A General Implicit Artificial Boundary Scheme for Chimera Methods

Dr. Marshall Galbraith
Massachusetts Institute of Technology
Department of Aeronautics and Astronautics
galbramc@mit.edu

Dr. Robert Knapke
University of Cincinnati
School of Aerospace Systems
knapkerd@email.uc.edu

Dr. Paul D. Orkwis
University of Cincinnati
School of Aerospace Systems
Paul.Orkwis@uc.edu

Dr. John Benek
Air Force Research Laboratory
Computational Science Branch Center of Excellence
John.Benek@wpafb.af.mil

12th Overset Symposium, 9 Oct 2014
Motivation

- Chimera Overset Grid Method
  - Complex Geometries
  - “Hot swap” Geometric Features
  - Moving Grids with Relative Motion
    - Store Separation
    - Rotorcraft

- Explicit Artificial Boundaries
  - Solve Decoupled System
  - Limits CFL Number with Increasing Number of Processor

- Implicit Artificial Boundaries
  - Significant Increased Parallel Performance
  - It’s easier than it sounds
Outline

• Discretization Assumptions
• Explicit/Implicit Chimera
• Sparse Iterative Solvers
  – Preconditioners
• Distributed Memory Parallelism
• Discontinuous Galerkin Method
• Inviscid/Viscous Flow Examples
• Conclusion and Future Work
Outline

• Discretization Assumptions
• Explicit/Implicit Chimera
• Sparse Iterative Solvers
  – Preconditioners
• Distributed Memory Parallelism
• Discontinuous Galerkin Method
• Inviscid/Viscous Flow Examples
• Conclusion and Future Work
Discretization Assumptions

- Euler/Navier-Stokes Equations in Conservation Form

\[ \nabla \cdot \vec{F}(Q) = 0 \]

- Discrete Form
  - Finite Difference
  - Finite Volume
  - Finite Element

- Newton's Method

\[ \frac{\partial R(Q)}{\partial Q} \Delta Q = -R(Q) \]

\[ A \Delta Q = -R(Q) \]

- Chimera Interpolation Operator
  - Linear Operator (don’t think 2nd order accuracy)
  - Polynomial Basis Functions
  - Radial Basis Functions
  - Trigonometric Basis Functions
  - etc.

\[ I_h(Q) = \sum Q_i \phi_i \]

\[ \frac{\partial I_h(Q)}{\partial Q} \Delta Q = I_h(\Delta Q) = \sum \Delta Q_i \phi_i \]
Explicit/Implicit Artificial Boundaries

Newton’s Method
\[ A \Delta Q = R \]

\[ R^1(Q^1, I_h(Q^2)) = 0 \quad R^2(Q^2, I_h(Q^1)) = 0 \]

Explicit Artificial Boundary
\[
\begin{bmatrix}
  A^1 & 0 \\
  0 & A^2
\end{bmatrix}
\begin{bmatrix}
  \Delta Q^1 \\
  \Delta Q^2
\end{bmatrix} =
\begin{bmatrix}
  R^1(Q^1, I_h(Q^2)) \\
  R^2(Q^2, I_h(Q^1))
\end{bmatrix}
\]

\[ A^1 = \frac{\partial R^1}{\partial Q^1} \quad A^2 = \frac{\partial R^2}{\partial Q^2} \]

Solve Decoupled System
\[ A^1 \Delta Q^1 = R^1(Q^1, I_h(Q^2)) \]
\[ A^2 \Delta Q^2 = R^2(Q^2, I_h(Q^1)) \]

Implicit Artificial Boundary
\[
\begin{bmatrix}
  A^1 & C^1 \\
  C^2 & A^2
\end{bmatrix}
\begin{bmatrix}
  \Delta Q^1 \\
  \Delta Q^2
\end{bmatrix} =
\begin{bmatrix}
  R^1(Q^1, I_h(Q^2)) \\
  R^2(Q^2, I_h(Q^1))
\end{bmatrix}
\]

\[ C^1 = \frac{\partial R^1}{\partial Q^2} \quad C^2 = \frac{\partial R^2}{\partial Q^1} \]

- **Unstructured A Matrix**
  - Explicitly add C_i Matrices

- **Structured A^i Matrix**
  - Tri-, Penta-, Hepta-diagonal

- **Sparse Iterative Solver**
  - No Explicit C_i Matrices
Sparse Iterative Solvers

- Iterative methods for sparse linear systems

- Restarted GMRES
  - Simple Fortran Code Available
  - [http://people.sc.fsu.edu/~jburkardt/f_src/mgmres/mgmres.html](http://people.sc.fsu.edu/~jburkardt/f_src/mgmres/mgmres.html)

- Fundamental Operations
  - Dot products
  - Sparse Matrix-Vector Multiplication

- Slow without Preconditioner
Sparse Iterative Solvers Preconditioners

- Incomplete LU
- ARC3D Beam-Warming block tridiagonal scheme.
- F3D Steger-Warming 2-factor scheme.
- ARC3D diagonalized Beam-Warming scalar pentadiagonal scheme.
- LU-SGS algorithm.
- D3ADI algorithm with Huang subiteration.
- ARC3D Beam-Warming with Steger-Warming flux split jacobians.
- SSOR algorithm (with subiteration)
Implicit Artificial Boundaries

Implicit Artificial Boundary

\[
\begin{bmatrix}
A^1 & C^1 \\
C^2 & A^2
\end{bmatrix}
\begin{bmatrix}
\Delta Q^1 \\
\Delta Q^2
\end{bmatrix} =
\begin{bmatrix}
R^1(Q^1, I_h(Q^2)) \\
R^2(Q^2, I_h(Q^1))
\end{bmatrix}
\]

\[ C^1 = \frac{\partial R^1}{\partial Q^2}, \quad C^2 = \frac{\partial R^2}{\partial Q^1} \]

Artificial Boundary Linearization

\[ C^1 \Delta Q^2 = \frac{\partial R^1(Q^1, I_h(Q^2))}{\partial Q^2} \Delta Q^2 \]

Chain Rule

\[ C^1 \Delta Q^2 = \frac{\partial R^1(Q^1, I_h(Q^2))}{\partial I_h(Q^2)} \frac{\partial I_h(Q^2)}{\partial Q^2} \Delta Q^2 \]

Linear Interpolation Operator

\[ \frac{\partial I_h(Q^2)}{\partial Q^2} \Delta Q^2 = I_h(\Delta Q^2) \]

GMRES: Matrix-Vector Multiplication

Interior Flux Linearization

\[ \frac{\partial R_i(Q^i_L, Q^i_R)}{\partial Q_L} \quad \frac{\partial R_i(Q^i_L, Q^i_R)}{\partial Q_R} \]

Artificial Boundary Linearization

\[ C^1 \Delta Q^2 = \frac{\partial R^1(Q^1, I_h(Q^2))}{\partial I_h(Q^2)} \Delta Q^2 \]

Matrix-Vector Product

\[ C^1 \Delta Q^2 = \frac{\partial R^1(Q^1, I_h(Q^2))}{\partial Q^1_{R_i}} I_h(\Delta Q^2) = \tilde{C}^1 I_h(\Delta Q^2) \]

Receiver Grid Interpolation Mapping

Array of Matrices

Array of Vectors

Interior Jacobian

RHS Interpolation Operator
Parallel GMRES Iterative Solver

Matrix-Vector Multiplication
Mask Communication with Local Calculations

\[ w_{n+1} = A v_n \]

Processor 1

\[
\begin{bmatrix}
  w_{n+1}^1 \\
  w_{n+1}^2
\end{bmatrix} =
\begin{bmatrix}
  A^1 & \bar{C}^1 \\
  \bar{C}^2 & A^2
\end{bmatrix}
\begin{bmatrix}
  v_n^1 \\
  v_n^2
\end{bmatrix}
\]

Processor 2

Interpolate \( I_h(v_n^1) \)

Non-Blocking Send \( I_h(v_n^1) \)

Compute \( w_{n+1}^1 = A^1 v_n^1 \)

Receive \( I_h(v_n^2) \)

Compute \( w_{n+1}^1 = w_{n+1}^1 + \bar{C}^1 I_h(v_n^2) \)

Compute \( w_{n+1}^2 = w_{n+1}^2 + \bar{C}^2 I_h(v_n^1) \)

Interpolate \( I_h(v_n^2) \)

Non-Blocking Send \( I_h(v_n^2) \)

Compute \( w_{n+1}^2 = A^2 v_n^2 \)

Receive \( I_h(v_n^1) \)

Dot Products

Compute Local Dot Product ➔ All Reduce

\[ O \left( \frac{n}{p} + \log(p) \right) \]

Parallel Efficient If

\[ \frac{n}{p} > \log(p) \]

Preconditioner Omits C Matrices

Jacobi as \( p \rightarrow n \)
Outline

• Discretization Assumptions
• Explicit/Implicit Chimera
• Sparse Iterative Solvers
  – Preconditioners
• Distributed Memory Parallelism
• Discontinuous Galerkin Method
• Inviscid/Viscous Flow Examples
• Conclusion and Future Work
**Discontinuous Galerkin Chimera Scheme**

- **Discontinuous Galerkin**
  - Weak Form
    \[ \int_{\Omega_e} \phi \nabla \cdot \vec{F} \, d\Omega = 0 \]
    \[ \phi - \text{Legendre Polynomials} \]
    
    \[ R(Q^+, Q^-) = \int_{\Gamma_e} \phi \vec{F}(Q^+, Q^-) \cdot \vec{n} \, d\Gamma - \int_{\Omega_e} \nabla \phi \cdot \vec{F}(Q^-) \, d\Omega = 0 \]
  - Approximate Riemann Solver by Roe
  - BR2 Viscous Scheme

- **DG-Chimera**
  - Natural Interpolation Operator (Solution is Polynomials)
  - Curved Elements
  - Reduces to a Zonal Interface (Abutting Meshes)
  - No Orphan Points due to Fringe Points
Outline

• Discretization Assumptions
• Explicit/Implicit Chimera
• Sparse Iterative Solvers
  – Preconditioners
• Distributed Memory Parallelism
• Discontinuous Galerkin Method
• Inviscid/Viscous Flow Examples
• Conclusion and Future Work
Inviscid/Viscous Flow Examples

- **Inviscid SKF 1.1 Airfoil**
  - \( M_\infty = 0.4 \)
  - \( \alpha = 2.5^\circ \)

- **Viscous Subsonic Circular Cylinder**
  - \( M_\infty = 0.25 \)
  - \( Re = 40 \)

- **Focus on Solution Time**
  - Explicit vs. Implicit Chimera
Time Integration and Compute Resources

- **Steady State**
  - Quasi-Newton
  
  \[
  \left( \frac{M}{\Delta t} + \frac{\partial R}{\partial Q} \right) \Delta Q = R
  \]

- **GMRES Krylov Solver**
  - ILU1 Preconditioner
  - Converged to $1e^{-11}$ Each Newton Iteration

- **Intel Core 2 Duo 3.0 GHz processor 8 GB RAM**
  - 10 Compute Nodes
  - Ethernet Connection

- **MPI Parallelism**
  - Timings for 1, 2, 4, and 8 Processors
  - 1 MPI Process per Node (Maximize Communication)

- **Shared Memory Multi-Threaded**
  - 1 Grid Per Thread

\[
CFL^{n+1} = CFL^0 \frac{||R^0||}{||R^n||}
\]

\[
CFL_{max} = 1e30
\]
SKF 1.1 Airfoil ($M_\infty = 0.4$, $\alpha=2.5^\circ$)

Meshes

- Single Mesh: $105 \times 30 N_g=3$
- O-Grid Chimera Mesh: $104 \times 14 N_g=1$
- R-Grid Chimera Mesh: $104 \times 104 N_g=1$

Chimera Interface
SKF 1.1 Airfoil ($M_\infty = 0.4$, $\alpha=2.5^\circ$)

Explicit Chimera Speedup

\[ A(Q_{Local}) \Delta Q = R(Q_{Local}, Q_{Chimera}) \]
SKF 1.1 Airfoil ($M_\infty = 0.4$, $\alpha=2.5^\circ$)
Explicit Chimera Convergence History

$$A(Q_{\text{Local}})\Delta Q = R(Q_{\text{Local}}, Q_{\text{Chimera}})$$
SKF 1.1 Airfoil ($M_\infty = 0.4$, $\alpha=2.5^\circ$) Implicit Chimera Speedup

\[ A(Q_{Local}, Q_{Chimera})\Delta Q = R(Q_{Local}, Q_{Chimera}) \]
SKF 1.1 Airfoil ($M_\infty = 0.4$, $\alpha = 2.5^\circ$)
Implicit Convergence History

$$A(Q_{\text{Local}}, Q_{\text{Chimera}}) \Delta Q = R(Q_{\text{Local}}, Q_{\text{Chimera}})$$
SKF 1.1 Airfoil ($M_\infty = 0.4$, $\alpha=2.5^\circ$) Solution Time

\begin{equation}
A(Q_{Local})\Delta Q = R(Q_{Local}, Q_{Chimera})
\end{equation}

\begin{equation}
A(Q_{Local}, Q_{Chimera})\Delta Q = R(Q_{Local}, Q_{Chimera})
\end{equation}
SKF 1.1 Airfoil ($M_\infty = 0.4$, $\alpha = 2.5^\circ$) Implicit/Explicit Solution Speedup
SKF 1.1 Airfoil ($M_\infty = 0.4$, $\alpha = 2.5^\circ$)

Cp Contour Lines

N=0
1\textsuperscript{st}-order

N=1
2\textsuperscript{nd}-order

N=2
3\textsuperscript{rd}-order

N=3
4\textsuperscript{th}-order
Circular Cylinder (Re = 40) Meshes

- Single 50x40 $N_g=3$
- O-Grid Chimera 50x22 $N_g=1$
- R-Grid Chimera 100x100 $N_g=1$

Meshes
Circular Cylinder (Re = 40)
Solution Time

\[ A(Q_{Local}) \Delta Q = R(Q_{Local}, Q_{Chimera}) \]

10 days!
Circular Cylinder (Re = 40)
Implicit/Explicit Solution Speedup
Subsonic Circular Cylinder (\(M_\infty = 0.25\))

Cp/Entropy Rise Contour Lines

- **N=1**
  - 2\(^{nd}\)-order
- **N=2**
  - 3\(^{rd}\)-order
- **N=3**
  - 4\(^{th}\)-order
Conclusion and Future Work

• Implicit Artificial Boundaries
  – Included with GMRES Matrix-Vector Multiplication
  – Omitted in Preconditioner
  – Minimal Information Communicated
  – Significantly Reduces Execution Time

• Few Modifications Required to Existing Codes
  – ~95% of Code already Exists
  – Spares Matrix-Vector Multiplication
  – Restarted GMRES Fortran Code
    • [Link](http://people.sc.fsu.edu/~jburkardt/f_src/mgmres/mgmres.html)

• Demonstrated on Inviscid/Viscous Flows
Thank you!
Questions?

This work was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate (NDSEG) Fellowship Program.