An Interface Driven Approach For Overset Domain Assembly
And Unstructured Mesh Adaptation via T-Infinity

Cameron T. Druyor
NASA Langley Research Center
• “NASA should develop and maintain an integrated simulation and software development infrastructure to enable rapid CFD technology maturation.”
  - CFD 2030 Vision Report Programmatic Recommendation #2

• We are attempting to guide new software development efforts and add benefits to existing software tools while keeping this recommendation in mind
Motivation

• Software should scale!
  – Scale with computer resources (Parallel Scalability)
  – Scale with human resources (Human Scalability)

• Challenges:
  – Problems are so complex, no one can be an expert in every aspect
  – Experts solving problems from a different domain of expertise is suboptimal

• Has someone else encountered this software complexity problem?
Basic Software Engineering Concepts

- **SOLID Design Principles**
  - **Single Responsibility Principle**
    - Software entities should have only one reason to change
  - **Open Closed Principle**
    - Software entities should be open for extension, but closed for modification
  - **Liskov Substitution Principle**
    - Subtypes must be substitutable for their base types.
  - **Interface Segregation Principle**
    - Clients should not be forced to depend on methods that they do not use
  - **Dependency Inversion Principle**
    - High level modules should not depend on low level modules, both should depend on abstractions
Traditional Approach

- What happens when Fluid Solver A needs to support Zoltan?
- What about Fluid Solver C?
- What about:
  - Coupling with Structural Solver A,B,C,D
  - Loads transfer between aero & structural mesh
  - Mesh deformation
  - Solution initialization
  - In-situ mesh slicing, filtering, sub-selecting
  - Overset assembly
  - Metric calculation
  - Mesh adaptation

Can’t share anything!
Interface Driven Approach

Fluid Solver A

Fluid Solver B

PreProcessor Interface

Plugin
- ParMETIS
- AFLR3 reader

Plugin
- Zoltan
- CGNS reader

Capability developed for Solver B also available to Solver A
Good Fences Make Good Neighbors

Changes to B,C,D,E can break A!

A is orthogonal to changes in B,C,D,E
T-infinity is Only Abstractions

- Visualization
- Preprocessing
- Fluid Solver
- Mesh Adaptation
- Loads Transfer
- Adaptation Metric Calculation
- Structural Solver
- Overset Domain Assembly
- Mesh Deformation
- Mesh Deformation
Unifying inputs and outputs gives flexibility
T-infinity: Our Approach

- Component interfaces
  - Interfaces defined in C, C++, Fortran
  - Plugin can be written in language of choice
- An optional library
  - Written in C++, also callable from Python
  - Simplifies dynamic plugin loading
  - Automatically wraps non-C++ plugins for seamless mixed-language interaction
- Provides tools that operate on T-Infinity objects
  - Rigid mesh transformations
  - Mesh filtering
  - Etc.
Goals for Interfaces

- Minimize number of functions for each component
- Minimize complexity of each function call
- Apply Open-Closed Principle
  - Open for extension, Closed for modification

T-infinity: The Dependency Inversion Principle for Rapid and Sustainable Multidisciplinary Software Development

Fluid Solver Interface

- Must be constructible given a Mesh and MPI comm
  
- 2 functions for putting data into the solver:
  - What variables do you expect in the solution?
  - Given a node id and solution, set it for the node

- 1 function to perform steady state solve

- 2 functions for getting data out of the solver:
  - What scalars can you provide at each node?
  - Given node id and scalar name, return value
Simple Application: Steady State CFD

- Inviscid Analysis
- Fluid Solver: FUN3D
- Visualization & Preprocessing: Parfait
Fluid Solver Interface (Additions for Overset Support)

- Given a set of node ids, “freeze” the solution
- Given a cell id, return the solution at a point within that cell
Notional Heavy Lift Configuration on Overset Grid System

- Three domains
- Fluid Solver: HyperSolve (Prototype Finite Volume)
- Domain Assembler: YOGA
- Visualization & Preprocessing: Parfait

![Diagram of Notional Heavy Lift Configuration on Overset Grid System](image)
Adding Dynamic Mesh Adaptation

Additional components
- Metric calculation
- Mesh adaptation
- Solution interpolation
Open-Closed Principle

- Components **unaware** that the simulation is Overset
  - Fluid Solver
  - Metric Calculator
  - Mesh Adaptation
  - Mesh-to-Mesh Interpolation
  - Pre-Processor
  - Visualization

- Components **aware** that the simulation is Overset
  - Domain Assembler

What, if any, are the consequences?

Are the current interfaces suitable for this scenario without modification?
10 Adaptation Cycles

Artifacts appear at Overset boundaries
10 Adaptation Cycles

Density contours match across Overset boundaries, but over refinement at boundaries is wasted effort.
Plumes after 10 Adaptation Cycles

- Metric attacks the Overset boundaries
- Plumes get smeared out
Possible Solution Paths

• Problem:
  • Refinement in frozen regions is wasteful
  • Prefer to leave intact, or ideally coarsen

• Possible approaches:
  • Add freezing to metric calculator interface or adaptation interfaces
  • Write a tool combines metric fields
    • Combine metrics according to a weight at each node
    • Set weight to 0 in frozen regions
  • Set the Overset Domain Assembler to interpolate all frozen regions if possible
10 Adaptation Cycles (Max receptor layers)

• Using more receptors yields a smoother metric
• Artifacts at Overset boundaries reduced
10 Adaptation Cycles (Max receptor layers)

Density contours start setting up earlier in the simulation
10 Adaptation Cycles (Max receptor layers)

- Artifacts at Overset boundaries reduced
- Anisotropy propagates between grids faster
- Remaining issues?
Future Work

• T-Infinity interfaces
  • Add node freezing concept to Metric Calculator interface (or find alternative)
  • Add more “cell-centered” support
  • Extend Fluid Solver interface to account for time-accurate simulations
  • Extend Mesh interface for moving body applications (grid speed, etc.)
  • Release open-source, along with library

• Plugins
  • Implement extra Overset functions for Fun3D plugin
  • Make Fun3D plugin reentrant to enable adaptation
  • 3rd party plugins?
Software Availability

- Apache 2
  - T-Infinity interfaces
  - T-Infinity library
  - Parfait
  - Yoga
  - Refine

- Part of Fun3D
  - Fun3D plugin
  - SFE
More details on interfaces in AIAA paper:

T-infinity: The Dependency Inversion Principle for Rapid and Sustainable Multidisciplinary Software Development