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
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

• 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

Nothing tends so much to the advancement of knowledge as the application of a new instrument.

- Sir Humphry Davy
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

• 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

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

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

<table>
<thead>
<tr>
<th></th>
<th>Today's Systems</th>
<th>Predicted Exascale Systems*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Peak</strong></td>
<td>$10^{16}$ flops/s</td>
<td>$10^{18}$ flops/s</td>
</tr>
<tr>
<td><strong>Node Memory Bandwidth</strong></td>
<td>$10^2$ GB/s</td>
<td>$10^3$ GB/s</td>
</tr>
<tr>
<td><strong>Interconnect Bandwidth</strong></td>
<td>$10^1$ GB/s</td>
<td>$10^2$ GB/s</td>
</tr>
<tr>
<td><strong>Memory Latency</strong></td>
<td>$10^{-7}$ s</td>
<td>$5 \cdot 10^{-8}$ s</td>
</tr>
<tr>
<td><strong>Interconnect Latency</strong></td>
<td>$10^{-6}$ s</td>
<td>$5 \cdot 10^{-7}$ s</td>
</tr>
</tbody>
</table>

![Diagram showing CPU-Cache-DRAM connections](image-url)
# Exascale System Projections

<table>
<thead>
<tr>
<th></th>
<th>Today's Systems</th>
<th>Predicted Exascale Systems*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Peak</strong></td>
<td>$10^{16}$ flops/s</td>
<td>$10^{18}$ flops/s</td>
</tr>
<tr>
<td><strong>Node Memory Bandwidth</strong></td>
<td>$10^2$ GB/s</td>
<td>$10^3$ GB/s</td>
</tr>
<tr>
<td><strong>Interconnect Bandwidth</strong></td>
<td>$10^1$ GB/s</td>
<td>$10^2$ GB/s</td>
</tr>
<tr>
<td><strong>Memory Latency</strong></td>
<td>$10^{-7}$ s</td>
<td>$5 \cdot 10^{-8}$ s</td>
</tr>
<tr>
<td><strong>Interconnect Latency</strong></td>
<td>$10^{-6}$ s</td>
<td>$5 \cdot 10^{-7}$ s</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Today's Systems</th>
<th>Predicted Exascale Systems*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Peak</strong></td>
<td>$10^{16}$ flops/s</td>
<td>$10^{18}$ flops/s</td>
</tr>
<tr>
<td><strong>Node Memory Bandwidth</strong></td>
<td>$10^2$ GB/s</td>
<td>$10^3$ GB/s</td>
</tr>
<tr>
<td><strong>Interconnect Bandwidth</strong></td>
<td>$10^1$ GB/s</td>
<td>$10^2$ GB/s</td>
</tr>
<tr>
<td><strong>Memory Latency</strong></td>
<td>$10^{-7}$ s</td>
<td>$5 \cdot 10^{-8}$ s</td>
</tr>
<tr>
<td><strong>Interconnect Latency</strong></td>
<td>$10^{-6}$ s</td>
<td>$5 \cdot 10^{-7}$ s</td>
</tr>
</tbody>
</table>

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

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

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

### Today's Systems vs. Predicted Exascale Systems*

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

*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

![Diagram of the iterative solver process]

1. Initial guess
2. Convergence to sufficient accuracy?
   - Yes: Return solution
   - No: Refine Solution

3. Yes: Return solution
4. No: Refine Solution

Initial guess

Convergence to sufficient accuracy?

Refine Solution

Yes

Return solution

No
**Krylov Subspace Method**: projection process onto the Krylov subspace

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

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

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

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 \( C_i \))
- **Linear systems**: Select approximate solution
  \[ x_i \in x_0 + \mathcal{K}_i(A, r_0) \]
  using \( r_i = b - Ax_i \perp C_i \)
**Krylov Subspace Method**: projection process onto the Krylov subspace

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

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

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

\[ r_i \perp \mathcal{K}_i(A, r_0) \iff \|x - x_i\|_A = \min_{z \in x_0 + \mathcal{K}_i(A, r_0)} \|x - z\|_A \quad \Rightarrow \quad r_{N+1} = 0 \]
Krylov Subspace Methods in the Wild

- Climate Modeling
- Computer Vision
- Chemical Engineering
- Medical Treatment
- Computational Cosmology
- Power Grid Modeling
- Financial Portfolio Optimization
- Latent Semantic Analysis
### Summit - IBM Power System AC922

<table>
<thead>
<tr>
<th>Site</th>
<th>Oak Ridge National Laboratory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer:</td>
<td>IBM</td>
</tr>
<tr>
<td>Cores:</td>
<td>2,282,544</td>
</tr>
<tr>
<td>Memory:</td>
<td>2,801,664 GB</td>
</tr>
<tr>
<td>Processor:</td>
<td>IBM POWER9 22C 3.07GHz</td>
</tr>
<tr>
<td>Interconnect:</td>
<td>Dual-rail Mellanox EDR Infiniband</td>
</tr>
</tbody>
</table>

### 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

| 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  |
### Summit - IBM Power System AC922

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

Current #1 on top500

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

Conjugate Gradient on the World's Fastest Computer

Summit - IBM Power System AC922
## Summit - IBM Power System AC922

<table>
<thead>
<tr>
<th>Site:</th>
<th>Oak Ridge National Laboratory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer:</td>
<td>IBM</td>
</tr>
<tr>
<td>Cores:</td>
<td>2,282,544</td>
</tr>
<tr>
<td>Memory:</td>
<td>2,801,664 GB</td>
</tr>
<tr>
<td>Processor:</td>
<td>IBM POWER9 22C 3.07GHz</td>
</tr>
<tr>
<td>Interconnect:</td>
<td>Dual-rail Mellanox EDR Infiniband</td>
</tr>
</tbody>
</table>

### Performance

<table>
<thead>
<tr>
<th>Theoretical peak:</th>
<th>187,659 TFlops/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINPACK benchmark:</td>
<td>122,300 Tflops/s</td>
</tr>
<tr>
<td>HPCG benchmark:</td>
<td>2,926 Tflops/s</td>
</tr>
</tbody>
</table>

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

**HPCG benchmark** (sparse $Ax=b$, iterative) 1.5% efficiency
The Conjugate Gradient (CG) Method

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]

for \( i = 1 : n_{\text{max}} \)

\[ \alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T Ap_{i-1}} \]

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

\[ r_i = r_{i-1} - \alpha_{i-1} Ap_{i-1} \]

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

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

end
The Conjugate Gradient (CG) Method

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]
for \( i = 1 : n_{\text{max}} \)

\[ \alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T Ap_{i-1}} \]

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

\[ r_i = r_{i-1} - \alpha_{i-1} Ap_{i-1} \]

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

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

end
The Conjugate Gradient (CG) Method

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]

for \( i = 1 : n_{\text{max}} \)

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

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

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

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

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

end
The Conjugate Gradient (CG) Method

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]

for \( i = 1:n_{\text{max}} \)

\[ \alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T Ap_{i-1}} \]

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

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

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

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

end
The Conjugate Gradient (CG) Method

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]

for \( i = 1 : n_{\text{max}} \)

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

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

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

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

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

end
The Conjugate Gradient (CG) Method

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]
for \( i = 1 : n_{\text{max}} \)

\[ \alpha_{i-1} = \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T Ap_{i-1}} \]

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

\[ r_i = r_{i-1} - \alpha_{i-1} Ap_{i-1} \]

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

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

end
→ Sparse matrix-vector multiplication (SpMV)
  • $O(\text{nnz})$ flops
  • Must communicate vector entries w/neighbor processors (nearest neighbor MPI collective)
→ 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
→ Sparse matrix-vector multiplication (SpMV)
  • $O(nnz)$ flops
  • Must communicate vector entries with 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
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

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

Both approaches are mathematically equivalent to classical CG
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 $A$, $\|b\| = 1$

$N = 112$, $\kappa(A) \approx 7e6$
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!

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,

  \[ \text{runtime} = (\text{time/iteration}) \times (\# \text{ iterations}) \]

- 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!
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,

\[
\text{runtime} = (\text{time/iteration}) \times (\# \text{ iterations})
\]

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, $b - A\hat{x}_i$, and the updated residual, $\hat{r}_i$, 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, $b - A\hat{x}_i$, and the updated residual, $\hat{r}_i$, to deviate
- Writing $b - A\hat{x}_i = \hat{r}_i + b - A\hat{x}_i - \hat{r}_i$,

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

• Accuracy \(\|x - \hat{x}_i\|\) generally not computable, \(x - \hat{x}_i = A^{-1}(b - A\hat{x}_i)\)

• Size of the true residual, \(\|b - A\hat{x}_i\|\), used as computable measure of accuracy

• Rounding errors cause the true residual, \(b - A\hat{x}_i\), and the updated residual, \(\hat{r}_i\), 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 \(\|\hat{r}_i\| \rightarrow 0\), \(\|b - A\hat{x}_i\|\) 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, $b - A\hat{x}_i$, and the updated residual, $\hat{r}_i$, 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 $\|b - A\hat{x}_i - \hat{r}_i\|$

In finite precision HSCG, iterates are updated by

\[ \hat{x}_i = \hat{x}_{i-1} + \hat{a}_{i-1}\hat{p}_{i-1} - \delta x_i \quad \text{and} \quad \hat{r}_i = \hat{r}_{i-1} - \hat{a}_{i-1}A\hat{p}_{i-1} - \delta r_i \]
In finite precision HSCG, iterates are updated by

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

Let \( f_i \equiv b - A\hat{x}_i - \hat{r}_i \)
In finite precision HSCG, iterates are updated by

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

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

\[ f_i = b - A(\hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \delta x_i) - (\hat{r}_{i-1} - \hat{\alpha}_{i-1}A\hat{p}_{i-1} - \delta r_i) \]
In finite precision HSCG, iterates are updated by

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

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

\[
\begin{align*}
    f_i &= b - A(\hat{x}_{i-1} + \hat{\alpha}_{i-1} \hat{p}_{i-1} - \delta x_i) - (\hat{r}_{i-1} - \hat{\alpha}_{i-1} A \hat{p}_{i-1} - \delta r_i) \\
    &= f_{i-1} + A \delta x_i + \delta r_i
\end{align*}
\]
In finite precision HSCG, iterates are updated by

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

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

\[
\begin{align*}
  f_i &= b - A(\hat{x}_{i-1} + \hat{\alpha}_{i-1} \hat{p}_{i-1} - \delta x_i) - (\hat{r}_{i-1} - \hat{\alpha}_{i-1} A \hat{p}_{i-1} - \delta r_i) \\
  &= f_{i-1} + A \delta x_i + \delta r_i \\
  &= f_0 + \sum_{m=1}^{i} (A \delta x_m + \delta r_m)
\end{align*}
\]
• In finite precision HSCG, iterates are updated by

\[
\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1}\hat{p}_{i-1} - \delta x_i
\]
and

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

• Let \( f_i \equiv b - A\hat{x}_i - \hat{r}_i \)

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

\[
= f_{i-1} + A\delta x_i + \delta r_i
\]

\[
= f_0 + \sum_{m=1}^{i}(A\delta x_m + \delta r_m)
\]

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

\[
\|f_i\| \leq O(\varepsilon) \|A\|(\|x\| + \max_{m=0,...,i} \|\hat{x}_m\|) \\
\text{Greenbaum, 1997}
\]

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

- Overall idea: use auxiliary recurrences and modified formulas for recurrence coefficients $\alpha_i$ and $\beta_i$ to reduce/decouple synchronization points
Pipelined CG (GVCG)

• Overall idea: use auxiliary recurrences and modified formulas for recurrence coefficients $\alpha_i$ and $\beta_i$ 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 $\alpha_i$ and $\beta_i$ to reduce/decouple synchronization points
- Long history of related work:
  - 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 $\alpha_i$ and $\beta_i$ 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 $\alpha_i$ 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 $\alpha_i$ and $\beta_i$ 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 $\alpha_i$ 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 $A^2r_i$
  • Removes sequential dependency between matrix-vector products and inner products
  • Computations can then be overlapped using nonblocking (asynchronous) communication $\Rightarrow$ hides the latency of global communications
GVCG (Ghysels and Vanroose 2014)

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]

\[ s_0 = Ap_0, \quad w_0 = Ar_0, \quad z_0 = Aw_0, \]

\[ \alpha_0 = \frac{r_0^T r_0}{p_0^T s_0} \]

for \( i = 1 : \text{nmax} \)

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

\[ r_i = r_{i-1} - \alpha_{i-1} s_{i-1} \]

\[ w_i = w_{i-1} - \alpha_{i-1} z_{i-1} \]

\[ q_i = Aw_i \]

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

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

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

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

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

end
\( r_0 = b - Ax_0, \ p_0 = r_0 \)

\( s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0, \)

\( \alpha_0 = r_0^T r_0 / p_0^T s_0 \)

for \( i = 1 \) to \( n_{\text{max}} \)

\[ \begin{align*}
    x_i &= x_{i-1} + \alpha_{i-1} p_{i-1} \\
    r_i &= r_{i-1} - \alpha_{i-1} s_{i-1} \\
    w_i &= w_{i-1} - \alpha_{i-1} z_{i-1} \\
    q_i &= Aw_i \\
    \beta_i &= \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \\
    \alpha_i &= \frac{r_i^T r_i}{w_{i-1}^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i} \\
    p_i &= r_i + \beta_i p_{i-1} \\
    s_i &= w_i + \beta_i s_{i-1} \\
    z_i &= q_i + \beta_i z_{i-1}
\end{align*} \]

end
\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]
\[ s_0 = Ap_0, \quad w_0 = Ar_0, \quad z_0 = Aw_0, \]
\[ \alpha_0 = r_0^T r_0 / p_0^T s_0 \]

for \( i = 1:n_{\text{max}} \)

\[ x_i = x_{i-1} + \alpha_{i-1} p_{i-1} \]
\[ r_i = r_{i-1} - \alpha_{i-1} s_{i-1} \]
\[ w_i = w_{i-1} - \alpha_{i-1} z_{i-1} \]
\[ q_i = Aw_i \]
\[ \beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \]
\[ \alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i} \]
\[ p_i = r_i + \beta_i p_{i-1} \]
\[ s_i = w_i + \beta_i s_{i-1} \]
\[ z_i = q_i + \beta_i z_{i-1} \]

end
GVCG (Ghysels and Vanroose 2014)

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]
\[ s_0 = Ap_0, \quad w_0 = Ar_0, \quad z_0 = Aw_0, \]
\[ \alpha_0 = \frac{r_0^T r_0}{p_0^T s_0} \]

for \( i = 1 \) : \( n_{\text{max}} \)

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

\[ q_i = Aw_i \]
\[ \beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \]
\[ \alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i/\alpha_{i-1}) r_i^T r_i} \]
\[ p_i = r_i + \beta_i p_{i-1} \]
\[ s_i = w_i + \beta_i s_{i-1} \]
\[ z_i = q_i + \beta_i z_{i-1} \]

end
GVCG (Ghysels and Vanroose 2014)

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

Diagram: Iteration Loop
- Vector Updates
- Inner Products
- SpMV
- Vector Updates
- End Loop
- Overlap
\[ r_0 = b - Ax_0, \ p_0 = r_0 \]
\[ s_0 = Ap_0, w_0 = Ar_0, z_0 = Aw_0, \]
\[ \alpha_0 = r_0^T r_0 / p_0^T s_0 \]

for \( i = 1: n_{\text{max}} \)

\[ x_i = x_{i-1} + \alpha_{i-1} p_{i-1} \]
\[ r_i = r_{i-1} - \alpha_{i-1} s_{i-1} \]
\[ w_i = w_{i-1} - \alpha_{i-1} z_{i-1} \]
\[ q_i = A w_i \]
\[ \beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \]
\[ \alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i} \]
\[ p_i = r_i + \beta_i p_{i-1} \]
\[ s_i = w_i + \beta_i s_{i-1} \]
\[ z_i = q_i + \beta_i z_{i-1} \]

end
GVC (Ghysels and Vanroose 2014)

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]
\[ s_0 = Ap_0, \quad w_0 = Ar_0, \quad z_0 = Aw_0, \]
\[ \alpha_0 = \frac{r_0^T r_0}{p_0^T s_0} \]

for \( i = 1:n_{\text{max}} \)

\[ x_i = x_{i-1} + \alpha_{i-1} p_{i-1} \]
\[ r_i = r_{i-1} - \alpha_{i-1} s_{i-1} \]
\[ w_i = w_{i-1} - \alpha_{i-1} z_{i-1} \]
\[ q_i = A w_i \]
\[ \beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \]
\[ \alpha_i = \frac{r_i^T r_i}{w_i^T r_i - (\beta_i / \alpha_{i-1}) r_i^T r_i} \]
\[ p_i = r_i + \beta_i p_{i-1} \]
\[ s_i = w_i + \beta_i s_{i-1} \]
\[ z_i = q_i + \beta_i z_{i-1} \]

end
• 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

\[
\begin{align*}
    r_0 &= b - Ax_0, p_0 = r_0, s_0 = Ap_0 \\
    &\text{for } i = 1:n_{\text{max}} \\
    \alpha_{i-1} &= \frac{(r_{i-1},r_{i-1})}{(p_{i-1},s_{i-1})} \\
    x_i &= x_{i-1} + \alpha_{i-1}p_{i-1} \\
    r_i &= r_{i-1} - \alpha_{i-1}s_{i-1} \\
    \beta_i &= \frac{(r_i,r_i)}{(r_{i-1},r_{i-1})} \\
    p_i &= r_i + \beta_ip_{i-1} \\
    s_i &= Ar_i + \beta_is_{i-1} \\
    \text{end}
\end{align*}
\]
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 $\alpha$ and $\beta$ as HSCG, but uses additional recurrence for $Ap_i$

\[
\begin{align*}
\alpha_{i-1} &= \frac{(r_{i-1},r_{i-1})}{(p_{i-1},s_{i-1})} \\
x_i &= x_{i-1} + \alpha_{i-1}p_{i-1} \\
r_i &= r_{i-1} - \alpha_{i-1}s_{i-1} \\
\beta_i &= \frac{(r_i,r_i)}{(r_{i-1},r_{i-1})} \\
p_i &= r_i + \beta_ip_{i-1} \\
s_i &= Ar_i + \beta_is_{i-1}
\end{align*}
\]

\[r_0 = b - Ax_0, p_0 = r_0, s_0 = Ap_0\]
for $i = 1:nmax$

see [C., Rozložník, Strakoš, Tíchy, Tůma, 2018]
\[ \hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1} \hat{p}_{i-1} + \delta x_i \quad \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1} \hat{s}_{i-1} + \delta r_i \]
\[
\hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1} \hat{p}_{i-1} + \delta x_i \\
\hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1} \hat{s}_{i-1} + \delta r_i \\
f_i = \hat{r}_i - (b - A\hat{x}_i)
\]
\[ \hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1} \hat{p}_{i-1} + \delta x_i \]
\[ \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1} \hat{s}_{i-1} + \delta r_i \]
\[ f_i = \hat{r}_i - (b - A \hat{x}_i) \]
\[ = f_{i-1} - \hat{\alpha}_{i-1} (\hat{s}_{i-1} - A \hat{p}_{i-1}) + \delta r_i + A \delta x_i \]
Attainable accuracy of simple pipelined CG

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

\[ \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1} \hat{s}_{i-1} + \delta r_i \]

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

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

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

where

\[ G_i = \hat{S}_i - A\hat{P}_i, \quad d_i = [\hat{\alpha}_0, \ldots, \hat{\alpha}_{i-1}]^T \]
Attainable accuracy of simple pipelined CG

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

\[ \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1} \hat{s}_{i-1} + \delta r_i \]

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

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

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

where

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

Classical CG: \[ f_i = f_0 + \sum_{m=1}^{i} (A\delta x_m + \delta r_m) \]
Attainable accuracy of simple pipelined CG

\[ \hat{x}_i = \hat{x}_{i-1} + \hat{\alpha}_{i-1} \hat{p}_{i-1} + \delta x_i \]
\[ \hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1} \hat{s}_{i-1} + \delta r_i \]

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

where

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

Classical CG: \[ f_i = f_0 + \sum_{m=1}^{i} (A\delta x_m + \delta r_m) \]
Attainable accuracy of simple pipelined CG

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

\[
\hat{U}_i = \begin{bmatrix} 1 & -\hat{\beta}_1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & 1 - \hat{\beta}_{i-1} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \quad \text{and} \quad \hat{U}_i^{-1} = \begin{bmatrix} 1 & \hat{\beta}_1 & \cdots & \cdots & \hat{\beta}_1 \hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\ 0 & 1 & \hat{\beta}_2 & \cdots & \hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & \hat{\beta}_{i-1} & \cdots \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}
\]
\[
\|G_i\| \leq \frac{O(\varepsilon)}{1 - O(\varepsilon)} (\kappa(\hat{U}_i) \|A\|\|\hat{P}_i\| + \|A\|\|\hat{R}_i\|\|\hat{U}_i^{-1}\|)
\]

\[
\hat{U}_i = \begin{bmatrix}
1 & -\hat{\beta}_1 & 0 & 0 \\
0 & 1 & \vdots & 0 \\
\vdots & \vdots & 1 & -\hat{\beta}_{i-1} \\
0 & \cdots & 0 & 1
\end{bmatrix} \quad \hat{U}_i^{-1} = \begin{bmatrix}
1 & \hat{\beta}_1 & \cdots & \cdots & \hat{\beta}_1\hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\
0 & 1 & \hat{\beta}_2 & \cdots & \hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & 1 \\
0 & \cdots & \cdots & 0 & 1
\end{bmatrix}
\]

\[
\beta_\ell \beta_{\ell+1} \cdots \beta_j = \frac{\|r_j\|^2}{\|r_{\ell-1}\|^2}, \quad \ell < j
\]
Attainable accuracy of simple pipelined CG

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

\[ \hat{U}_i = \begin{bmatrix} 1 & -\hat{\beta}_1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & -\hat{\beta}_{i-1} \\ 0 & \ldots & 0 & 1 \end{bmatrix} \quad \hat{U}_i^{-1} = \begin{bmatrix} 1 & \hat{\beta}_1 & \ldots & \ldots & \hat{\beta}_1 \hat{\beta}_2 \ldots \hat{\beta}_{i-1} \\ 0 & 1 & \hat{\beta}_2 & \ddots & \ldots \hat{\beta}_2 \ldots \hat{\beta}_{i-1} \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & 1 & \hat{\beta}_{i-1} \\ 0 & \ldots & \ldots & 0 & 1 \end{bmatrix} \]

\[
\beta_{\ell} \beta_{\ell+1} \ldots \beta_j = \frac{\|r_j\|^2}{\|r_{\ell-1}\|^2}, \quad \ell < j
\]

- Residual oscillations can cause these factors to be large!
- Errors in computed recurrence coefficients can be amplified!
Attainable accuracy of simple pipelined CG

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

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

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

- 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

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

\[ \hat{U}_i = \begin{bmatrix} 1 & -\hat{\beta}_1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & -\hat{\beta}_{i-1} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \quad \hat{U}_i^{-1} = \begin{bmatrix} 1 & \hat{\beta}_1 & \cdots & \cdots & \hat{\beta}_1\hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\ 0 & 1 & \hat{\beta}_2 & \cdots & \hat{\beta}_2 \cdots \hat{\beta}_{i-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 & \hat{\beta}_{i-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \]

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

- 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

![Graph showing the A-norm of the error versus iteration for HSCG method.](image)
Simple pipelined CG

effect of using auxiliary vector $s_i \equiv A p_i$
Simple pipelined CG

Effect of changing formula for recurrence coefficient $\alpha$ and using auxiliary vector $s_i \equiv Ap_i$
Simple pipelined CG

effect of changing formula for recurrence coefficient $\alpha$ 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 $\alpha$ and $\beta$ (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 $i$th 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

\[
\int \lambda^{-1} d\omega(\lambda) = \sum_{i=1}^{i} \omega_{\ell}^{(i)} \left\{ \theta_{\ell}^{(i)} \right\}^{-1} + \frac{\|x - x_i\|_{A}^2}{\|r_0\|^2}
\]
Towards understanding convergence delay

- Coefficients $\alpha$ and $\beta$ (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 $i$th 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

\[
\int \lambda^{-1} d\omega(\lambda) = \sum_{\ell=1}^{i} \omega^{(i)}_{\ell} \left\{\theta^{(i)}_{\ell}\right\}^{-1} + \frac{\|x - x_i\|_A^2}{\|r_0\|^2}
\]
Towards understanding convergence delay

• Coefficients $\alpha$ and $\beta$ (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 $i$th 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

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

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

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

where $F_i$ is small relative to error term?
Towards understanding convergence delay

- Coefficients $\alpha$ and $\beta$ (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 $i$th Gauss-Christoffel quadrature.

- CG method = matrix formulation of Gauss-Christoffel quadrature (see, e.g., [Liesen & Strakosh, 2013]).

- $A$-norm of CG error for $f(\lambda) = \lambda^{-1}$ given as scaled quadrature error:

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

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

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

where $F_i$ is small relative to error term?

- For classical CG, yes; proved by Greenbaum [1989].
Towards understanding convergence delay

- Coefficients $\alpha$ and $\beta$ (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 $i$th 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

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

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

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

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 $\gamma_i, \delta_i$ in Jacobi matrices $T_i$ in HSCG vs. GVCG
(matrix bcsstk03)
eigenvalues of $A$

- $\times$ eigenvalues of $\hat{T}_{400}$, HSCG
- $\circ$ 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
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-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:
Key observation: After iteration $i$, for $j \in \{0, \ldots, s\}$,

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

**s steps of s-step CG:**

- **Expand solution space $s$ dimensions at once**
  - Compute “basis” matrix $\mathbf{y}$ such that
    \[ \text{span}(\mathbf{y}) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i) \]
    according to the recurrence $A\mathbf{y} = \mathbf{y} \mathbf{B}$

- **Compute inner products basis vectors in one synchronization**
  \[ g = \mathbf{y}^T \mathbf{y} \]

$O(1)$ messages
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:

**Expand solution space $s$ dimensions at once**

- Compute “basis” matrix $Y$ such that
  \[
  \text{span}(Y) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)
  \]
  according to the recurrence $AY = YB$

**Compute inner products basis vectors in one synchronization**

$$g = Y^T Y$$

**Compute $s$ iterations of vector updates**

Perform $s$ iterations of vector updates by updating coordinates in basis $Y$:

$$x_{i+j} - x_i = Yx'_j, \quad r_{i+j} = Yr'_j, \quad p_{i+j} = Yp'_j$$

$O(1)$ messages

no data movement
s-step CG

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

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

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

s steps of s-step CG:

Expand solution space $s$ dimensions at once

- Compute “basis” matrix $Y$ such that
  $$\text{span}(Y) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$$
  according to the recurrence $AY = YB$

Compute inner products basis vectors in one synchronization

$$G = Y^T Y$$

Compute $s$ iterations of vector updates

- Perform $s$ iterations of vector updates by updating coordinates in basis $Y$:
  $$x_{i+j} - x_i = Yx'_j, \quad r_{i+j} = Yr'_j, \quad p_{i+j} = Yp'_j$$

$O(1)$ messages

no data movement
s-step CG

\[ r_0 = b - Ax_0, p_0 = r_0 \]

for \( k = 0 \):nmax/s

- Compute \( y_k \) and \( B_k \) such that \( Ay_k = y_k B_k \) and \( \text{span}(y_k) = K_{s+1}(A, p_{sk}) + K_s(A, r_{sk}) \)
- \( G_k = y_k^T y_k \)
- \( x'_0 = 0, r'_0 = e_{s+2}, p'_0 = e_1 \)

for \( j = 1 : s \)

\[ \alpha_{sk+j-1} = \frac{r_{j-1}^T G_k r'_{j-1}}{p_{j-1}^T G_k B_k p'_{j-1}} \]
\[ x'_j = x'_{j-1} + \alpha_{sk+j-1} p'_{j-1} \]
\[ r'_j = r'_{j-1} - \alpha_{sk+j-1} B_k p'_{j-1} \]
\[ \beta_{sk+j} = \frac{r_{j}^T G_k r_{j}}{r_{j-1}^T G_k r'_{j-1}} \]
\[ p'_j = r'_j + \beta_{sk+j} p'_{j-1} \]

end

\[ [x_{s(k+1)} - x_{sk}, r_{s(k+1)}, p_{s(k+1)}] = y_k [x'_s, r'_s, p'_s] \]

end
\[ \begin{align*}
r_0 &= b - Ax_0, p_0 = r_0 \\
\text{for } k = 0:\text{nmax}/s \\
\text{Compute } y_k \text{ and } b_k \text{ such that } A y_k = y_k b_k \text{ and } \\
\text{span}(y_k) = \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk}) \\
G_k &= y_k^T y_k \\
x'_0 = 0, r'_0 = e_{s+2}, p'_0 = e_1 \\
\text{for } j = 1:s \\
\alpha_{sk+j-1} &= \frac{r'_{j-1}^T g_k r'_{j-1}}{p'_{j-1}^T g_k b_k p'_{j-1}} \\
x'_j = x'_{j-1} + \alpha_{sk+j-1} p'_{j-1} \\
r'_j = r'_{j-1} - \alpha_{sk+j-1} b_k p'_{j-1} \\
\beta_{sk+j} &= \frac{r'_j^T g_k r'_j}{r'_{j-1}^T g_k r'_{j-1}} \\
p'_j = r'_j + \beta_{sk+j} p'_{j-1} \\
\end{align*} \]

\[ [x_s(k+1) - x_{sk}, r_s(k+1), p_s(k+1)] = y_k [x'_s, r'_s, p'_s] \]

end
s-step CG

\[ r_0 = b - Ax_0, p_0 = r_0 \]

for \( k = 0:nmax/s \)

Compute \( y_k \) and \( B_k \) such that \( A y_k = y_k B_k \) and

\[ \text{span}(y_k) = K_{s+1}(A, p_{sk}) + K_s(A, r_{sk}) \]

\[ G_k = y_k^T y_k \]

\( x'_0 = 0, r'_0 = e_{s+2}, p'_0 = e_1 \)

for \( j = 1:s \)

\[ \alpha_{sk+j-1} = \frac{r_{j-1}^T G_k r'_{j-1}}{p_{j-1}^T G_k B_k p'_{j-1}} \]

\[ x'_j = x'_{j-1} + \alpha_{sk+j-1} p'_{j-1} \]

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

\[ \beta_{sk+j} = \frac{r_{j}^T G_k r'_{j}}{r_{j-1}^T G_k r'_{j-1}} \]

\[ p'_j = r'_j + \beta_{sk+j} p'_{j-1} \]

end

\[ [ x_{s(k+1)-x_{sk}}, r_{s(k+1)-r_{sk}}, p_{s(k+1)-p_{sk}} ] = y_k [ x'_s, r'_s, p'_s ] \]

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

\[ r_0 = b - Ax_0, \quad p_0 = r_0 \]

for \( k = 0 : \text{nmax}/s \)

Compute \( y_k \) and \( B_k \) such that \( A y_k = y_k B_k \) and

\[ \text{span}(y_k) = \mathcal{K}_{s+1}(A,p_{sk}) + \mathcal{K}_s(A,r_{sk}) \]

\[ \mathcal{G}_k = y_k^T y_k \]

\[ x'_0 = 0, r'_0 = e_{s+2}, p'_0 = e_1 \]

for \( j = 1 : s \)

\[ \alpha_{sk+j-1} = \frac{r_{j-1}^T \mathcal{G}_k r_{j-1}}{p_{j-1}^T \mathcal{G}_k B_k p_{j-1}} \]

\[ x'_j = x'_{j-1} + \alpha_{sk+j-1} p'_{j-1} \]

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

\[ \beta_{sk+j} = \frac{r_{j}^T \mathcal{G}_k r_{j}}{r_{j-1}^T \mathcal{G}_k r_{j-1}} \]

\[ p'_j = r'_j + \beta_{sk+j} p'_{j-1} \]

end

\[ [x_{s(k+1)} - x_{sk}, r_{s(k+1)}, p_{s(k+1)}] = y_k [x'_s, r'_s, p'_s] \]

end
Numerical Behavior of s-step CG

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

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

\( A: \text{bcsstk03 from UFSMC} \)

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

\( N = 112, \kappa(A) \approx 7e6 \)

s-step CG with monomial basis \( (y = [p_i, Ap_i, ..., A^sp_i, r_i, Ar_i, ... A^{s-1}r_i]) \)
Numerical Behavior of s-step CG

\(A: \text{bcsstk03 from UFSMC}\)

\(b: \text{equal components in the eigenbasis of } A \text{ and } \|b\| = 1\)

\(N = 112, \kappa(A) \approx 7e6\)

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

\( A: \text{bcsstk03} \) from UFSMC

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

\( N = 112, \kappa(A) \approx 7e6 \)

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

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

s-step CG with monomial basis \[ \{y = [p_i, Ap_i, ..., A^s p_i, r_i, Ar_i, ... A^{s-1} r_i]\} \]

Effects of roundoff error:
1. convergence delay
2. loss of accuracy
Numerical Behavior of s-step CG

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

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

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:

\[ A \hat{Y}_k = \hat{Y}_k B_k + \Delta Y_k \]

Updating coordinate vectors in the inner loop, \( j = 1: s \):

\[
\begin{align*}
\hat{x}'_j &= \hat{x}'_{j-1} + \hat{q}'_{j-1} + \xi_j \\
\hat{r}'_j &= \hat{r}'_{j-1} - B_k \hat{q}'_{j-1} + \eta_j \\
\text{with} & \quad \hat{q}'_{j-1} = \text{fl}(\hat{a}_{sk+j-1} \hat{p}'_{j-1})
\end{align*}
\]

Recovering CG vectors for use in next outer loop:

\[
\begin{align*}
\hat{x}_{sk+s} &= \hat{Y}_k \hat{x}'_j + \hat{x}_{sk} + \phi_{sk+s} \\
\hat{r}_{sk+s} &= \hat{Y}_k \hat{r}'_j + \psi_{sk+s}
\end{align*}
\]
Sources of Roundoff Error in s-step CG

Error in outer iteration k:

Computing the s-step Krylov subspace basis:

\[ A\hat{Y}_k = \hat{Y}_k B_k + \Delta Y_k \]

Updating coordinate vectors in the inner loop, \( j = 1: s \):

\[ \hat{x}'_j = \hat{x}'_{j-1} + \hat{q}'_{j-1} + \xi_j \]
\[ \hat{r}'_j = \hat{r}'_{j-1} - B_k \hat{q}'_{j-1} + \eta_j \]

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

Recovering CG vectors for use in next outer loop:

\[ \hat{x}_{sk+s} = \hat{Y}_k \hat{x}'_j + \hat{x}_{sk} + \phi_{sk+s} \]
\[ \hat{r}_{sk+s} = \hat{Y}_k \hat{r}'_j + \psi_{sk+s} \]
Sources of Roundoff Error in s-step CG

Error in outer iteration $k$:

Computing the $s$-step Krylov subspace basis:

$$A \hat{Y}_k = \hat{Y}_k B_k + \Delta Y_k$$

Error in computing $s$-step basis

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

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

with

$$\hat{q}'_{j-1} = \text{fl}(\hat{a}_{sk+j-1} \hat{p}'_{j-1})$$

Error in updating coefficient vectors

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+s} = \hat{Y}_k \hat{x}'_j + \hat{x}_{sk} + \phi_{sk+s}$$
$$\hat{r}_{sk+s} = \hat{Y}_k \hat{r}'_j + \psi_{sk+s}$$
Sources of Roundoff Error in s-step CG

Error in outer iteration $k$:

Computing the $s$-step Krylov subspace basis:

$$A\hat{Y}_k = \hat{Y}_kB_k + \Delta Y_k$$

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

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

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

Recovering CG vectors for use in next outer loop:

$$\hat{x}_{sk+s} = \hat{Y}_k\hat{x}'_j + \hat{x}_{sk} + \phi_{sk+s}$$
$$\hat{r}_{sk+s} = \hat{Y}_k\hat{r}'_j + \psi_{sk+s}$$
Attainable Accuracy of s-step CG

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

For CG:

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

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

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

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

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

[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

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

\[
\hat{V}_m = [\hat{v}_1, ..., \hat{v}_m], \quad \delta\hat{V}_m = [\delta\hat{v}_1, ..., \delta\hat{v}_m], \quad \hat{T}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & \cdots & \hat{\beta}_{m+1} \\ \hat{\beta}_2 & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots \\ \hat{\beta}_m & \cdots & \ddots & \ddots \\ \hat{\beta}_{m+1} & \cdots & \cdots & \ddots & \ddots \\ \end{bmatrix}
\]

for \( i \in \{1, ..., m\} \)

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

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

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

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

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)

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

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

for \(i \in \{1, ..., m\},\)

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

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

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

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

Lanczos [Paige, 1976]

\[
\varepsilon_0 = O(\varepsilon N)
\]
\[
\varepsilon_1 = O(\varepsilon n \theta)
\]

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

\[
\varepsilon_0 = O(\varepsilon N \Gamma^2)
\]
\[
\varepsilon_1 = O(\varepsilon n \theta \Gamma)
\]

\[
\Gamma = c \cdot \max_{\ell \leq k} \|\hat{G}_\ell^+\| \|\hat{G}_\ell\|
\]
Convergence of Ritz Values in s-step Lanczos

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

$$\Gamma \leq (24\varepsilon(N + 11s + 15))^{-1/2} \approx \frac{1}{\sqrt{N\varepsilon}}$$
Convergence of Ritz Values in s-step Lanczos

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

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

• Bounds on accuracy of Ritz values depend on $\Gamma^2$
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

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

- Bounds on accuracy of Ritz values depend on \( \Gamma^2 \)
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

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

- Bounds on accuracy of Ritz values depend on \( \Gamma^2 \)

\[ (\Gamma = c \cdot \max_{\ell \leq k} \| \tilde{Y}_\ell^+ \| \| \tilde{Y}_\ell \|) \]
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
  \[ \Gamma \leq \left( 24\varepsilon(N + 11s + 15) \right)^{-1/2} \approx \frac{1}{\sqrt{N\varepsilon}} \]

- Bounds on accuracy of Ritz values depend on \( \Gamma^2 \)

If \( \Gamma \approx 1 \):

s-step Lanczos behaves the same numerically as classical Lanczos
A different problem...

\( A: \text{nos4 from SuiteSparse} \)

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

\( N = 100, \kappa(A) \approx 2e3 \)
A different problem...

\( A: \text{ nos4 from SuiteSparse} \)
\( b: \) equal components in the eigenbasis of \( A \) and \( \|b\| = 1 \)
\( N = 100, \kappa(A) \approx 2e3 \)
A different problem…

\( A: \text{nos4} \text{ from SuiteSparse} \)

\( b: \) 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 \(\|b\| = 1\)

\(N = 100, \kappa(A) \approx 2e3\)

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 \(\|b\| = 1\)

\(N = 100, \kappa(A) \approx 2e3\)

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

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

\[f_i \equiv b - A\hat{x}_i - \hat{r}_i\]
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:

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

- If our application requires relative accuracy $\varepsilon^*$, we must have:

\[
\Gamma_k \equiv c \cdot \|\hat{y}_k^+\| \|\hat{y}_k\| \lesssim \frac{\varepsilon^*}{\varepsilon \max_{j\in\{0,\ldots,s\}} \|\hat{r}_{m+j}\|}
\]

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

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

- If our application requires relative accuracy $\varepsilon^*$, we must have

\[
\Gamma_k \equiv c \cdot \|\hat{\gamma}_k\| \|\hat{\gamma}_k\| \leq \frac{\varepsilon^*}{\varepsilon \max_{j \in \{0, \ldots, s\}} \|\hat{r}_{m+j}\|}
\]

- $\|\hat{r}_i\|$ large $\rightarrow$ $\Gamma_k$ must be small; $\|\hat{r}_i\|$ small $\rightarrow$ $\Gamma_k$ 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

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

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

- If our application requires relative accuracy $\varepsilon^*$, we must have

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

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

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

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

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

\( s=8, \ v^*=1.0\times10^{-14} \)

[Graph showing the convergence of the residual norm with iterations for different methods, labeled as s-step CG and adaptive s-step CG compared to CG.]
Adaptive s-step CG

mesh3e1 (UFSMC)

$n = 289$

$\kappa(A) \approx 10$

$b_i = 1/\sqrt{N}$

$s = 8, \varepsilon^* = 1 \times 10^{-6}$
runtime = \left( \text{time per iteration} \right) \times \left( \text{number of iterations until convergence} \right)
runtime = \left( \text{time per iteration} \right) \times \left( \text{number of iterations until convergence} \right)

Reduce time per iteration

- approximate operators
- modify algorithm to reduce communication
- asynchronous execution
- reduced precision
runtime = \left( \text{time per iteration} \right) \times \left( \text{number of iterations until convergence} \right)

**Takeaway**

- 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
runtime = \left( \text{time per iteration} \right) \times \left( \text{number of iterations until convergence} \right)

**Takeaway**

- **Reduce time per iteration**
  - approximate operators
  - modify algorithm to reduce communication
  - asynchronous execution
  - reduced precision

- **Reduce number of iterations**
  - block methods
  - preconditioning
  - eigenvalue deflation
  - subspace recycling
  - increased precision
Takeaway

\[ \text{runtime} = \left( \frac{\text{time per iteration}}{\text{iteration}} \right) \times \left( \frac{\text{number of iterations}}{\text{until convergence}} \right) \]

Reduce time per iteration:
- approximate operators
- asynchronous execution
- reduced precision
- modify algorithm to reduce communication

Reduce number of iterations:
- block methods
- preconditioning
- eigenvalue deflation
- subspace recycling
- increased precision
Takeaway

$$\text{runtime} = \left(\frac{\text{time per iteration}}{\text{iteration}}\right) \times \left(\frac{\text{number of iterations}}{\text{until convergence}}\right)$$

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

$$Ax = b \Rightarrow M_L^{-1} AM_R^{-1}u = M_L^{-1}b$$
$$x = M_R^{-1}u$$
runtime = \((\text{time per iteration}) \times (\text{number of iterations until convergence})\)

Reduce time per iteration
- approximate operators
- modify algorithm to reduce communication
- asynchronous execution
- reduced precision

Reduce number of iterations
- block methods
- preconditioning
- eigenvalue deflation
- subspace recycling
- increased precision

\[ Ax = b \Rightarrow M_L^{-1} A M_R^{-1} u = M_L^{-1} b \]

\[ x = M_R^{-1} u \]
runtime = \((\text{time per iteration}) \times (\text{number of iterations until convergence})\)

**Takeaway**

- Reduce time per iteration:
  - approximate operators
  - modify algorithm to reduce communication
  - asynchronous execution
  - reduced precision

- Reduce number of iterations:
  - block methods
  - preconditioning
  - eigenvalue deflation
  - subspace recycling
  - increased precision

**doubled precision → twice as many bits moved**
runtime = \left( \text{time per iteration} \right) \times \left( \text{number of iterations until convergence} \right)

Reduce time per iteration
- approximate operators
- modify algorithm to reduce communication
- asynchronous execution
- reduced precision

Reduce number of iterations
- block methods
- preconditioning
- eigenvalue deflation
- subspace recycling
- increased precision

Takeaway
runtime = \left( \text{time per iteration} \right) \times \left( \text{number of iterations until convergence} \right)

\hat{A}x \approx Ax

Reduce time per iteration
- approximate operators
- modify algorithm to reduce communication
- asynchronous execution
- reduced precision

Reduce number of iterations
- block methods
- preconditioning
- eigenvalue deflation
- subspace recycling
- increased precision
runtime = \left( \text{time per iteration} \right) \times \left( \text{number of iterations until convergence} \right)

Reduce time per iteration
- approximate operators
- modify algorithm to reduce communication
- asynchronous execution
- reduced precision

Reduce number of iterations
- block methods
- preconditioning
- eigenvalue deflation
- subspace recycling
- increased precision

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

\[ \text{runtime} = \left( \text{time per iteration} \right) \times \left( \text{number of iterations until convergence} \right) \]

- **Reduce time per iteration**
  - approximate operators
  - modify algorithm to reduce communication
  - asynchronous execution
  - reduced precision

- **Reduce number of iterations**
  - block methods
  - preconditioning
  - eigenvalue deflation
  - subspace recycling
  - increased precision

convergence criteria never met: divergence, or convergence to inaccurate solution
runtime = (time per iteration) × (number of iterations until convergence)

To minimize runtime, must understand how modifications affect:
1) attainable accuracy  
2) convergence rate  
3) time per iteration

Reduce time per iteration
- approximate operators
- modify algorithm to reduce communication
- asynchronous execution
- reduced precision

Reduce number of iterations
- block methods
- preconditioning
- eigenvalue deflation
- subspace recycling
- increased precision
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

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

• 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

• 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
• Numerical linear algebra routines are the core computational kernels in many data science and machine learning applications
• 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 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
  
  (% accuracy on test dataset) ↔ (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!

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 $r_i$:

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

- In finite precision:

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

$$\hat{r}_i = \hat{r}_{i-1} - \hat{\alpha}_{i-1}\hat{s}_{i-1} + \delta r_i$$

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

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

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

where

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

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

Example: Tridiagonal matrix

\[
\begin{pmatrix}
  y_0 \\
  y_1 \\
  y_2 \\
  y_3 \\
  y_4 \\
\end{pmatrix}
= \begin{pmatrix}
  x & x \\
  x & x & x \\
  x & x & x & x \\
  x & x & x & x & x \\
\end{pmatrix}
\begin{pmatrix}
  x_0 \\
  x_1 \\
  x_2 \\
  x_3 \\
  x_4 \\
\end{pmatrix}
\]
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

Tridiagonal Example:

Sequential

Parallel

Also works for general graphs!
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 $s \leq \sqrt{n/p}$)

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
& \text{Flops} & & \text{Words Moved} & & \text{Messages} \\
& \text{SpMV} & \text{Orth.} & \text{SpMV} & \text{Orth.} & \text{SpMV} & \text{Orth.} \\
\hline
\text{Classical CG} & \frac{sn}{p} & \frac{sn}{p} & \sqrt{\frac{n}{p}} & s \log_2 p & s & s \log_2 p \\
\hline
\text{s-step CG} & \frac{sn}{p} & \frac{s^2 n}{p} & \sqrt{\frac{n}{p}} & s^2 \log_2 p & 1 & \log_2 p \\
\hline
\end{array}
\]

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, \( \mathcal{K}_m(A, v) = \text{span}\{v, Av, \ldots, A^{m-1}v\} \)

• Simple loop unrolling gives monomial basis \( Y = [p, Ap, A^2p, A^3p, \ldots] \)
  
  • Condition number can grow exponentially with \( s \)
  
  • Condition number = ratio of largest to smallest eigenvalues, \( \lambda_{\text{max}}/\lambda_{\text{min}} \)
  
  • 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
- Walker: GMRES
- Leland: CG
- Leland: GMRES
- Chronopoulos and Gear: CG
- Chronopoulos and Kim: Orthomin, Lanczos
- Chronopoulos: MINRES, GCR, Orthomin
- Chronopoulos: GMRES
- Bai, Hu, and Reichel: GMRES
- de Sturler: GMRES
- Joubert and Carey: GMRES
- Kim and Chronopoulos: Arndoli, Symm.
- Chronopoulos and Kim: Nonsymm.
- de Sturler and van der Vorst: GMRES
- Toledo: CG
- Erhel: GMRES
- Chronopoulos and Kinkaid: Orthodir

First termed "$s$-step methods"
Recent Years...

- **2010**: First termed "CA" methods; first TSQR, general matrix powers kernel.
  - Hoemmen: Arnoldi, GMRES, Lanczos, CG

- **2011**: First CA-BICGSTAB method.
  - Carson, Knight, and Demmel: BICG, CGS, BICGSTAB

- **2012**: First theoretical results on finite precision behavior.
  - Carson and Demmel: CG-RR, BICG-RR

- **2013**: Feuerriegel and Bücker: Lanczos, BICG, QMR

- **2014**: 
  - Carson and Demmel: 2-term Lanczos
  - Grigori, Moufawad, Nataf: CG
  - Ballard, Carson, Demmel, Hoemmen, Knight, Schwartz: Arnoldi, GMRES, Nonsymm. Lanczos
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 SpMV (P2P) per iteration as BICGSTAB
Example: stencil with variable coefficients

- Implicit structure, explicit values
- Implicit structure, implicit values

Example: stencil with constant coefficients

Example: general sparse matrix

- Explicit structure, explicit values
- Explicit structure, implicit values

Example: Laplacian matrix of a graph

Hoemmen (2010), Fig 2.5
For $s$ iterations of updates, inner products and SpMVs (in basis $\mathcal{Y}$) can be computed by independently by each processor without communication:
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 $d_{sk+j} \leq \hat{\varepsilon}\|r_{sk+j}\|$ and $d_{sk+j+1} > \hat{\varepsilon}\|r_{sk+j+1}\|$ and $d_{sk+j+1} > 1.1d_{init}$
    
z = z + 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} = y_k p'_{k,j+1}
    
end

break from inner loop and begin new outer loop
```
(2.10) \[ \|r_i\|_2 = \mu_i^{(2)} \|A\|_2 \|x - \widehat{x}_i\|_2. \]

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

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

The bound tells us that \( \mu_i^{(2)} \) will be much less than 1 if \( r_i \) contains a significant component in the subspace \( \text{span}(U_k) \) 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.