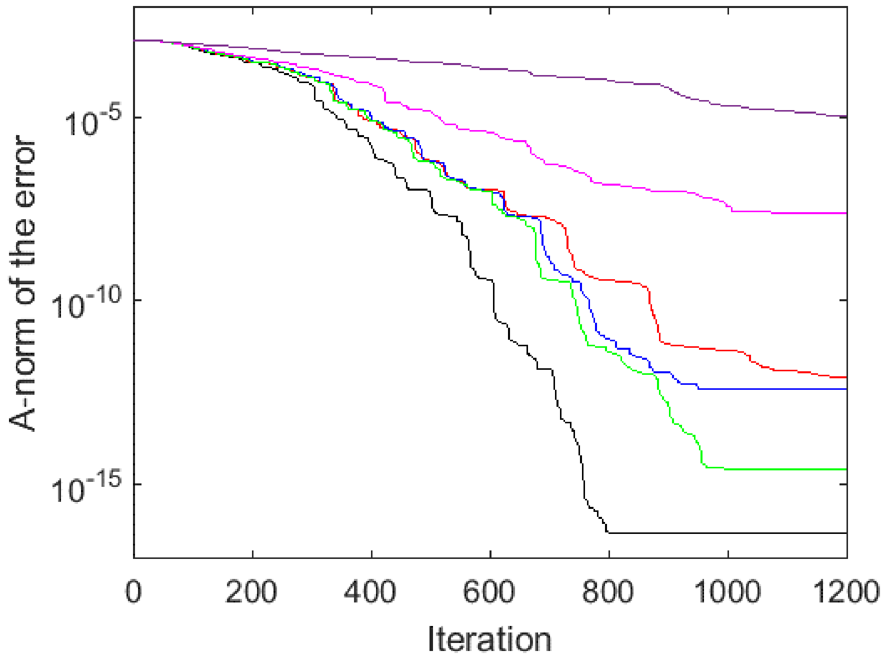
Optimizing high performance iterative solvers

- 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**

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**

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



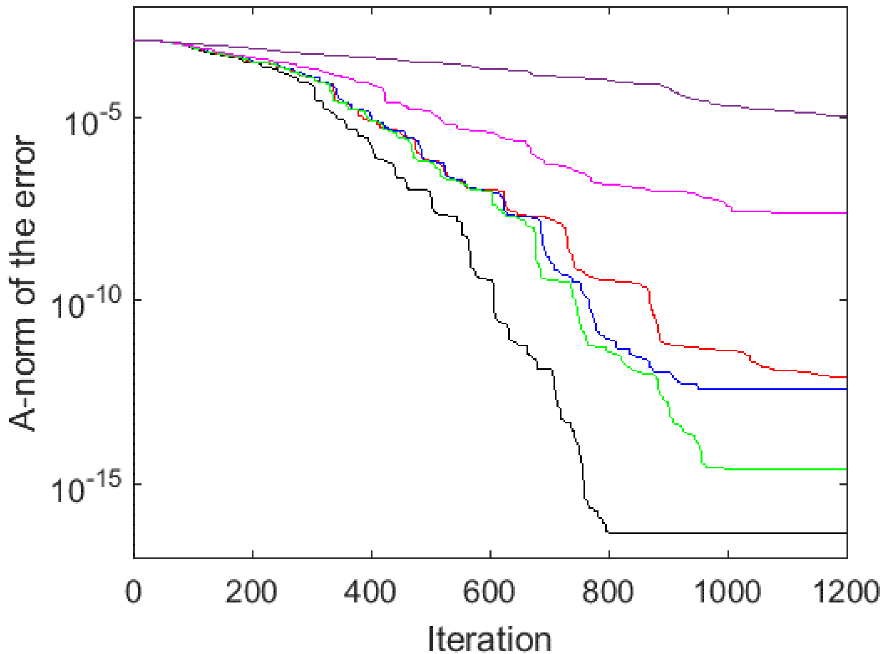
A: bcsstk03 from UFSMC, b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7e6$

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**

- 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!



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

Maximum attainable accuracy

- Accuracy depends on the size of the true residual: $\|b - A\hat{x}_i\|$

Maximum attainable accuracy

- Accuracy depends on the size of the true residual: $\|b - A\hat{x}_i\|$
- Rounding errors cause the **true residual, $\boldsymbol{b - A\hat{x}_i}$**, and the **updated residual, $\boldsymbol{\hat{r}_i}$**, to deviate

Maximum attainable accuracy

- Accuracy depends on the size of the true residual: $\|b - A\hat{x}_i\|$
- 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 depends on the size of the true residual: $\|b - A\hat{x}_i\|$
- 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

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

and

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

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

Maximum attainable accuracy of HSCG

- In finite precision HSCG, iterates are updated by

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

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

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

accumulation of local rounding errors

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

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

accumulation of local rounding errors

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

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

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

Attainable accuracy of pipelined CG

- Pipelined CG updates x_i and r_i via:

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

Attainable accuracy of pipelined CG

- Pipelined CG updates x_i and r_i via:

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

- In finite precision:

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

Attainable accuracy of pipelined CG

- Pipelined CG updates x_i and r_i via:

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

- In finite precision:

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

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

$$= f_{i-1} - \hat{\alpha}_{i-1}(\hat{s}_{i-1} - A\hat{p}_{i-1}) + \delta r_i + A\delta x_i$$

$$= f_0 + \sum_{m=1}^{i}(\delta r_m + A\delta x_m) - G_i d_i$$

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 pipelined CG

- Pipelined CG updates x_i and r_i via:

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

- In finite precision:

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

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

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 pipelined CG

- Pipelined CG updates x_i and r_i via:

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

- In finite precision:

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

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

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 pipelined CG

- Pipelined CG updates x_i and r_i via:

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

- In finite precision:

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

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

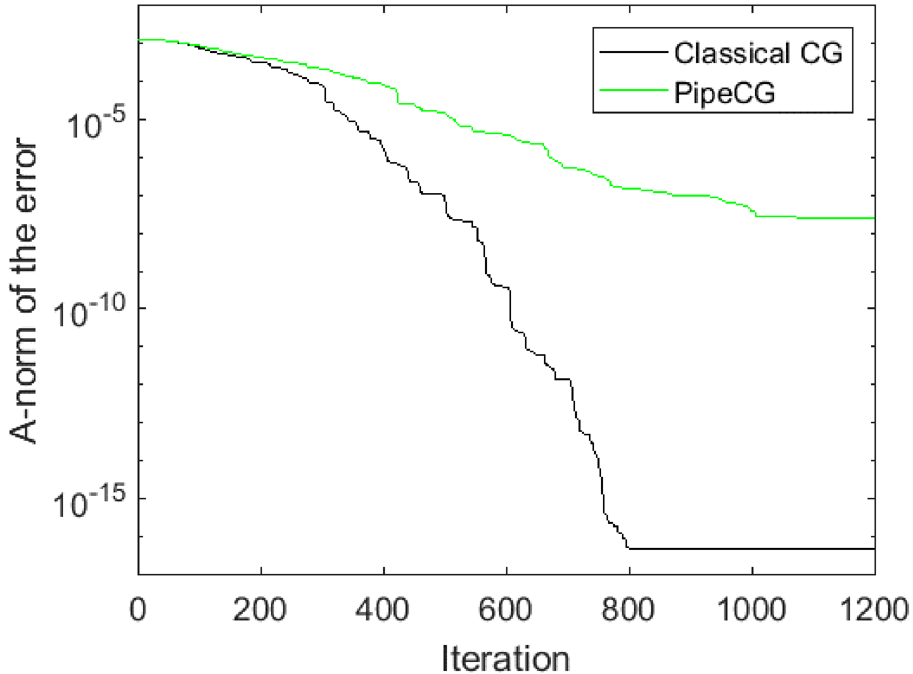
where

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

⇒ Amplification of local rounding errors possible depending on $\alpha_i's$ and $\beta_i's$

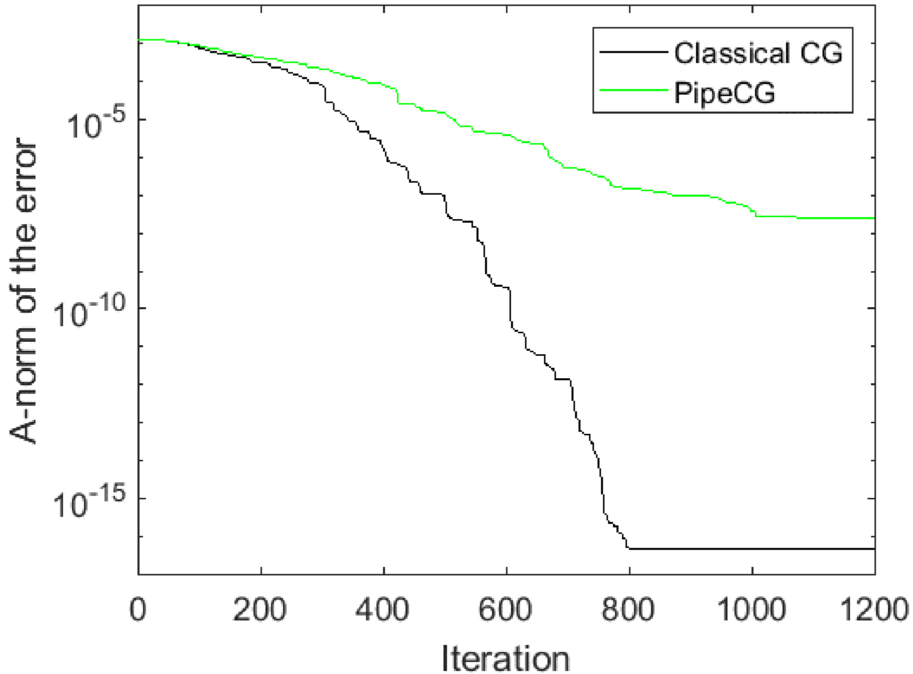
See recent work: (Cools et al., 2017), (Carson et al., 2017)

Numerical Example

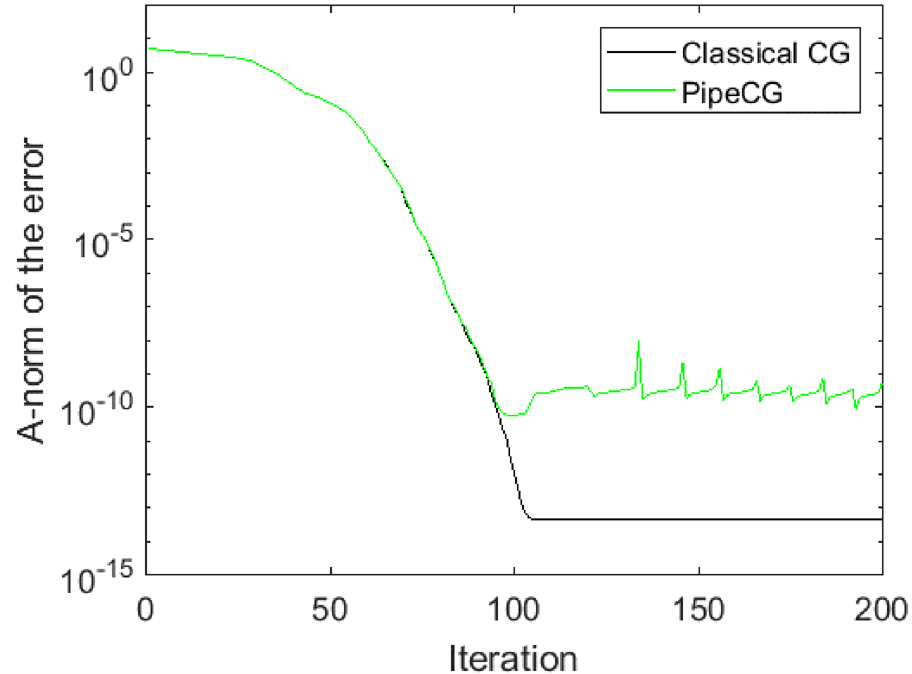


A: bcsstk03 from UFSMC, b: equal components in the eigenbasis of A and $\|b\| = 1$
$N = 112, \kappa(A) \approx 7e6$

Numerical Example



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



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

Attainable accuracy of s-step CG

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

For CG:

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

Attainable accuracy of s-step CG

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

For CG:

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

For s-step CG: $i \equiv sk + j$ (see C., 2015)

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

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

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

Attainable accuracy of s-step CG

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

For CG:

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

For s-step CG: $i \equiv sk + j$ (see C., 2015)

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

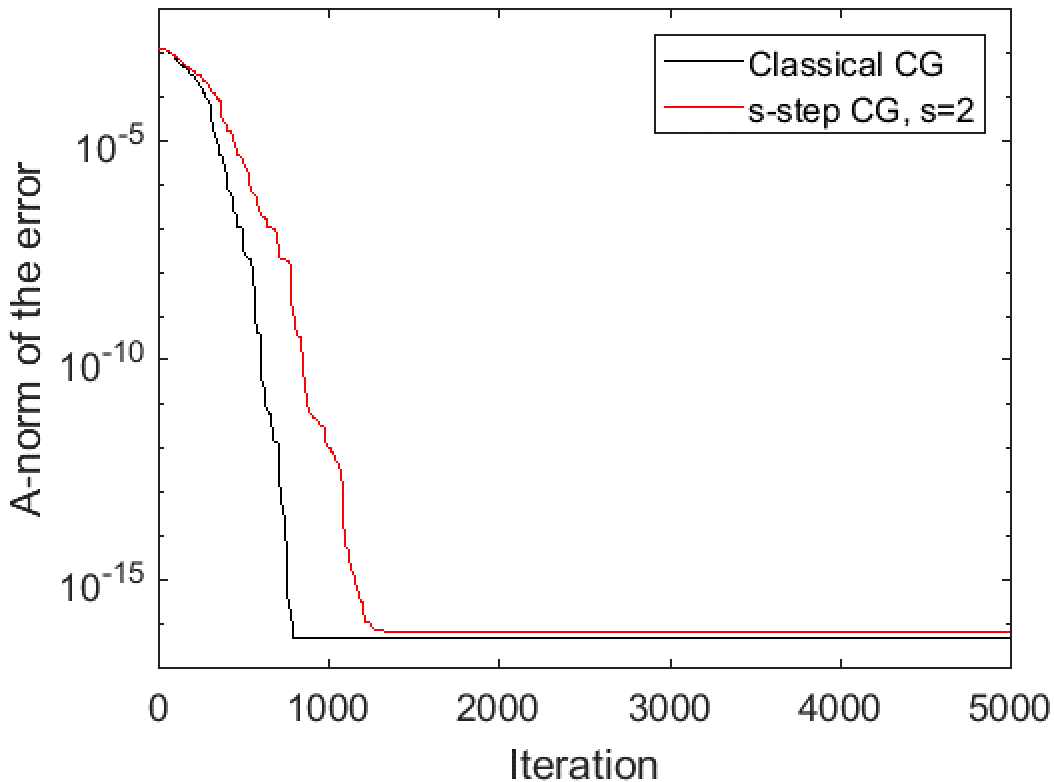
where c is a low-degree polynomial in s, and

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

⇒ Amplification of local rounding errors possible depending on conditioning of basis

Numerical example

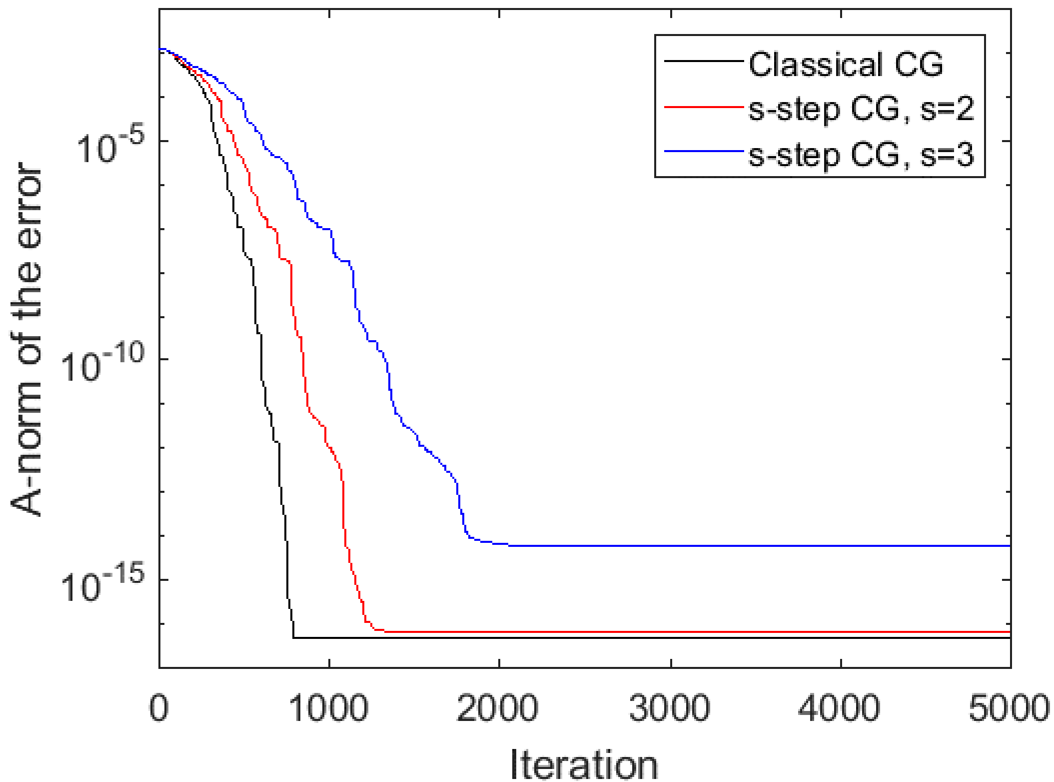
s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \dots, A^s p_i, r_i, Ar_i, \dots A^{s-1} r_i]$)



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

Numerical example

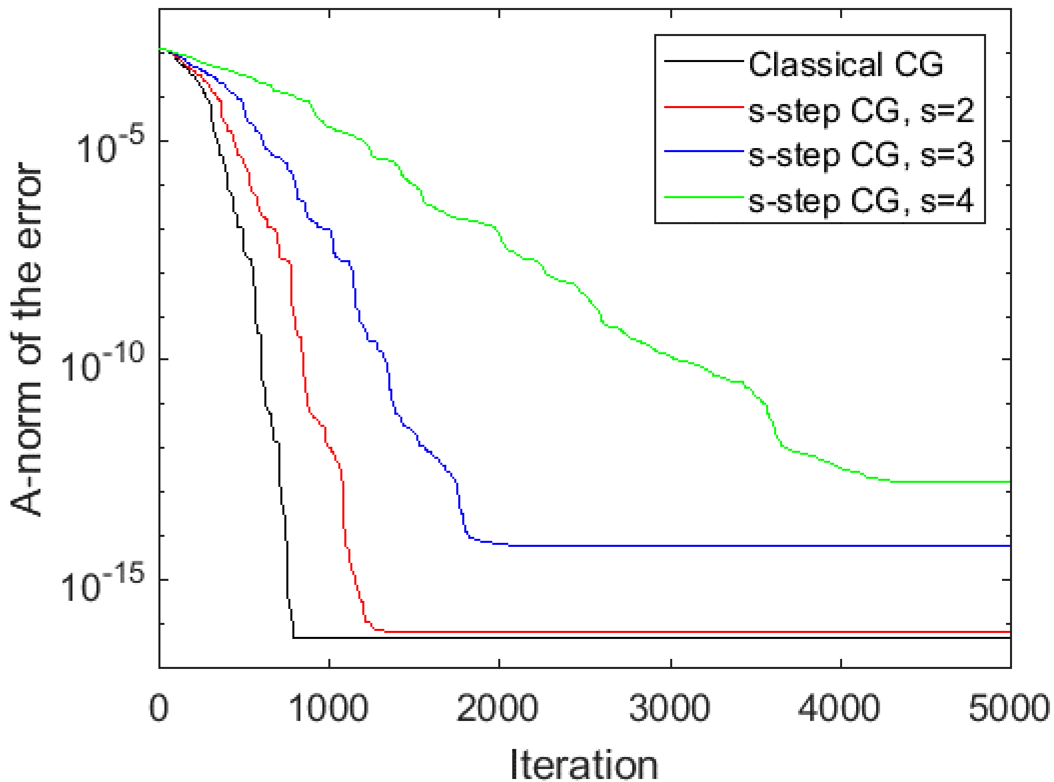
s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \dots, A^s p_i, r_i, Ar_i, \dots A^{s-1} r_i]$)



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

Numerical example

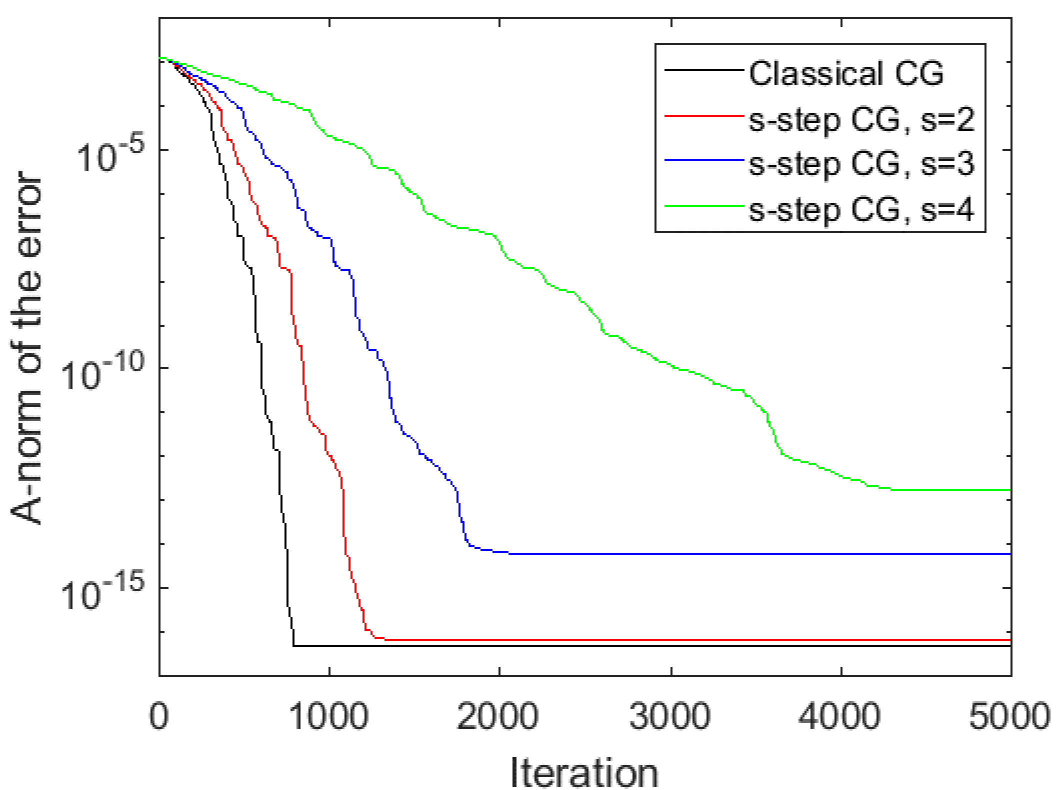
s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \dots, A^s p_i, r_i, Ar_i, \dots A^{s-1} r_i]$)



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

Numerical example

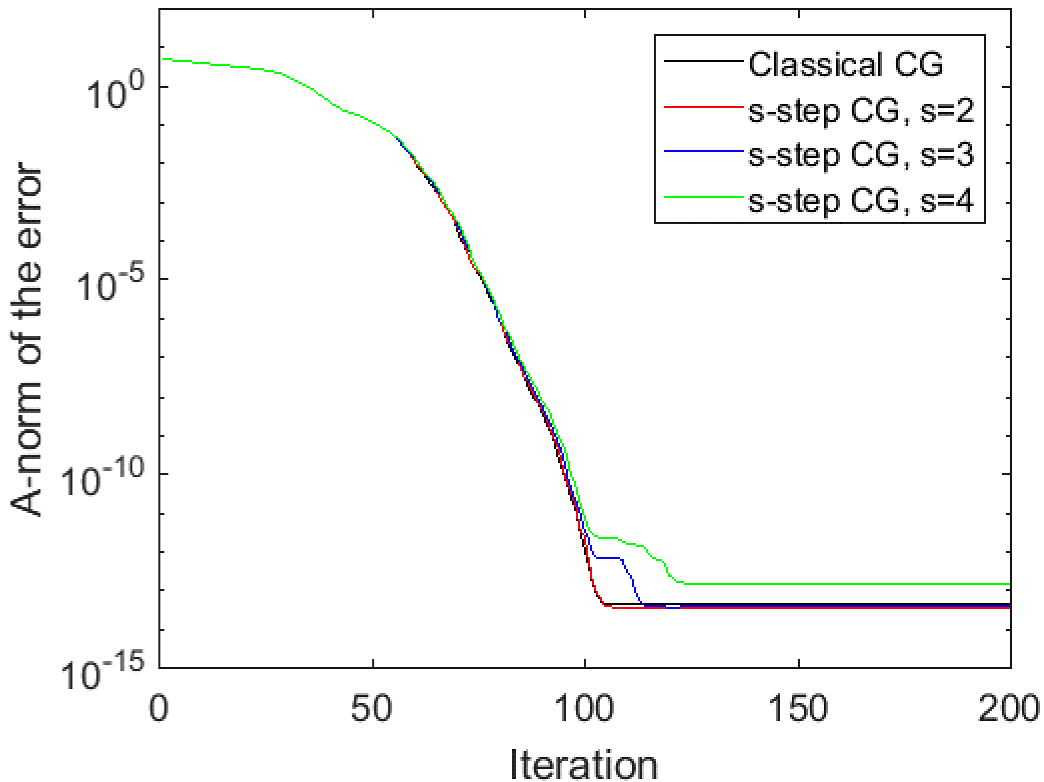
s-step CG with monomial basis ($\mathcal{Y} = [p_i, Ap_i, \dots, A^s p_i, r_i, Ar_i, \dots A^{s-1} r_i]$)



* Can also use other, more well-conditioned bases to improve convergence rate and accuracy (see, e.g. Philippe and Reichel, 2012).

Numerical example

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



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

Residual Replacement

- Idea: improve accuracy by replacing $\hat{r}_i$ with $\mathrm{fl}(b - A\hat{x}_i)$ in certain iterations (Van der Vorst and Ye, 2000)

Residual Replacement

- Idea: improve accuracy by replacing $\hat{r}_i$ with $\mathrm{fl}(b - A\hat{x}_i)$ in certain iterations (Van der Vorst and Ye, 2000)
- Choose when to replace based on estimate of $\|f_i\| \equiv \|b - A\hat{x}_i - \hat{r}_i\|$
  - Replace often enough such that $\|f_i\|$ remains small
  - But don't replace when error in computing $\mathrm{fl}(b - A\hat{x}_i)$ would perturb recurrence and cause convergence delay
    - See (Strakoš and Tichý, 2002)

Residual Replacement

- Idea: improve accuracy by replacing $\hat{r}_i$ with $\mathrm{fl}(b - A\hat{x}_i)$ in certain iterations (Van der Vorst and Ye, 2000)
- Choose when to replace based on estimate of $\|f_i\| \equiv \|b - A\hat{x}_i - \hat{r}_i\|$
  - Replace often enough such that $\|f_i\|$ remains small
  - But don't replace when error in computing $\mathrm{fl}(b - A\hat{x}_i)$ would perturb recurrence and cause convergence delay
    - See (Strakoš and Tichý, 2002)

- This strategy can be adapted for both pipelined KSMs (Cools and Vanroose, 2017) and s-step KSMs (Carson and Demmel, 2014)

Residual Replacement

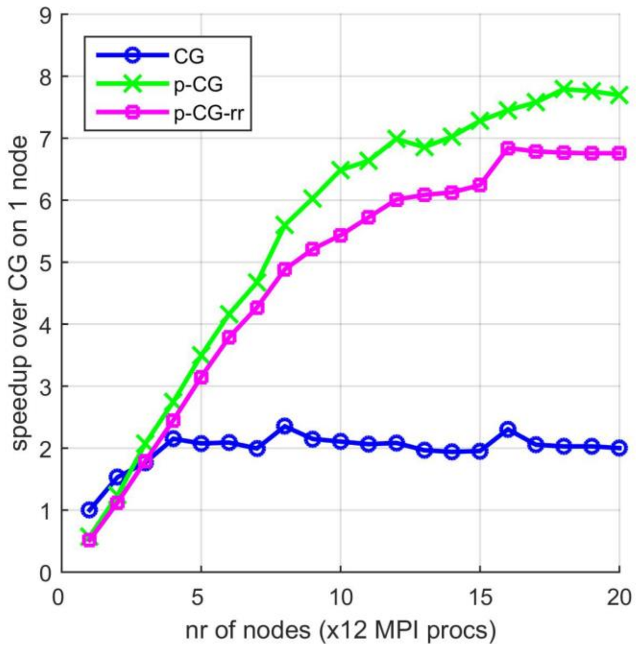
- Idea: improve accuracy by replacing $\hat{r}_i$ with $\mathrm{fl}(b - A\hat{x}_i)$ in certain iterations (Van der Vorst and Ye, 2000)
- Choose when to replace based on estimate of $\|f_i\| \equiv \|b - A\hat{x}_i - \hat{r}_i\|$
  - Replace often enough such that $\|f_i\|$ remains small
  - But don't replace when error in computing $\mathrm{fl}(b - A\hat{x}_i)$ would perturb recurrence and cause convergence delay
    - See (Strakoš and Tichý, 2002)

- This strategy can be adapted for both pipelined KSMs (Cools and Vanroose, 2017) and s-step KSMs (Carson and Demmel, 2014)
  - In both cases, estimate of $\|f_i\|$ can be computed inexpensively
  - Improves accuracy to comparable level as classical method in many cases

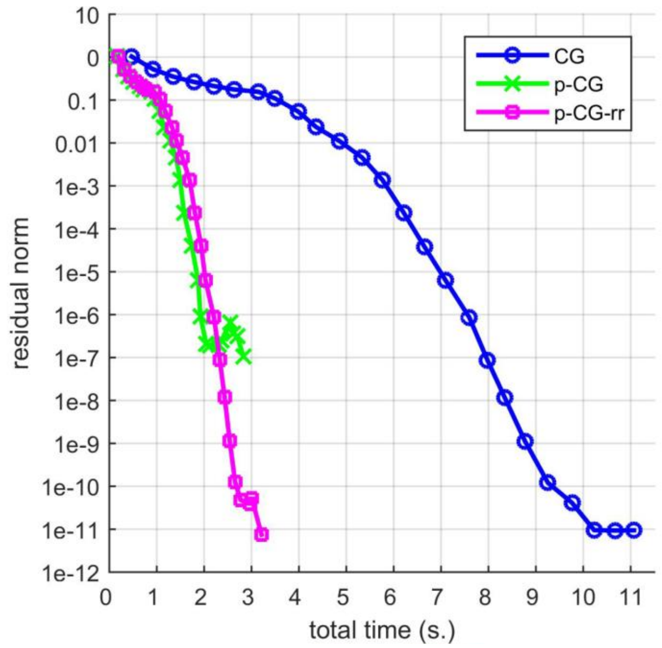
Scalability of pipelined CG with RR

- PETSc implementation using MPICH-3.1.3 communication
- Benchmark problem: 2D Laplacian model, 1,000,000 unknowns
- System specs: 20 nodes, two 6-core Intel Xeon X5660 Nehalem 2.8GHz CPUs/node

Speedup over single-node CG (12-240 cores)



Accuracy i.f.o. total time spent (240 cores)



Conclusions and takeaways

- Need new approaches to reducing synchronization cost in Krylov subspace methods for large-scale problems
  - Pipelined methods, s-step methods, hybrid approaches
  - But must also consider finite precision behavior!

Conclusions and takeaways

- Need new approaches to reducing synchronization cost in Krylov subspace methods for large-scale problems
    - Pipelined methods, s-step methods, hybrid approaches
    - But must also consider finite precision behavior!

- Other communication-avoiding and communication-hiding approaches possible
    - e.g., Enlarged Krylov subspace methods - Talk by O. Tissot, MS28 (this session)

Conclusions and takeaways

- Need new approaches to reducing synchronization cost in Krylov subspace methods for large-scale problems
  - Pipelined methods, s-step methods, hybrid approaches
  - But must also consider finite precision behavior!

- Other communication-avoiding and communication-hiding approaches possible
  - e.g., Enlarged Krylov subspace methods - Talk by O. Tissot, MS28 (this session)

- Best approach is *highly application-dependent*
  - Application of pipelined KSMs to graph partitioning - Talk by P. Ghysels in Part II: MS40

Conclusions and takeaways

- Need new approaches to reducing synchronization cost in Krylov subspace methods for large-scale problems
  - Pipelined methods, s-step methods, hybrid approaches
  - But must also consider finite precision behavior!

- Other communication-avoiding and communication-hiding approaches possible
  - e.g., Enlarged Krylov subspace methods - Talk by O. Tissot, MS28 (this session)

- Best approach is *highly application-dependent*
  - Application of pipelined KSMs to graph partitioning - Talk by P. Ghysels in Part II: MS40

- Many interesting open problems and challenges as we push toward exascale-level computing!

Thank You!

erinc@cims.nyu.edu

math.nyu.edu/~erinc