Communication-Avoiding Krylov Subspace Methods in Theory and Practice

Erin Carson, NYU

DMML Workshop
October 23, 2015
Why Avoid “Communication”? 

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

- On today’s computers, communication is expensive, computation is cheap, in terms of both time and energy!
## Future Exascale Systems

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>System Peak</td>
<td>$2 \cdot 10^{15}$ flops</td>
<td>$10^{18}$ flops</td>
<td>~1000</td>
</tr>
<tr>
<td>Node Memory Bandwidth</td>
<td>25 GB/s</td>
<td>0.4-4 TB/s</td>
<td>~10-100</td>
</tr>
<tr>
<td>Total Node Interconnect Bandwidth</td>
<td>3.5 GB/s</td>
<td>100-400 GB/s</td>
<td>~100</td>
</tr>
<tr>
<td>Memory Latency</td>
<td>100 ns</td>
<td>50 ns</td>
<td>~1</td>
</tr>
<tr>
<td>Interconnect Latency</td>
<td>1 $\mu$s</td>
<td>0.5 $\mu$s</td>
<td>~1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>System Peak</td>
<td>$2 \cdot 10^{15}$ flops</td>
<td>$10^{18}$ flops</td>
<td>~1000</td>
</tr>
<tr>
<td>Node Memory Bandwidth</td>
<td>25 GB/s</td>
<td>0.4-4 TB/s</td>
<td>~10-100</td>
</tr>
<tr>
<td>Total Node Interconnect Bandwidth</td>
<td>3.5 GB/s</td>
<td>100-400 GB/s</td>
<td>~100</td>
</tr>
<tr>
<td>Memory Latency</td>
<td>100 ns</td>
<td>50 ns</td>
<td>~1</td>
</tr>
<tr>
<td>Interconnect Latency</td>
<td>1 $\mu$s</td>
<td>0.5 $\mu$s</td>
<td>~1</td>
</tr>
</tbody>
</table>

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

- Gaps between communication/computation cost only growing larger in future systems
# Future Exascale Systems

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>System Peak</td>
<td>$2 \cdot 10^{15}$ flops</td>
<td>$10^{18}$ flops</td>
<td>~1000</td>
</tr>
<tr>
<td>Node Memory Bandwidth</td>
<td>25 GB/s</td>
<td>0.4-4 TB/s</td>
<td>~10-100</td>
</tr>
<tr>
<td>Total Node Interconnect Bandwidth</td>
<td>3.5 GB/s</td>
<td>100-400 GB/s</td>
<td>~100</td>
</tr>
<tr>
<td>Memory Latency</td>
<td>100 ns</td>
<td>50 ns</td>
<td>~1</td>
</tr>
<tr>
<td>Interconnect Latency</td>
<td>1 $\mu$s</td>
<td>0.5 $\mu$s</td>
<td>~1</td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

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

![Diagram showing the projection process](image)
Krylov Solvers: Limited by Communication

In terms of communication:
Krylov Solvers: Limited by Communication

In terms of communication:

“Add a dimension to $\mathcal{K}_m$”

→ Sparse Matrix-Vector Multiplication (SpMV)
  • Parallel: comm. vector entries w/ neighbors
  • Sequential: read $A$/vectors from slow memory
Krylov Solvers: Limited by Communication

In terms of communication:

“Add a dimension to $\mathcal{K}_m$”

$\rightarrow$ Sparse Matrix-Vector Multiplication (SpMV)
- Parallel: comm. vector entries w/ neighbors
- Sequential: read $A$/vectors from slow memory

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

$\rightarrow$ Inner products
- Parallel: global reduction (All-Reduce)
- Sequential: multiple reads/writes to slow memory
Krylov Solvers: Limited by Communication

In terms of communication:

“Add a dimension to $\mathcal{H}_m$”

→ Sparse Matrix-Vector Multiplication (SpMV)
  • Parallel: comm. vector entries w/ neighbors
  • Sequential: read $A$/vectors from slow memory

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

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

Dependencies between communication-bound kernels in each iteration limit performance!
Example: Classical Conjugate Gradient (CG)

Given: initial approximation $x_0$ for solving $Ax = b$
Let $p_0 = r_0 = b - Ax_0$
for $m = 0, 1, 2, \ldots$, until convergence do

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

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

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

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

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

end for
Example: Classical Conjugate Gradient (CG)

Given: initial approximation \( x_0 \) for solving \( Ax = b \)
Let \( p_0 = r_0 = b - Ax_0 \)
for \( m = 0, 1, 2, \ldots, \) until convergence do

\[
\alpha_m = \frac{r_m^T r_m}{p_m^T A p_m}
\]

\[
x_{m+1} = x_m + \alpha_m p_m
\]

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

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

\[
p_{m+1} = r_{m+1} + \beta_{m+1} p_m
\]

end for

SpMV
Example: Classical Conjugate Gradient (CG)

Given: initial approximation $x_0$ for solving $Ax = b$
Let $p_0 = r_0 = b - Ax_0$

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

- $\alpha_m = \frac{r_m^T r_m}{p_m^T Ap_m}$
- $x_{m+1} = x_m + \alpha_m p_m$
- $r_{m+1} = r_m - \alpha_m Ap_m$

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

end for
Communication-Avoiding KSMs

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

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

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

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

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

• Resurgence of interest in recent years due to growing problem sizes; growing relative cost of communication
Communication-Avoiding KSMs: CA-CG

• Main idea: Unroll iteration loop by a factor of $s$; split iteration loop into an outer loop and an inner loop

• Key observation: starting at some iteration $m$,

\[ x_{m+j} - x_m, \ r_{m+j}, \ p_{m+j} \in \mathcal{K}_{s+1}(A, p_m) + \mathcal{K}_s(A, r_m) \quad \text{for} \quad j \in \{0, \ldots, s\} \]
Communication-Avoiding KSMs: CA-CG

• Main idea: Unroll iteration loop by a factor of $s$; split iteration loop into an outer loop and an inner loop

• Key observation: starting at some iteration $m$,

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

Outer loop $k$: Communication step

Expand solution space $s$ dimensions at once

• Compute “basis matrix” $Y_k$ with columns spanning

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

• Requires reading $A$/communicating vectors only once
  • Using “matrix powers kernel”

Orthogonalize all at once

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

$$G_k = Y_k^T Y_k$$

• Communication cost of one global reduction
Perform $s$ iterations of updates

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

$$x_{sk+j} - x_{sk} = Y_k x'_j, \quad r_{sk+j} = Y_k r'_j, \quad p_{sk+j} = Y_k p'_j$$
Inner loop: Computation steps, no communication!

Perform $s$ iterations of updates
- Using $Y_k$ and $G_k$, this requires no communication!
- Represent $n$-vectors by their $O(s)$ coordinates in $Y_k$:
  \[
  x_{sk+j} - x_{sk} = Y_k x'_j, \quad r_{sk+j} = Y_k r'_j, \quad p_{sk+j} = Y_k p'_j
  \]

Communication-Avoiding KSMs: CA-CG
**Communication-Avoiding KSMs: CA-CG**

Inner loop:
Computation steps, no communication!

Perform $s$ iterations of updates

- Using $Y_k$ and $G_k$, this requires **no communication**!
- Represent $n$-vectors by their $O(s)$ coordinates in $Y_k$:
  
  $x_{sk+j} - x_{sk} = Y_kx'_j$, $r_{sk+j} = Y_kr'_j$, $p_{sk+j} = Y_kp'_j$

\[
\begin{align*}
Aq_{sk+j} & \quad \rightarrow \\
B_{k,p_{k,j}} & \\
\end{align*}
\]

\[
\begin{array}{c}
o(s) \quad \times \\
o(s) \quad \times \\
\end{array}
\]
Perform $s$ iterations of updates

- Using $Y_k$ and $G_k$, this requires no communication!
- Represent $n$-vectors by their $O(s)$ coordinates in $Y_k$:
  \[ x_{sk+j} - x_{sk} = Y_k x'_j, \quad r_{sk+j} = Y_k r'_j, \quad p_{sk+j} = Y_k p'_j \]

Inner loop:
Computation steps, no communication!

```latex
\begin{align*}
A p_{sk+j} & \quad B_{k} p'_{k,j} \\
\end{align*}
```
**Communication-Avoiding KSMs: CA-CG**

**Inner loop:**
Computations steps, no communication!

Perform $s$ iterations of updates
- Using $Y_k$ and $G_k$, this requires **no communication**!
- Represent $n$-vectors by their $O(s)$ coordinates in $Y_k$:
  
  $$x_{sk+j} - x_{sk} = Y_k x'_j, \quad r_{sk+j} = Y_k r'_j, \quad p_{sk+j} = Y_k p'_j$$

**Computation steps, no communication!**
Example: CA-Conjugate Gradient

Given: initial approximation \( x_0 \) for solving \( Ax = b \)
Let \( p_0 = r_0 = b - Ax_0 \)
for \( k = 0, 1, \ldots \), until convergence do
  Compute \( Y_k \), compute \( G_k = Y_k^T Y_k \)
  Let \( x'_0 = 0_{2s+1} \), \( r'_0 = e_{s+2} \), \( p'_0 = e_1 \)
for \( j = 0, \ldots, s - 1 \) do
  \[ \alpha_{sk+j} = \frac{(r'_j)^T G_k r'_j}{(p'_j)^T G_k B_k p'_j} \]
  \[ x_{j+1}' = x'_j + \alpha_{sk+j} p'_j \]
  \[ r_{j+1}' = r'_j - \alpha_{sk+j} B_k p'_j \]
  \[ \beta_{sk+j+1} = \frac{(r_{j+1}')^T G_k r_{j+1}'}{(r'_j)^T G_k r'_j} \]
  \[ p_{j+1}' = r_{j+1}' + \beta_{sk+j+1} p'_j \]
end for
Compute \( x_{sk+s} = Y_k x'_s + x_{sk} \), \( r_{sk+s} = Y_k r'_s \), \( p_{sk+s} = Y_k p'_s \)
end for
Example: CA-Conjugate Gradient

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

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

$$\beta_{sk+j+1} = \frac{(r'_{j+1})^T G_k r'_{j+1}}{(r'_j)^T G_k r'_j}$$
$$p'_{j+1} = r'_{j+1} + \beta_{sk+j+1} p'_j$$
end for
Compute $x_{sk+s} = Y_k x'_s + x_{sk}$, $r_{sk+s} = Y_k r'_s$, $p_{sk+s} = Y_k p'_s$
end for
Example: CA-Conjugate Gradient

Given: initial approximation $x_0$ for solving $Ax = b$
Let $p_0 = r_0 = b - Ax_0$
for $k = 0, 1, \ldots$, until convergence do
  Compute $Y_k$, compute $G_k = Y_k^T Y_k$
  Let $x'_0 = 0_{2s+1}$, $r'_0 = e_{s+2}$, $p'_0 = e_1$
  for $j = 0, \ldots, s - 1$ do
    $\alpha_{sk+j} = \frac{(r'_j)^T G_k r'_j}{(p'_j)^T G_k B_k p'_j}$
    $x'_{j+1} = x'_j + \alpha_{sk+j} p'_j$
    $r'_{j+1} = r'_j - \alpha_{sk+j} B_k p'_j$
    $\beta_{sk+j+1} = \frac{(r'_{j+1})^T G_k r'_{j+1}}{(r'_j)^T G_k r'_j}$
    $p'_{j+1} = r'_{j+1} + \beta_{sk+j+1} p'_j$
  end for
Compute $x_{sk+s} = Y_k x'_s + x_{sk}$, $r_{sk+s} = Y_k r'_s$, $p_{sk+s} = Y_k p'_s$
end for
Example of parallel (per processor) complexity for $s$ iterations of CG vs. CA-CG for a 2D 9-point stencil:

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

<table>
<thead>
<tr>
<th></th>
<th>Flops</th>
<th>Words Moved</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>$sn/p$</td>
<td>$sn/p$</td>
<td>$s\sqrt{n/p}$</td>
</tr>
<tr>
<td>CG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA-CG</td>
<td>$sn/p$</td>
<td>$s^2 n/p$</td>
<td>$s\sqrt{n/p}$</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th></th>
<th>Flops</th>
<th>Words Moved</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical CG</td>
<td>$sn/p$</td>
<td>$sn/p$</td>
<td>$s\sqrt{n/p}$</td>
</tr>
<tr>
<td>CA-CG</td>
<td>$sn/p$</td>
<td>$s^2n/p$</td>
<td>$s\sqrt{n/p}$</td>
</tr>
</tbody>
</table>

All values in the table meant in the Big-O sense (i.e., lower order terms and constants not included)
Example of parallel (per processor) complexity for $s$ iterations of CG vs. CA-CG for a 2D 9-point stencil:
(Assuming each of $p$ processors owns $n/p$ rows of the matrix and $s \leq \sqrt{n/p}$)

<table>
<thead>
<tr>
<th></th>
<th>Flops</th>
<th>Words Moved</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical CG</td>
<td>$\frac{sn}{p}$</td>
<td>$\frac{sn}{p}$</td>
<td>$s\sqrt{n/p}$</td>
</tr>
<tr>
<td>CA-CG</td>
<td>$\frac{sn}{p}$</td>
<td>$\frac{s^2 n}{p}$</td>
<td>$s\sqrt{n/p}$</td>
</tr>
</tbody>
</table>

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

• Parameter $s$ is limited by machine parameters and matrix sparsity structure

• We can auto-tune to find the best $s$ based on these properties
  • That is, find $s$ that gives the fastest speed per iteration
From Theory to Practice

• Parameter $s$ is limited by machine parameters and matrix sparsity structure

• We can auto-tune to find the best $s$ based on these properties
  • That is, find $s$ that gives the fastest speed per iteration

• In practice, we don’t just care about speed per iteration, but also the number of iterations
  
  Runtime = (time/iteration) x (# iterations)
From Theory to Practice

• Parameter $s$ is limited by machine parameters and matrix sparsity structure

• We can auto-tune to find the best $s$ based on these properties
  • That is, find $s$ that gives the fastest *speed per iteration*

• In practice, we don’t just care about speed per iteration, but also the number of iterations
  
  \[
  \text{Runtime} = (\text{time/iteration}) \times (\# \text{ iterations})
  \]

• We also need to consider how convergence rate and accuracy are affected by choice of $s$!
From Theory to Practice

• CA-KSMs are mathematically equivalent to classical KSMs
From Theory to Practice

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

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

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

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

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

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

Runtime = (time/iteration) x (# iterations)
From Theory to Practice

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

Runtime = (time/iteration) x (# iterations)
Model Problem: 2D Poisson (5-pt stencil),

\[ n = 512^2, \text{nnz} \approx 10^6, \kappa(A) \approx 10^4 \]

\[ b = A(1\sqrt{n} \cdot \text{ones}(n, 1)) \]
Model Problem: 2D Poisson (5-pt stencil),

\[ n = 512^2, \text{nnz} \approx 10^6, \kappa(A) \approx 10^4 \]

\[ b = A(1\sqrt{n} \cdot \text{ones}(n, 1)) \]
Model Problem: 2D Poisson (5-pt stencil),
\[ n = 512^2, \text{nnz} \approx 10^6, \kappa(A) \approx 10^4 \]
\[ b = A(1\sqrt{n} \cdot \text{ones}(n,1)) \]
Model Problem: 2D Poisson (5-pt stencil), \( n = 512^2, \text{nnz} \approx 10^6, \kappa(A) \approx 10^4 \)
\[
b = A(1/\sqrt{n} \cdot \text{ones}(n, 1))
\]
Model Problem: 2D Poisson (5-pt stencil),
\[ n = 512^2, \text{nnz} \approx 10^6, \kappa(A) \approx 10^4 \]
\[ b = A(1/\sqrt{n} \cdot \text{ones}(n, 1)) \]
Model Problem: 2D Poisson (5-pt stencil),
\[ n = 512^2, \text{nnz} \approx 10^6, \kappa(A) \approx 10^4 \]
\[ b = A(1\sqrt{n} \cdot \text{ones}(n, 1)) \]
Model Problem: 2D Poisson (5-pt stencil),
\[ n = 512^2, \text{nnz} \approx 10^6, \kappa(A) \approx 10^4 \]
\[ b = A(1\sqrt{n} \cdot \text{ones}(n, 1)) \]
Better basis choice allows higher s values

But can still see loss of accuracy/convergence delay

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

\[ b = A(1\sqrt{n} \cdot \text{ones}(n, 1)) \]
Maximum attainable accuracy of CG

- In classical CG, iterates are updated by
  \[ x_{m+1} = x_m + \alpha_m p_m \quad \text{and} \quad r_{m+1} = r_m - \alpha_m A p_m \]

- Formulas for \( x_{m+1} \) and \( r_{m+1} \) do not depend on each other - rounding errors cause the true residual, \( b - A x_{m+1} \), and the updated residual, \( r_{m+1} \), to deviate
Maximum attainable accuracy of CG

- In classical CG, iterates are updated by
  \[ x_{m+1} = x_m + \alpha_m p_m \quad \text{and} \quad r_{m+1} = r_m - \alpha_m A p_m \]

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

- The size of the true residual is bounded by
  \[
  \| b - Ax_{m+1} \| \leq \| r_{m+1} \| + \| b - Ax_{m+1} - r_{m+1} \| 
  \]

- When \( \| r_{m+1} \| \gg \| b - Ax_{m+1} - r_{m+1} \| \), \( \| r_{m+1} \| \) and \( \| b - Ax_{m+1} \| \) have similar magnitude.
- When \( \| r_{m+1} \| \rightarrow 0 \), \( \| b - Ax_{m+1} \| \) depends on \( \| b - Ax_{m+1} - r_{m+1} \| \).
Maximum attainable accuracy of CG

- In classical CG, iterates are updated by
  \[ x_{m+1} = x_m + \alpha_m p_m \quad \text{and} \quad r_{m+1} = r_m - \alpha_m A p_m \]

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

- The size of the true residual is bounded by
  \[ \| b - Ax_{m+1} \| \leq \| r_{m+1} \| + \| b - Ax_{m+1} - r_{m+1} \| \]

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


- We have applied a similar analysis to upper bound the maximum attainable accuracy in finite precision CA-KSMs
Residual Replacement Strategy for CG

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

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

• Choose when to replace $r_{m+1}$ with $b - Ax_{m+1}$ to meet two constraints:
Residual Replacement Strategy for CG

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

• Choose when to replace $r_{m+1}$ with $b - Ax_{m+1}$ to meet two constraints:

  1. **Replace often enough** so that at termination, $\|b - Ax_{m+1} - r_{m+1}\|$ is small relative to $\varepsilon N \|A\| \|x_{m+1}\|$
Residual Replacement Strategy for CG

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

• Choose when to replace $r_{m+1}$ with $b - Ax_{m+1}$ to meet two constraints:

  1. **Replace often enough** so that at termination, $\|b - Ax_{m+1} - r_{m+1}\|$ is small relative to $\epsilon N \|A\||x_{m+1}\|

  2. **Don’t replace so often** that original convergence mechanism of updated residuals is destroyed (avoid large perturbations to finite precision CG recurrence)
Residual Replacement Strategy for CG

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

• Choose when to replace $r_{m+1}$ with $b - Ax_{m+1}$ to meet two constraints:

  1. Replace often enough so that at termination, $\|b - Ax_{m+1} - r_{m+1}\|$ is small relative to $\varepsilon N \|A\| \|x_{m+1}\|$  

  2. Don’t replace so often that original convergence mechanism of updated residuals is destroyed (avoid large perturbations to finite precision CG recurrence)

• We can implement an analogous strategy for CA-CG and CA-BICG based on derived bound on deviation of residuals

  • Estimating quantities in bound has negligible cost → residual replacement strategy does not asymptotically increase communication or computation!
Model Problem: 2D Poisson (5-pt stencil),
\[ n = 512^2, \text{nnz} \approx 10^6, \kappa(A) \approx 10^4 \]
\[ b = A(1 \sqrt{n} \cdot \text{ones}(n, 1)) \]
Model Problem: 2D Poisson (5-pt stencil),
\( n = 512^2 \), \( nnz \approx 10^6 \), \( \kappa(A) \approx 10^4 \)
\[ b = A(1\sqrt{n} \cdot \text{ones}(n, 1)) \]
Model Problem: 2D Poisson (5-pt stencil),
\( n = 512^2, \; \text{nnz} \approx 10^6, \; \kappa(A) \approx 10^4 \)
\( b = A(1\sqrt{n} \cdot \text{ones}(n, 1)) \)
Model Problem: 2D Poisson (5-pt stencil), $n = 512^2$, $\text{nnz} \approx 10^6$, $\kappa(A) \approx 10^4$

\[ b = A(1/\sqrt{n} \cdot \text{ones}(n, 1)) \]
Paige’s Results for Classical Lanczos

- Using bounds on local rounding errors in Lanczos, Paige showed that
  1. The computed Ritz values always lie between the extreme eigenvalues of $A$ to within a small multiple of machine precision.
  2. At least one small interval containing an eigenvalue of $A$ is found by the $n$th iteration.
  3. The algorithm behaves numerically like Lanczos with full reorthogonalization until a very close eigenvalue approximation is found.
  4. The loss of orthogonality among basis vectors follows a rigorous pattern and implies that some Ritz values have converged.
Paige’s Results for Classical Lanczos

• Using bounds on local rounding errors in Lanczos, Paige showed that
  1. The computed Ritz values always lie between the extreme eigenvalues of $A$ to within a small multiple of machine precision.
  2. At least one small interval containing an eigenvalue of $A$ is found by the $n$th iteration.
  3. The algorithm behaves numerically like Lanczos with full reorthogonalization until a very close eigenvalue approximation is found.
  4. The loss of orthogonality among basis vectors follows a rigorous pattern and implies that some Ritz values have converged.

Do the same statements hold for CA-Lanczos?
Paige’s Lanczos Convergence Analysis

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

\[ A\hat{V}_m = \hat{V}_m \hat{T}_m + \hat{\beta}_{m+1} \hat{v}_{m+1} e^T_m + \delta \hat{V}_m \]

\[ \hat{V}_m = [\hat{v}_1, \ldots, \hat{v}_m], \quad \delta \hat{V}_m = [\delta \hat{v}_1, \ldots, \delta \hat{v}_m], \quad \hat{T}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 \\ \hat{\beta}_2 & \ddots & \ddots \\ & \ddots & \ddots & \hat{\beta}_m \\ & & \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}. \]
Paige’s Lanczos Convergence Analysis

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

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

\[
\hat{V}_m = [\hat{v}_1, ..., \hat{v}_m], \quad \delta\hat{V}_m = [\delta\hat{v}_1, ..., \delta\hat{v}_m], \quad \hat{r}_m = \begin{bmatrix}
\hat{\alpha}_1 & \hat{\beta}_2 & \vdots \\
\hat{\beta}_2 & \ddots & \ddots \\
\vdots & \ddots & \ddots \\
\hat{\beta}_m & \ddots & \ddots \\
\hat{\beta}_m & \ddots & \ddots \\
\hat{\alpha}_m & \ddots & \ddots \\
\end{bmatrix}
\]

Classic Lanczos rounding error result of Paige (1976):

\[
\text{for } i \in \{1, ..., m\},
\begin{align*}
\|\delta\hat{v}_i\|_2 & \leq \varepsilon_1\sigma \\
|\hat{\beta}_{i+1}\hat{v}_i^T\hat{v}_{i+1}| & \leq 2\varepsilon_0\sigma \\
|\hat{v}_{i+1}^T\hat{v}_{i+1} - 1| & \leq \varepsilon_0/2 \\
\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A\hat{v}_i\|_2^2 & \leq 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2
\end{align*}
\]

where \(\sigma \equiv \|A\|_2\), \(\theta\sigma \equiv \|||A||_2\) , \(\varepsilon_0 \equiv 2\varepsilon(n + 4)\), and \(\varepsilon_1 \equiv 2\varepsilon(N\theta + 7)\)
Paige’s Lanczos Convergence Analysis

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

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

\(\hat{V}_m = [\hat{v}_1, \ldots, \hat{v}_m],\) \(\delta \hat{V}_m = [\delta \hat{v}_1, \ldots, \delta \hat{v}_m],\) \(\hat{t}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & \cdots & \cdots & \cdots & \cdots & \hat{\beta}_m \\ \hat{\beta}_2 & \ddots & \ddots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \cdots \\ \hat{\beta}_m & \cdots & \cdots & \cdots & \cdots & \cdots & \hat{\alpha}_m \end{bmatrix}
\)

Classic Lanczos rounding error result of Paige (1976):

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

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

\(\varepsilon_0 = O(\varepsilon n)\) \quad \varepsilon_1 = O(\varepsilon N \theta)\)
Paige’s Lanczos Convergence Analysis

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

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

$$\hat{V}_m = [\hat{v}_1, ..., \hat{v}_m], \quad \delta \hat{V}_m = [\delta \hat{v}_1, ..., \delta \hat{v}_m], \quad \hat{r}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & \vdots \\ \hat{\beta}_2 & \ddots & \vdots \\ \vdots & \ddots & \hat{\beta}_m \\ \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}$$

Classic Lanczos rounding error result of Paige (1976):

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

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

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

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

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

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

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

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

→ These results form the basis for Paige’s influential results in (Paige, 1980).
CA-Lanczos Convergence Analysis

For CA-Lanczos, we have:

\[
\begin{align*}
\text{for } i \in \{1, \ldots, m=sk+j\}, \\
\|\delta \hat{v}_i\|_2 & \leq \varepsilon_1 \sigma \\
\hat{\beta}_{i+1} |\hat{v}_i^T \hat{v}_{i+1}| & \leq 2\varepsilon_0 \sigma \\
|\hat{v}_{i+1}^T \hat{v}_{i+1} - 1| & \leq \varepsilon_0 / 2 \\
|\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A \hat{v}_i\|_2^2| & \leq 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2
\end{align*}
\]

\[
\varepsilon_0 \equiv 2\varepsilon(n+11s+15) \Gamma^2 = O(\varepsilon n \Gamma^2),
\]

\[
\varepsilon_1 \equiv 2\varepsilon(N+2s+5)\theta + (4s+9)\tau + 10s+16)\Gamma = O(\varepsilon N \theta \Gamma),
\]

where \( \sigma \equiv \|A\|_2, \quad \theta \sigma \equiv |||A|||_2, \quad \tau \sigma \equiv \max_{\ell \leq k} \|||B_\ell|||_2, \) and

\[
\Gamma \leq \max_{\ell \leq k} \|Y_\ell^+\|_2 \cdot \|||Y_\ell|||_2 \leq (2s+1) \cdot \max_{\ell \leq k} \kappa(Y_\ell).
\]
CA-Lanczos Convergence Analysis

For CA-Lanczos, we have:

\[
\|\delta \hat{v}_i\|_2 \leq \varepsilon_1 \sigma \\
|\hat{\beta}_{i+1} \hat{\nu}_i^T \hat{\nu}_{i+1}| \leq 2\varepsilon_0 \sigma \\
|\hat{\nu}_{i+1}^T \hat{\nu}_{i+1} - 1| \leq \varepsilon_0 / 2 \\
|\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A \hat{\nu}_i\|_2^2| \leq 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2
\]

\(\varepsilon_0 \equiv 2\varepsilon(n+11s+15)\Gamma^2 = O(\varepsilon n \Gamma^2),\) \hspace{1cm} (vs. \(O(\varepsilon n)\) for Lanczos)

\(\varepsilon_1 \equiv 2\varepsilon((N+2s+5)\theta + (4s+9)\tau + 10s+16)\Gamma = O(\varepsilon N\theta \Gamma),\) \hspace{1cm} (vs. \(O(\varepsilon N\theta)\) for Lanczos)

where \(\sigma \equiv \|A\|_2, \ \theta \sigma \equiv |||A|||_2, \ \tau \sigma \equiv \max_{\ell \leq k} \|||B_\ell|||_2,\) and

\[\Gamma \leq \max_{\ell \leq k} \|Y_\ell^+\|_2 \cdot \|||Y_\ell|||_2 \leq (2s+1) \cdot \max_{\ell \leq k} \kappa(Y_\ell).\]
The Amplification Term $\Gamma$

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

- A loose bound for the amplification term:
  \[
  \Gamma \leq \max_{\ell \leq k} \| y^+_\ell \|_2 \cdot \| y_\ell \|_2 \leq (2s+1) \cdot \max_{\ell \leq k} \kappa(y_\ell)
  \]

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

- **Tighter bound on $\Gamma$ possible**; requires some light bookkeeping

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

  \[
  \Gamma_{k,j} \equiv \max_{x \in \{\hat{w}_{k,j}, \hat{u}_{k,j}, \hat{v}_{k,j}, \hat{v}_{k,j-1}\}} \frac{\| |\hat{Y}_k| x\|_2}{\| \hat{Y}_k x \|_2}
  \]
Results for CA-Lanczos

• Back to our question: Do Paige’s results, e.g., loss of orthogonality → eigenvalue convergence hold for CA-Lanczos?
Results for CA-Lanczos

• Back to our question: Do Paige’s results, e.g.,
  loss of orthogonality $\rightarrow$ eigenvalue convergence
  hold for CA-Lanczos?

• The answer is YES! ...but
Results for CA-Lanczos

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

• Back to our question: Do Paige’s results, e.g., loss of orthogonality → eigenvalue convergence hold for CA-Lanczos?
• The answer is YES! ...but
• Only if:
  • $\epsilon_0 \equiv 2\epsilon(n+11s+15) \Gamma^2 \leq \frac{1}{12}$
  • i.e., $\Gamma \leq \left(24\epsilon(n + 11s + 15)\right)^{-1/2} = O(n\epsilon)^{-1/2}$
  • Otherwise, e.g., can lose orthogonality due to computation with (numerically) rank-deficient basis

• Take-away: we can use this bound on $\Gamma$ to design a better algorithm!
  • Mixed precision, selective reorthogonalization, dynamic basis size, etc.
Extending the results of Greenbaum (1989):

Eigenvalue approximations generated at each step by a perturbed Lanczos recurrence for \( A \) are equal to those generated by exact Lanczos applied to a matrices whose eigenvalues lie within intervals about the eigenvalues of \( A \).
Extending the results of Greenbaum (1989):

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

\[ \lambda \]
Extending the results of Greenbaum (1989):

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

\[
\lambda \quad O(\varepsilon n^3 \|A\|)
\]
Extending the results of Greenbaum (1989):

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

\begin{align*}
\text{Classical Lanczos} & \quad O(\epsilon n^3 \|A\|) \\
\lambda & \quad O(\epsilon n^3 \|A\|^{\Gamma^2}) \\
\text{CA-Lanczos} & 
\end{align*}
Extending the results of Greenbaum (1989):

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

\begin{align*}
\text{Classical Lanczos} & \quad O(\epsilon n^3 \|A\|) \\
\lambda & \\
\text{CA-Lanczos} & \quad O(\epsilon n^3 \|A\| \Gamma^2)
\end{align*}

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

Problem specifics:
$Lu = (a\alpha - b\nabla \cdot \beta\nabla)u = f$
\[ \alpha = \beta = 1.0, \quad a = b = 0.9 \]
• Periodic boundary conds.
• RHS: 3D triangle wave w/period spanning entire domain
• Timing for coarse grid solves in geometric multigrid method
• 3D Helmholtz equation with $n = 1.6 \cdot 10^6$
• 24K cores on NERSC’s Hopper (Cray XE6)

Problem specifics:

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

- Periodic boundary conditions.
- RHS: 3D triangle wave with period spanning entire domain

4.2x speedup in Krylov solve!
Future Directions

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

- **New Algorithms/Applications**
  - Application of communication-avoiding ideas and solvers to new computational science domains
  - Design of new high-performance preconditioners

- **Finite-Precision Analysis**
  - Bounds on stability and convergence for other Krylov methods (particularly in the nonsymmetric case)
  - Extension of “Backwards-like” error analyses

- **Improving Usability**
  - Automating parameter selection via “numerical auto-tuning”
  - Integration into high-performance libraries
Thank you!

Happy Birthday, Jim!

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