Strategies for OVERFLOW Modularization and Integration into HELIOS

Rohit Jain  Mark Potsdam
Aeroflightdynamics Directorate (AMRDEC)
U. S. Army Research Development and Engineering Center
Moffett Field, California
Outline

• HELIOS Architecture
• Motivation
• Approach for Modularization and Integration
• Demonstration Examples
• Summary & Ongoing Work
• High-fidelity, automated software tool for rotorcraft aeromechanics modeling

• Sponsored by DoD HPCMO (CREATE-AV) and the US Army

• A framework – Flexibility, Extensibility, Modularity

• GOALS: Accuracy, Speed, Ease-of-use (automation)
Object Oriented Python Integration Framework

DCF
Domain Connectivity

FSI
Fluid Structure Interface

PUNDIT

Rotor-FSI

Module Interfaces

Mesh Motion

Near-Body
CFD Solver

Off-Body
CFD Solver

Computational
Structural Dynamics

Co-Visualization

NSU3D
(U. Wyoming)

SAMARC
(LLNL/NASA Ames)

RCAS
(AFDD, US Army)
CAMRADII
(Johnson Aeronautics)

ParaView
FieldView

Distributed Memory processors communicating via MPI

shared data
HELIOS Dual-mesh Approach

✓ **Unstructured** mesh solver for near-body regions
  - Fuselage, hub, blades
  - Direct CAD to CFD mesh generation

✓ **Cartesian** adaptive mesh solver for off-body regions

Ref: AIAA 2012-713
Why have the structured solver option?

- A near-future solution to an efficient, high-order near-body solver
  - Long term solution – high-order, strand-based solver, DG-based unstructured solvers...

- Use structured solver in combination with unstructured solver
  - **Structured** solver for simple geometries/topologies
    - Fast, high-order accurate (5th order)
    - Efficient (storage, domain decomposition)
    - Acceleration methods (line relaxation, multi-grid)
    - Mesh generation easy for simple geometries
  - **Unstructured** solver for complex geometries – hub, fuselage...

Dombroski et al, AHS Forum 2012
Why OVERFLOW?

- **Overset structured grid solver**

- **Validated** for a wide variety of **rotorcraft** problems
  - Rotor, fuselage, hub, flaps, coaxial rotor system
  - Coupling with CSD (computational structural dynamics)
  - Steady and maneuver flights
  - Dynamic stall

- **Key desirable features**
  - High-order schemes
  - Near-body grid adaptation
  - Turbulence and transition modeling

- **Industry users are vested**
  - Effort spent in mesh generation, validation, developing know-hows...

- **Continuously being developed and supported**
Modularized OVERFLOW

- Is a full-featured software...

What is retained in the modularized version?
- Mesh Motion (GMP/XML) & Mesh Deformation
- Near-body Connectivity (viscous stencil repair)
- FOMOCO
- Parallel grid partitioning

What is not?
- Off-body region
- FSI
- File-based CFD/CSD Coupling
Data Interface/Exchange

Module Interfaces

DCF
Domain Connectivity

FSI
Fluid Structure Interface

Object Oriented Python Integration Framework

P0
P1
P2
PN

Distributed Memory processors communicating via MPI

shared data
OVERFLOW DATA

Mesh Motion

Near-Body
CFD Solver

Near-Body
CFD Solver

Off-Body
Solver

Computational
Structural Dynamics

Co-Visualization

OVERFLOW DATA

Grid coordinates
Grid connectivity
Wall nodes
Outer nodes

Surface Forces
(FOMOCO)

Surface Deflections

Solution
I-BLANK

OVERFLOW
(NASA Ames)

NSU3D
(U. Wyoming)

SAMARC
(LLNL/NASA Ames)

RCAS
(AFDD, US Army)
CAMRADII
(Johnson Aeronautics)

ParaView
FieldView
Near-body Connectivity Strategy

- Group grids into Components/Bodies (blade, hub, fuselage...)
- Use grid names in over.namelist

**OVERFLOW COMMUNICATION**
- Intra-component grid connectivity & communication (viscous stencil repair)

**HELIOS COMMUNICATION**
- Inter-component grid connectivity & communication
- Components (near-body) to off-body communication

Other components or off-body grids
Near-body Connectivity Strategy

Component # 1
GRID-1
GRID-2

Component # 2
GRID-3
GRID-4
Near-body Connectivity Strategy

OVERFLOW COMMUNICATION
- Intra-component grid connectivity & communication

HELIOS COMMUNICATION
- Inter-component grid connectivity & communication
- Components (near-body) to off-body communication
Traditional approach:
- integrate along blade grid line
- integration force/moment about quarter chord
- integration error, inconsistencies between CFD & CSD

New approach inherited from the HELIOS FSI module:
- integrate face-by-face on stitched FOMOCO surface
- convert to 1-D beam forcing based on principle of virtual work
- accurate, conservative
- applicable to large surface deformation and flapped rotor cases
Co-Visualization

- On-the-fly, parallel co-visualization
- Typically handy for large dataset simulation on remote clusters

- Slices, iso-surface, streamlines, point/line/surf probes
- User defined types
- Need access to FORTRAN-90 derived data types in OVERFLOW
• OVERFLOW
  – A single, common source code repository
    o *make* – compiles standalone executable
    o *make library* – compiles the *python* version
    o preprocessor directives (#ifdef PYTHON)
    o all python-related code contained in a separate python subdirectory

• HELIOS
  – *Common python interface* for NSU3D and OVERFLOW
Examples

- Flow over a cylinder
- Isolated rotor
- Multiple rotor
- Rotor-hub-fuselage
- Multiple rotor + fuselage
Flow Over a Cylinder

DCF
Domain Connectivity

Object Oriented Python Integration Framework

P0  P1  P2  PN

Distributed Memory processors communicating via MPI

shared data

Near-Body CFD Solver
Off-Body Solver

OVERFLOW  SAMARC
WALL-WALL/Nearbody-Nearbody overlap region overset communication and parallel partitioning handled by OVERFLOW

Outer boundary overset communication handled by PUNDIT
Rotor with Prescribed Blade Deformations

Object Oriented Python Integration Framework

DCF
Domain Connectivity

FSI
Fluid Structure Interface

PUNDIT

Rotor-FSI

DCF
Near-Body CFD Solver

FSI
Off-Body CFD Solver

P0 P1 P2 PN

Distributed Memory processors communicating via MPI

shared data

OVERFLOW

SAMARC
**Wall-wall overlap** region overset communication b/w tip cap and blade grid handled by **OVERFLOW**

**Outer boundary** overset communication b/w near-body OVERFLOW grid and off-body SAMARC grid handled by **PUNDIT**
Two Near-body Solvers

Object Oriented Python Integration Framework

DCF
Domain Connectivity

FSI
Fluid Structure Interface

Distributed Memory processors communicating via MPI

shared data

Mesh Motion

Near-Body CFD Solver-1
Near-Body CFD Solver-2
Near-Body CFD Solver-n

Off-Body Solver

PUNDIT
Rotor-FSI

OverFLOW
(NASA Ames)

NSU3D
(U. Wyoming)

SAMARC
(LLNL/NASA Ames)
Hub + Rotor

- OVERFLOW: Hexahedrals
- NSU3D: Tetrahedrals
- SAMARC: Cartesian
Coax Rotor, Mixed Grids

OVERFLOW

NSU3D

SAMARC
Co-Visualization

Upper rotor: **OVERFLOW** (structured)

Lower rotor: **NSU3D** (unstructured)

Off-body: **SAMARC** (Cartesian)

Iso-surfaces of Q criterion computed on-the-fly using the ParaView plug-in
Hub + Rotor + Fuselage

Hub, unstructured

Fuselage, unstructured

Blades, structured

Notional hub geometry courtesy of Sikorsky
Fuselage grid courtesy of NASA Langley
Hub + Rotor + Fuselage

Notional hub geometry courtesy of Sikorsky
Fuselage geometry courtesy of NASA Langley
• HELIOS’s modular, python-based framework is flexible and extensible for incorporating new modules

• The OVERFLOW code has been modularized into a HELIOS component as a near-body solver

• HELIOS framework supports multi-solver capability (NSU3D, OVERFLOW, SAMARC)

• **Lends great flexibility to users**

<table>
<thead>
<tr>
<th>HELIOS</th>
<th>OVERFLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluid-structure Interface</td>
<td>High-order near-body</td>
</tr>
<tr>
<td>Co-visualization</td>
<td>Structured near-body</td>
</tr>
<tr>
<td>Unstructured near-body</td>
<td>Near-body AMR</td>
</tr>
<tr>
<td>Implicit hole cutting</td>
<td>Turbulence and Transition Modeling</td>
</tr>
<tr>
<td>Off-body Adaptive Mesh Refinement (AMR)</td>
<td>Efficiency</td>
</tr>
<tr>
<td>Modular – future add ons</td>
<td>Future add ons</td>
</tr>
<tr>
<td>GUI</td>
<td></td>
</tr>
</tbody>
</table>

• **Ongoing work** – Alpha testing, benchmarking, and validation
End