Multiprecision GMRES in Trilinos packages Belos and Kokkos

Jennifer Loe, Christian Glusa, Erik Boman, Ichitaro Yamazaki, Siva Rajamanickam
Why use lower precisions in Krylov solvers?
(e.g. float instead of double)

- Krylov solvers typically memory-bound; reduce cost and time of data movement
- Cheaper floating-point operations
- Take advantage of new hardware for low-precision computations

Challenges:
- Lower precision computations result in more roundoff error!
- Applications still need high level of accuracy in solutions.
Multiprecision Krylov Solvers in Trilinos

- Belos: Linear Solvers package in Trilinos:
  - All linear algebra kernels are abstracted through “adapter” interface.
  - Solvers are templated upon scalar type.
  - Solvers interface does not support mixing precisions! Mixed precision must occur through the adapter.

- Kokkos and Kokkos Kernels:
  - Portable parallel linear algebra
  - Performant BLAS kernels for GPU (single node)

- New Mixed Precision Krylov Solvers Software:
  - New adapter to use Kokkos as the linear algebra backend for solvers
  - Can implement mixed precision operations via the adapter
  - Now: Testing performance improvements on a single node with GPU
  - Future: Make mixed precision solvers available to applications using Trilinos
Iterative Refinement with GMRES (Algorithm)

**Algorithm 1** Iterative Refinement with GMRES Error Correction

1. \( r_0 = b - Ax_0 \) [double]
2. for \( i = 1, 2, \ldots \) until convergence: do
3. Use GMRES(\( m \)) to solve \( Au_i = r_i \) for correction \( u_i \) [single]
4. \( x_{i+1} = x_i + u_i \) [double]
5. \( r_{i+1} = b - Ax_{i+1} \) [double]
6. end for

Using Iterative Refinement with GMRES (single precision) as the correction step; At each restart, update solution vector and recompute residuals in double precision.

Related Work:

- Neil Lindquist, Piotr Luszczek, and Jack Dongarra. *Improving the performance of the GMRES method using mixed-precision techniques.*


- Erin Carson and Nicholas J. Higham. *Accelerating the solution of linear systems by iterative refinement in three precisions.*
Convergence: GMRES(50)

Matrix is 2D convection-diffusion problem over a 5-pt stencil. (Highly nonsymmetric.)
n = 2.25 million, nnz = 11,244,000

Running GMRES(50) to tolerance of 1e-10. (No preconditioning.)
For GMRES-IR: Residuals recomputed in double at each restart (each 50 iterations).
Tests run on a V100 GPU.

Double: 12967 iterations  50.26 seconds
IR: 13150 iterations  38.03 seconds
**Kernel Speedup**

**Double:** 12,967 iterations  
50.26 seconds

**IR:** 13,150 iterations  
38.03 seconds  
(3.98 sec on double precision operations)

(Timings do not include making an extra copy of the matrix $A$ in single precision.)

<table>
<thead>
<tr>
<th>Solver</th>
<th>GMRES double</th>
<th>GMRES IR</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total time:</strong></td>
<td>50.26</td>
<td>38.03</td>
<td>1.322</td>
</tr>
<tr>
<td>Ortho: GEMV Trans</td>
<td>20.20</td>
<td>15.78</td>
<td>1.280</td>
</tr>
<tr>
<td>Ortho: GEMV No Trans</td>
<td>19.01</td>
<td>12.10</td>
<td>1.571</td>
</tr>
<tr>
<td>Ortho (norm)</td>
<td>1.71</td>
<td>1.49</td>
<td>1.152</td>
</tr>
<tr>
<td>$A^*x$</td>
<td>7.33</td>
<td>2.95</td>
<td>2.484</td>
</tr>
</tbody>
</table>

[Using CGS(2) orthogonalization]
Kernel speedups with other matrices

Speedup of GMRES(50)-IR over GMRES(50) double run to convergence of 1e-10.

SpMV gets 2.4 to 2.6 times speedup in float precision.

Total of 30 to 40% speedup over 3 different PDE problems
Choosing a Restart Size – Good speedups for difficult problem

BentPipe 2D convection-diffusion problem with different restart sizes for GMRES.

<table>
<thead>
<tr>
<th>Subspace Size</th>
<th>GMRES Double</th>
<th>GMRES-IR</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iterations</td>
<td>Solve Time</td>
<td>Iterations</td>
</tr>
<tr>
<td>50</td>
<td>12967</td>
<td>50.04</td>
<td>13150</td>
</tr>
<tr>
<td>100</td>
<td>12023</td>
<td>73.94</td>
<td>12000</td>
</tr>
<tr>
<td>150</td>
<td>11250</td>
<td>95.82</td>
<td>12300</td>
</tr>
<tr>
<td>200</td>
<td>10867</td>
<td>117.80</td>
<td>12200</td>
</tr>
<tr>
<td>300</td>
<td>10478</td>
<td>164.00</td>
<td>12600</td>
</tr>
<tr>
<td>400</td>
<td>10236</td>
<td>209.40</td>
<td>12400</td>
</tr>
</tbody>
</table>

We maintain speedups of 20 to 44% over several subspace sizes.
Considerations for choosing a restart size:

For large problems where memory limitations necessitate a small restart size, GMRES-IR gives improvement. However, it may need longer to converge with infrequent restarts.
Polynomial Preconditioning

Stretched2D: Laplacian on a stretched grid.

Polynomial preconditioner based upon the GMRES polynomial.**

Double precision GMRES(50) with double precision preconditioner. GMRES(50)-IR with single precision preconditioner.

Convergence tolerance = $1 \times 10^{-10}$

(Solve time does not include preconditioner creation.)

GMRES-IR gives consistent solve time improvement ~33% over double precision with preconditioning.

Polynomial preconditioning shifts main expense to SpMV rather than dense orthogonalization kernels.

**For polynomial preconditioning details, see: Jennifer Loe, Erik Boman, and Heidi Thornquist. *Polynomial Preconditioned GMRES in Trilinos: Practical Considerations for High-Performance Computing*
Conclusions and Future Work

• GMRES(50)-IR gives improved performance over double precision GMRES(50)
• Speedups in Orthogonalization and SpMV – Impact depends on preconditioning scheme
• Using Trilinos library (Kokkos backend)

Future:
• Choice of restart size for GMRES
• Test other Krylov methods
• More preconditioning options (Block Jacobi, etc….)
• Half precision (esp. Gemm in Block Jacobi)
• Low-precision multigrid preconditioning (Christian Glusa)
• Make available to applications using Trilinos