Mixed Precision s-step Lanczos and Conjugate Gradient Algorithms

Erin Carson
Charles University

Platform for Advanced Scientific Computing (PASC) Conference 2021
July 7, 2021

We acknowledge funding from Charles Univ. PRIMUS project No. PRIMUS/19/SCI/11, Charles Univ. Research Program No. UNCE/SCI/023, and the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Admin.
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 \( r_i = b - Ax_i \perp C_i \)
Conjugate Gradient Method

$A$ is symmetric positive definite, $C_i = K_i(A, r_0)$

$$r_i \perp K_i(A, r_0) \iff \|x - x_i\|_A = \min_{z \in x_0 + K_i(A, r_0)} \|x - z\|_A$$

$$\Rightarrow \quad r_{n+1} = 0$$
Conjugate Gradient Method

If $A$ is symmetric positive definite, $\mathcal{C}_i = \mathcal{K}_i(A, r_0)$

\[ r_i \perp \mathcal{K}_i(A, r_0) \iff \|x - x_i\|_A = \min_{z \in x_0 + \mathcal{K}_i(A, r_0)} \|x - z\|_A \]

\[ \Rightarrow r_{n+1} = 0 \]

Connection with Lanczos:

• With $v_1 = r_0/\|r_0\|$, $i$ iterations of Lanczos produces $n \times i$ matrix $V_i = [v_1, ..., v_i]$, and $i \times i$ tridiagonal matrix $T_i$ such that

\[ AV_i = V_i T_i + \delta_{i+1} v_{i+1} e_i^T, \quad T_i = V_i^* A V_i \]

• CG approximation $x_i$ is obtained by solving the reduced model

\[ T_i y_i = \|r_0\| e_1, \quad x_i = x_0 + V_i y_i \]
Classical CG

- HSCG: Hestenes and Stiefel (1952)
  - Uses three 2-term recurrences for updating $x_i, r_i, p_i$

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

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
Classical CG

- HSCG: Hestenes and Stiefel (1952)
  - Uses three 2-term recurrences for updating $x_i, r_i, p_i$

\[
\begin{align*}
  r_0 &= b - Ax_0, \quad p_0 = r_0 \\
  \text{for } i &= 1:n_{\text{max}} \\
  \alpha_{i-1} &= \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T A p_{i-1}} \\
  x_i &= x_{i-1} + \alpha_{i-1} p_{i-1} \\
  r_i &= r_{i-1} - \alpha_{i-1} A p_{i-1} \\
  \beta_i &= \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \\
  p_i &= r_i + \beta_i p_{i-1}
\end{align*}
\]
Classical CG

- HSCG: Hestenes and Stiefel (1952)
  - Uses three 2-term recurrences for updating $x_i, r_i, p_i$

\[
\begin{align*}
  r_0 &= b - Ax_0, \quad p_0 = r_0 \\
  &\text{for } i = 1:n_{\text{max}} \\
  \alpha_{i-1} &= \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^TAp_{i-1}} \\
  x_i &= x_{i-1} + \alpha_{i-1}p_{i-1} \\
  r_i &= r_{i-1} - \alpha_{i-1}Ap_{i-1} \\
  \beta_i &= \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \\
  p_i &= r_i + \beta_i p_{i-1} \\
\end{align*}
\]
Classical CG

- HSCG: Hestenes and Stiefel (1952)
  - Uses three 2-term recurrences for updating $x_i, r_i, p_i$

\[
\begin{align*}
r_0 &= b - Ax_0, \quad p_0 = r_0 \\
\text{for } i = 1:n_{\text{max}} \\
\alpha_{i-1} &= \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T Ap_{i-1}} \\
x_i &= x_{i-1} + \alpha_{i-1} p_{i-1} \\
r_i &= r_{i-1} - \alpha_{i-1} Ap_{i-1} \\
\beta_i &= \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \\
p_i &= r_i + \beta_i p_{i-1} \\
\end{align*}
\]
Classical CG

- HSCG: Hestenes and Stiefel (1952)
  - Uses three 2-term recurrences for updating $x_i, r_i, p_i$

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

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
Classical CG

- HSCG: Hestenes and Stiefel (1952)
  - Uses three 2-term recurrences for updating $x_i, r_i, p_i$

\[
\begin{align*}
  r_0 &= b - Ax_0, \quad p_0 = r_0 \\
  \text{for } i &= 1:nmax \\
  \alpha_{i-1} &= \frac{r_{i-1}^T r_{i-1}}{p_{i-1}^T Ap_{i-1}} \\
  x_i &= x_{i-1} + \alpha_{i-1} p_{i-1} \\
  r_i &= r_{i-1} - \alpha_{i-1} Ap_{i-1} \\
  \beta_i &= \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \\
  p_i &= r_i + \beta_i p_{i-1}
\end{align*}
\]
→ Sparse matrix-vector multiplication (SpMV)
  • Must communicate vector entries w/ neighboring processors (P2P communication)

→ Inner products
  • global synchronization (MPI_Allreduce)
  • all processors must exchange data and wait for all communication to finish before proceeding

Dependencies between communication-bound kernels in each iteration limit performance!
<table>
<thead>
<tr>
<th>Rank</th>
<th>System</th>
<th>Rpeak (Tflops/s)</th>
<th>HPCG (Tflops/s)</th>
<th>HPCG %peak</th>
<th>HPL (Tflops/s)</th>
<th>HPL % peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Supercomputer Fugaku, RIKEN, Japan</td>
<td>537,212</td>
<td>16004.50</td>
<td>3.0%</td>
<td>442,010</td>
<td>82.3%</td>
</tr>
<tr>
<td>2</td>
<td>Summit, ORNL, USA</td>
<td>200,794.9</td>
<td>2925.75</td>
<td>1.5%</td>
<td>148,600</td>
<td>74.0%</td>
</tr>
<tr>
<td>3</td>
<td>Perlmutter, LBNL, USA</td>
<td>89,794.5</td>
<td>1905.44</td>
<td>2.0%</td>
<td>64,590</td>
<td>72.0%</td>
</tr>
<tr>
<td>4</td>
<td>Sierra, LLNL, USA</td>
<td>125,712.0</td>
<td>1795.67</td>
<td>1.4%</td>
<td>94,640</td>
<td>75.3%</td>
</tr>
<tr>
<td>5</td>
<td>Selene, NVIDIA, USA</td>
<td>79,215.0</td>
<td>1622.51</td>
<td>2.1%</td>
<td>63,460</td>
<td>80.1%</td>
</tr>
<tr>
<td>6</td>
<td>JUWELS Booster Module, FZJ, Germany</td>
<td>70,980.0</td>
<td>1275.36</td>
<td>1.8%</td>
<td>44,120</td>
<td>62.2%</td>
</tr>
</tbody>
</table>
s-step Krylov subspace methods

- Idea: Compute blocks of $s$ iterations at once
  - Compute updates in a different basis
  - Communicate every $s$ iterations instead of every iteration
  - Reduces number of synchronizations per iteration by a factor of $s$

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 $A\mathbf{y} = \mathbf{yB}$

\[
A p_{i+j} = A \mathbf{y} p'_j = \mathbf{y}(Bp'_j)
\]

\[
(r_{i+j}, r_{i+j}) = r'_j^T \mathbf{y}^T \mathbf{y} r'_j = r'_j^T G r'_j
\]
s-step Krylov subspace methods

- Idea: Compute blocks of $s$ iterations at once
  - Compute updates in a different basis
  - Communicate every $s$ iterations instead of every iteration
  - Reduces number of synchronizations per iteration by a factor of $s$

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 $A\mathbf{y} = \mathbf{y} B$

\[
\begin{align*}
A p_{i+j} & \quad = \quad A \mathbf{y} p_{j}' = \quad \mathbf{y} (B p_{j}') \\
(r_{i+j}, r_{i+j}) & \quad = \quad r_{j}'^T \mathbf{y}^T \mathbf{y} r_{j}' = \quad r_{j}'^T G r_{j}' \\

\end{align*}
\]
s-step CG

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

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

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

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

\[ G_k = y_k^T y_k \]

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

for \( j = 1 : s \)

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

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

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

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

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

end

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

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

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

Compute \( y_k \) and \( B_k \) such that \( A \overline{y_k} = y_k B_k \) and \( \text{span}(y_k) = \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk}) \)

\( G_k = y_k^T y_k \)

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

for \( j = 1 : s \)

\[ \alpha_{sk+j-1} = \frac{r_{j-1}^T G_k r_j}{p_{j-1}^T G_k B_k p_j} \]

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

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

\[ \beta_{sk+j} = \frac{r_j^T G_k r_j}{r_{j-1}^T G_k r_j} \]

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

end

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

e.g., [Van Rosendale, 1983], [Chronopoulos & Gear, 1989], [Toledo, 1995]
\( r_0 = b - Ax_0, p_0 = r_0 \)

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

Compute \( Y_k \) and \( B_k \) such that \( AY_k = Y_kB_k \) and
\[
\text{span}(Y_k) = \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_{s}(A, r_{sk})
\]

\( G_k = Y_k^T Y_k \)

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

for \( j = 1 : s \)

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

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

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

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

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

end

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

end

s-step CG e.g., [Van Rosendale, 1983], [Chronopoulos & Gear, 1989], [Toledo, 1995]
s-step CG e.g., [Van Rosendale, 1983], [Chronopoulos & Gear, 1989], [Toledo, 1995]

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

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

Compute \( y_k \) and \( B_k \) such that \( A\bar{y}_k = \bar{y}_kB_k \) 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 \( j = 1:s \)

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

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

\[ r_j' = r_{j-1}' - \alpha_{sk+j-1} B_k p_{j-1}' \]

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

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

end

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

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

Numerical Example

$A$: bcsstk03 from SuiteSparse,
$b$: equal components in the eigenbasis of $A$, $\|b\| = 1$
$N = 112$, $\kappa(A) \approx 6.8e6$
Finite precision Lanczos process: \((A\) is \(n \times n\) with at most \(N\) nonzeros per row)

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

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

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

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

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

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

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

where \(\sigma \equiv \|A\|_2\), and \(\theta \sigma \equiv \|\|A\|_2\|

Classical Lanczos (Paige, 1976):

\[
\varepsilon_0 = O(\varepsilon n)
\]

\[
\varepsilon_1 = O(\varepsilon N \theta)
\]
Lanczos Convergence Analysis [Paige, 1976]

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

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

$$\hat{V}_m = [\hat{v}_1, ..., \hat{v}_m], \quad \delta\hat{V}_m = [\delta\hat{v}_1, ..., \delta\hat{v}_m], \quad \hat{T}_m = \begin{bmatrix} \alpha_1 & \hat{\beta}_2 & \cdots & \cdots & \cdots & \hat{\beta}_m \\ \hat{\beta}_2 & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ \hat{\beta}_m & \ddots & \ddots & \ddots & \ddots \\ & & & \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}$$

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

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

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

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

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

where $\sigma \equiv \|A\|_2$, and $\theta\sigma \equiv \||A||_2$

Classical Lanczos (Paige, 1976):

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

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

s-step Lanczos (C., 2015):

$$\varepsilon_0 = O(\varepsilon n\Gamma^2)$$

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

$$\Gamma = \max_{\ell \leq k} \|Y_\ell^+\|_2 \cdot \|\|Y_\ell\|\|_2$$
Using bounds on local rounding errors in Lanczos, showed that

1. The computed eigenvalues 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 computed eigenvalues have converged.
Results for s-step Lanczos

• Do Paige’s results, e.g.,
  loss of orthogonality $\rightarrow$ eigenvalue convergence
hold for s-step Lanczos?
Results for s-step Lanczos

• Do Paige’s results, e.g.,
  loss of orthogonality $\rightarrow$ eigenvalue convergence
  hold for s-step Lanczos?

• The answer is YES!
Results for s-step Lanczos

• Do Paige’s results, e.g.,
  loss of orthogonality $\rightarrow$ eigenvalue convergence
  hold for s-step Lanczos?

• The answer is YES! ...but

• Only if:
  • $\varepsilon_0 \equiv 2\varepsilon(n+11s+15) \Gamma^2 \leq \frac{1}{12}$
  • i.e., $\Gamma \leq \left(24\varepsilon(n + 11s + 15)\right)^{-1/2} = O\left(\frac{1}{\sqrt{n\varepsilon}}\right)$
• Do Paige’s results, e.g.,
  loss of orthogonality $\rightarrow$ eigenvalue convergence hold for s-step Lanczos?
• The answer is **YES!** …but

• With the caveat:
• Paige’s results say: orthogonality is not lost until an eigenvalue has stabilized to within $O(\varepsilon)$ of an eigenvalue of $A$
• For s-step Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $O(\varepsilon \Gamma^2)$ within an eigenvalue of $A$
  • So the result is weaker: an eigenvalue is considered to be “stabilized” within a larger radius for the s-step case, and thus orthogonality is lost sooner
    • This explains the worse convergence behavior!
The case for mixed precision

- The term $\Gamma$ enters the bounds due to computation in the computed s-step basis
  - SpMVs cause $\Gamma$ terms in the bounds
  - Inner products (computed using the Gram matrix) cause $\Gamma^2$ terms in the bounds
The case for mixed precision

• The term $\Gamma$ enters the bounds due to computation in the computed s-step basis
  • SpMVs cause $\Gamma$ terms in the bounds
  • Inner products (computed using the Gram matrix) cause $\Gamma^2$ terms in the bounds

• Idea: use higher precision in computing and applying the Gram matrix
  • Computation only happens once every s iterations (doubles the size of the Allreduce)
  • Applying to vector happens every iteration, but the matrix is very small ($s \times s$, fits in cache)
Mixed Precision Lanczos Analysis

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

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

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

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

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

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

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

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

where \(\sigma \equiv \|A\|_2\), and

\[
\theta\sigma \equiv \|\|A\|\|_2
\]

Classical Lanczos
(Paige, 1976):

\[
\varepsilon_0 = O(\varepsilon n)
\]

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

s-step Lanczos
(C., 2015):

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

\[
\varepsilon_1 = O(\varepsilon N\theta\Gamma)
\]

\[\Gamma = \max_{\ell \leq k} \|Y_\ell^+\|_2 \cdot \|\|Y_\ell\|\|_2\]
Mixed Precision Lanczos Analysis

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

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

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

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

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

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

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

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

where $\sigma \equiv ||A||_2$, and $\theta\sigma \equiv |||A|||_2$

Classical Lanczos (Paige, 1976):

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

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

s-step Lanczos (C., 2015):

$$\varepsilon_0 = O(\varepsilon n \Gamma^2)$$

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

Mixed precision s-step Lanczos (C. & Gergelits, 2021):

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

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

$$\Gamma = \max_{\ell \leq k} ||Y_\ell^+||_2 \cdot |||Y_\ell|||_2$$
Mixed precision s-step Lanczos analysis

Classical Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $O(\varepsilon)$ of an eigenvalue of $A$

Uniform precision s-step Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $O(\varepsilon \Gamma^2)$ of an eigenvalue of $A$

Results hold if $\Gamma \leq O \left( \frac{1}{n\varepsilon} \right)$

Mixed precision s-step Lanczos: orthogonality is not lost until an eigenvalue has stabilized to within $O(\varepsilon \Gamma)$ of an eigenvalue of $A$

Results hold if $\Gamma \leq O \left( \frac{1}{n\varepsilon} \right)$

⇒ For mixed precision case, expect orthogonality (and thus convergence behavior) to be somewhere between classical and (fixed precision) s-step Lanczos
⇒ Expect mixed precision algorithm can handle more ill-conditioned bases versus uniform precision algorithm
Diagonal test problem, 
\[ n = 100, \lambda_1 = 10^{-3}, \lambda_n = 10^2 \]
\[ \lambda_i = \lambda_1 + \left(\frac{i-1}{n-1}\right)(\lambda_n - \lambda_1)0.65^{n-i}, \quad i = 2, \ldots, n - 1 \]
Starting vector \( \nu_1 \) has entries \( 1/\sqrt{n} \)
nos4 from SuiteSparse, starting vector $\nu_1$ has entries $1/\sqrt{n}$
nos4 from SuiteSparse, starting vector $\nu_1$ has entries $1/\sqrt{n}$
nos4 from SuiteSparse, starting vector $v_1$ has entries $1/\sqrt{n}$
Extension to s-step CG

• s-step CG based on underlying s-step Lanczos procedure
• Expectation is that better Ritz value accuracy and orthogonality in s-step Lanczos will lead to better convergence behavior of mixed precision s-step CG
• But: extended precision computations in Gram matrix computations will not improve attainable accuracy (this is primarily determined by precision in matrix-vector products)

• Greenbaum (1989): finite precision classical CG behaves like exact CG applied to a larger matrix whose eigenvalues are in tight clusters around the eigenvalues of A.
• Can we extend this analysis?
  • Prediction: Cluster radius will contain a $\Gamma^2$ term for the uniform precision case, $\Gamma$ term for the mixed precision case
Diagonal test problem, \[ n = 100, \lambda_1 = 10^{-3}, \lambda_n = 10^2 \]
\[ \lambda_i = \lambda_1 + \left( \frac{i-1}{n-1} \right) (\lambda_n - \lambda_1) 0.65^{n-i}, \quad i = 2, \ldots, n-1 \]
RHS: equal components in the eigenbasis of \( A \), unit 2-norm
nos4 from SuiteSparse
RHS: equal components in the eigenbasis of $A$, unit 2-norm
lundb from SuiteSparse
RHS: equal components in the eigenbasis of $A$, unit 2-norm
What is the overhead?

- 3D Laplace matrix with $n = 100^3$
- 500 iterations of $s$-step CG with $s = 5$ on a NVIDIA V100 GPU
- Single/double: Uses KokkosBlas::DotBasedGemm for Gram matrix, computes $C = \alpha A^T B + \beta C$
  - Do not compute multiplication with $\alpha$ ($= 1$)
  - Only compute upper triangular part of $C$ since symmetric
  - Input cast to double before being passed in

![Diagram showing time comparison between uniform (single) and mixed (single/double) with values 0.177 and 0.178 for orthogonalization and 0.0780 and 0.0787 for total time.]
What is the overhead?

- 3D Laplace matrix with \( n = 100^3 \)
- 500 iterations of s-step CG with \( s = 5 \) on a NVIDIA V100 GPU
- Single/double: Uses KokkosBlas::DotBasedGemm for Gram matrix, computes \( C = \alpha A^T B + \beta C \)
  - Do not compute multiplication with \( \alpha \) (= 1)
  - Only compute upper triangular part of \( C \) since symmetric
  - Input cast to double before being passed in

- Double/double-double: Software implementation of double-double (each multiply-add operation requires 16 double-precision operations)
  - Since Kokkos does not support double-double arithmetic, the implementation uses a custom reducer for mixed-precision inner products on a GPU
  - For small double-double computations with the Gram matrix, we use multiprecision BLAS on the host CPU
Strong Scaling

- Same problem
- Strong scaling up to 18 GPUs on Summit (6 GPUs per node)
- Using double/double-double

- Overhead of using software-implemented precision decreases as we scale up the hardware
  - Likely because latency becomes more dominant
Conclusions

Big picture idea: Selective use of higher precision can improve numerical behavior (and time to solution) with minimal overhead

For s-step Lanczos and CG:

Overhead is negligible when restricting to precisions available in hardware +

Convergence rate improved =

Likely to see improved time-to-solution in many scenarios
• Performance results are preliminary – a thorough performance study is needed!

• Extending the analysis of Greenbaum for s-step CG

• Benefits to extended precision for other s-step Krylov subspace methods?

• Benefit to mixed precision in pipelined variants?

• Combine mixed precision with residual replacement to also improve accuracy?
Thank you!

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

MATLAB codes: https://github.com/eccarson/mixedsstep