Scalable Parallel Newton-Krylov Solvers for Discontinuous Galerkin Discretizations

Per-Olof Persson

Department of Mathematics
University of California, Berkeley

47th AIAA Aerospace Sciences Meeting, Orlando, Florida

January 6, 2009
Motivation

- Low order methods perform poorly for problems where high numerical accuracy is required
  - Wave propagation (e.g. aeroacoustics)
  - Turbulent flow (e.g. drag & transition prediction)
  - Non-linear interactions (e.g. fluid-structure coupling)
- High order discontinuous Galerkin methods attractive options:
  - Low dissipation, stabilization, complex geometries
- Parallel computers required for realistic problems because of high computational and storage costs with DG
Implicit Time Integration

- Implicit solvers typically required because of CFL restrictions from viscous effects, low Mach numbers, and adaptive/anisotropic grids
- Jacobian matrices are large even at $p = 2$ or $p = 3$, however:
  - They are required for non-trivial preconditioners
  - They are very expensive to recompute
- Therefore, we consider matrix-based Newton-Krylov solvers
- In [Persson/Peraire, SISC’08], we proposed efficient preconditioners with block-ILU(0) and automatic element ordering
- Our goal is to develop distributed parallel versions of these methods
Problem Formulation

- Consider time-dependent systems of conservation laws:
  \[
  \frac{\partial u}{\partial t} + \nabla \cdot F(u, \nabla u) = S(u, \nabla u)
  \]

- Standard DG discretization using degree \( p \) simplex elements

- Approximate Riemann solver for inviscid fluxes, the Compact Discontinuous Galerkin (CDG) method for viscous fluxes [Peraire/Persson, SISC’08]

- Leads to system of ODEs of the form \( M\dot{u} = r(u) \)

- Implicit BDF or DIRK methods with Newton’s method lead to linear systems of the form
  \[
  A \equiv \alpha_0 M - \Delta t \frac{\partial r}{\partial u} \equiv \alpha_0 M - \Delta t K
  \]

- Strong dependence on the timestep \( \Delta t \)
- Dual mesh connectivity, with each entry a large complete graph $A_{ij}$
- Off-diagonal blocks actually sparser with CDG, but assume dense for simplicity
- Size $N$ of submatrices $A_{ij}$ is often $\geq 100$
- Block-based storage format essential for high performance using BLAS routines
- For other structures (e.g. elimination matrices), use block-wise compressed column
Preconditioners for Krylov Methods

- Preconditioning required for fast convergence in Krylov methods
- Standard point-wise Jacobi, ILU, etc, ineffective for DG
- Block Jacobi and Gauss Seidel are generally poor:

\[
\tilde{A}_{ij}^{J} = \begin{cases} 
A_{ij} & \text{if } i = j, \\
0 & \text{if } i \neq j,
\end{cases}
\]

and

\[
\tilde{A}_{ij}^{GS} = \begin{cases} 
A_{ij} & \text{if } i \leq j, \\
0 & \text{if } i > j.
\end{cases}
\]

- Block-ILU(0) algorithm \(\tilde{A}^{ILU} = \tilde{L}\tilde{U}\) effective for good orderings
- Block-ILU(0) postsmoothing for coarse scale correction

[Persson/Peraire ’08], cheap, general purpose preconditioner
Computational Cost

- Pre-calculation cost (block-size \( N \), dimension \( D \), \( ne \) elements):

<table>
<thead>
<tr>
<th>Operation</th>
<th>Flop count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi Factorization</td>
<td>( (2/3)N^3ne )</td>
</tr>
<tr>
<td>Gauss Seidel Factorization</td>
<td>( (2/3)N^3ne )</td>
</tr>
<tr>
<td>ILU(0) Factorization</td>
<td>( (2D + 8/3)N^3ne )</td>
</tr>
</tbody>
</table>

- This assumes full blocks (CDG is significantly cheaper)

- Cost of assembling a nonlinear \( A \) is \( O(N^3ne) \) with large constant

- Cost per iteration:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Flop count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi solve ( (\tilde{A}^J)^{-1}p )</td>
<td>( (2)N^2ne )</td>
</tr>
<tr>
<td>Gauss Seidel solve ( (\tilde{A}^{GS})^{-1}p )</td>
<td>( (D + 3)N^2ne )</td>
</tr>
<tr>
<td>ILU(0) solve ( (\tilde{A}^{ILU})^{-1}p )</td>
<td>( (2D + 4)N^2ne )</td>
</tr>
<tr>
<td>Matrix-vector product ( Ax )</td>
<td>( (2D + 4)N^2ne )</td>
</tr>
</tbody>
</table>

- Only factor of 2 between ILU + MATVEC and Jacobi + MATVEC
Properties of Gauss Seidel and ILU preconditioners highly dependent on the ordering of the elements

For *upwinded scalar convection*, “ordering by lines” gives an optimal upper triangular matrix

But for viscous or multivariate problems, best ordering not clear

Matrix-approach: Minimize error in the approximations, rather than using physical observations
Greedy algorithm for element ordering [Persson/Peraire ’08]: At step \( j \), if \( j' \) is chosen next, we would discard the fill

\[
\Delta \tilde{U}^{(j,j')}_{ik} = -\tilde{U}_{ij'} \tilde{U}_{j'j'}^{-1} \tilde{U}_{j'k}, \quad \text{for neighbors } i \geq j, k \geq j \text{ of element } j'
\]

Choose the \( j' \) that minimizes the norm of the discarded fill

\[
w^{(j,j')} = \| \Delta \tilde{U}^{(j,j')} \|_F
\]

Some simplifications, min-heap data structure \( \Rightarrow O(n \log n) \) cost

Increased locality: Consider only neighbors for \( j' \)
Effect of Ordering on ILU

- MDF ordering makes block-ILU0 with coarse grid correction almost perfect for convection-diffusion
- Good element ordering critical for Navier-Stokes as well:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Element Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Random</td>
</tr>
<tr>
<td>Inviscid</td>
<td>51</td>
</tr>
<tr>
<td>Laminar, Re=1,000</td>
<td>200</td>
</tr>
<tr>
<td>Laminar, Re=20,000</td>
<td>197</td>
</tr>
<tr>
<td>RANS, Re=10^6</td>
<td>98</td>
</tr>
</tbody>
</table>
No convergence after 1,000 iterations, from [Persson/Peraire '08]

<table>
<thead>
<tr>
<th>Problem</th>
<th>Parameters</th>
<th>Preconditioner/Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Delta t$</td>
<td>$M$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inviscid</td>
<td>$10^{-3}$</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$10^{-1}$</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$10^{-3}$</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$10^{-1}$</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>0.01</td>
</tr>
<tr>
<td>Laminar</td>
<td>$10^{-3}$</td>
<td>0.2</td>
</tr>
<tr>
<td>Re=1,000</td>
<td>$10^{-1}$</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$10^{-3}$</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$10^{-1}$</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>0.01</td>
</tr>
</tbody>
</table>
No convergence after 1,000 iterations, from [Persson/Peraire ’08]

<table>
<thead>
<tr>
<th>Problem</th>
<th>Parameters</th>
<th>Preconditioner/Iterations</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Block Jacobi</td>
<td>Block G-S</td>
<td>Block ILU0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>∆t, M</td>
<td>BJ</td>
<td>BJ-p1</td>
<td>BGS</td>
<td>BGS-p1</td>
<td>BILU0</td>
</tr>
<tr>
<td>Laminar</td>
<td>10^{-3} 0.2</td>
<td>26</td>
<td>19</td>
<td>14</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>Re=20,000</td>
<td>10^{-1} 0.2</td>
<td>456</td>
<td>220</td>
<td>219</td>
<td>113</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>∞ 0.2</td>
<td>× ×</td>
<td>× ×</td>
<td>× ×</td>
<td>236</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>10^{-3} 0.01</td>
<td>160</td>
<td>90</td>
<td>61</td>
<td>38</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10^{-1} 0.01</td>
<td>× ×</td>
<td>× ×</td>
<td>× 735</td>
<td>80</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>∞ 0.01</td>
<td>× ×</td>
<td>× ×</td>
<td>× ×</td>
<td>× 35</td>
<td></td>
</tr>
<tr>
<td>RANS</td>
<td>10^{-3} 0.2</td>
<td>76</td>
<td>56</td>
<td>33</td>
<td>28</td>
<td>8</td>
</tr>
<tr>
<td>Re=10^{6}</td>
<td>10^{-1} 0.2</td>
<td>× ×</td>
<td>× ×</td>
<td>× ×</td>
<td>35</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>∞ 0.2</td>
<td>× ×</td>
<td>× ×</td>
<td>× ×</td>
<td>70</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>10^{-3} 0.01</td>
<td>411</td>
<td>231</td>
<td>174</td>
<td>110</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>10^{-1} 0.01</td>
<td>× ×</td>
<td>× ×</td>
<td>× ×</td>
<td>46</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>∞ 0.01</td>
<td>× ×</td>
<td>× ×</td>
<td>× ×</td>
<td>132</td>
<td>28</td>
</tr>
</tbody>
</table>
Parallelization of Residual Evaluation

- Two critical properties for parallelization:
  - High computational cost per degree-of-freedom
  - Memory accessed in localized chunks
- Essentially perfect speed-up of residual calculation in MPI
- Communication not an issue with overlapping computations
- However, memory bandwidth limitations for 8 cores on single node

![Graph showing speedup vs. number of processes](image)
Serial nature of Gaussian elimination – Hard to parallelize ILU(0)

Standard approach: *Color* the graph to find independent sets

Elements of the same color can be updated independently

But this is typically a poor ILU-ordering for convective problems!

The coloring eliminates the sequential dependencies found by the MDF ordering
Better approach: Partition-wise ILUs with MDF ordering

Partition using the weights $C_{ij} = \|A_{ii}^{-1}A_{ij}\|_F$

Essentially a “non-overlapping Schwartz preconditioner with incomplete solutions”

Approaches Jacobi as # partitions $\rightarrow$ # elements

Good option for many problems – GMRES iterations cheap compared to matrix creation
Convergence Properties

- SD7003 test problem: $\text{Re} = 8000$, $M = 0.1$, CFL $\Delta t < 10^{-6}$
- Coloring essentially destroys the excellent convergence of MDF
- Partition-wise ILUs scale better, only 2-3 times slower for large $\Delta t$
Assembly scales almost perfectly, as expected
Linear solver scales poorly up to the 8 cores in each compute node, due to increased iterations and limited memory bandwidth
Beyond the 8 cores, the scaling per compute node is excellent
Partition-wise block-ILU outperforms block-Jacobi even at moderate timesteps $\Delta t$

Jacobi’s good scalability does not compensate for its poor preconditioning properties
Elliptic Wing Problem

- Elliptic wing, $\text{Re} = 2000$, $M = 0.3$, $\text{AoA} = 30^\circ$
- 190,000 tetrahedral elements, $p = 3$, about 19 million DOFs
- Time integration by 2-stage, 3rd order accurate DIRK scheme
- Jacobian storage 36GB, no comparison with serial possible
The Elliptic Wing Problem – Timings and Speedup

- Almost perfect speedup per compute node with 512 processes
- Solution time dominated by assembly for small $\Delta t$
Conclusions

- Block-ILU superior to block-Jacobi and Gauss Seidel
- Element ordering critical for high performance
- Good orderings found by our Minimum Discarded Fill algorithm
- Graph coloring poor for ILU
- Domain decomposition ILU performs well for many problems, in particular with weighted partitionings
- Excellent scaling in general, but multicore architectures pose new challenges on algorithms and memory usage
- Current work: Applications, parallelization of our ILU/coarse grid correction preconditioner, specialized solvers for multicore nodes