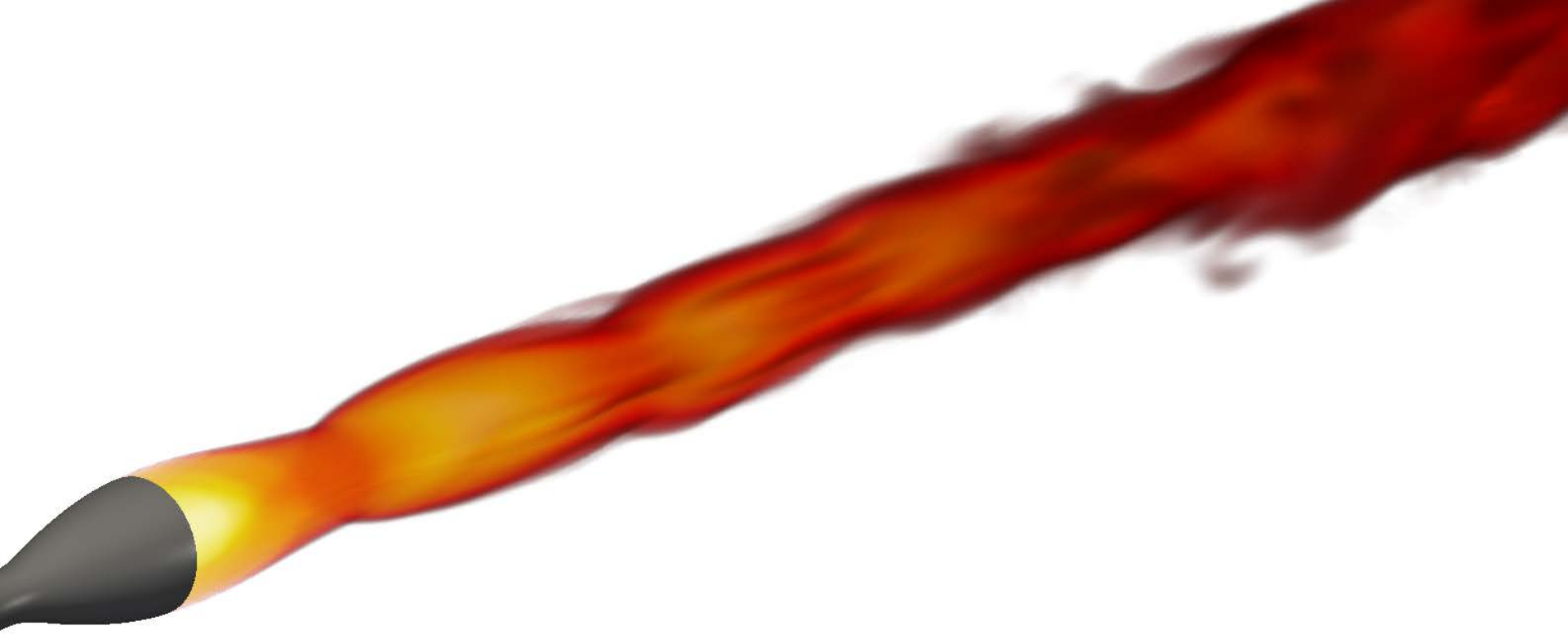


A General Purpose, Open-Source Overset Library for Caelus based on Suggar++

Dr. Chris Sideroff

Dr. Darrin Stephens

Overset with OpenFOAM

- Previous library/solver efforts:
  - foamedOver (Boger et al, PSU-ARL)
  - naoeFOAM-os-SJTU (Shen at al, Shanghai Jiao Tong University)
  - Others?
- Previous OGA efforts:
  - OperaFOAM (Chandar et al IHPC Singapore)
  - Bellerophon (Groß, Univ. of Rostock)
  - Others?
- None have been publicly released or available for purchase (to our knowledge)

Library

- Caelus (OpenFOAM derivative) is collection of libraries
- Goals:
  - Hide overset-specific details
  - Minimal changes to existing library and top-level applications
  - Use existing linear solvers
  - Minimal changes to existing workflow
  - Use Suggar++ for OGA
- Features
  - Static, prescribed and dynamic motions
  - 6-DoF ODE solver w/ constraints, restraints and loads
  - No restriction on parallel decomposition
  - static/dynamic, single-phase/VOF solvers

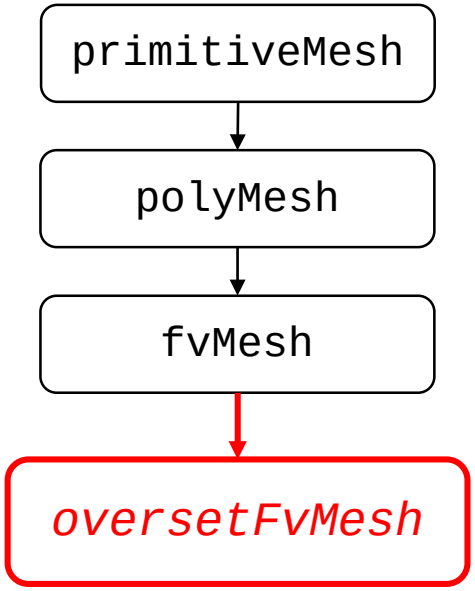
Implementation

- oversetFvMesh: derived from fvMesh class
- Important because (almost) all existing top-level functionality uses fvMesh
  - Inherits all non-overset functionality
  - Handle overset specific duties with virtual and overloaded functions

```
class oversetFvMesh : public fvMesh
{
…
}
```

Low Level Changes

- Few changes required to main Caelus library to support oversetFvMesh library. E.g.
  - Dummy equation and matrix manipulation functions for non-overset
  - Small matrix support for 6 DoF ODE functionality
  - Find functions for mesh primitives required virtualization



Implementation

- Manipulate matrix and field values of fringe cells
- Templated matrix (momentum, pressure) and field (volume fraction) update functions
- Calls DiRTlib and libSuggar++ to update fringe

```
fvVectorMatrix Ueqn
(
    fvm::ddt(U) +
    fvm::div(phi, U) –
    fvm::laplacian(nu, U)
    ==
    - fvc::grad(p)
);
```

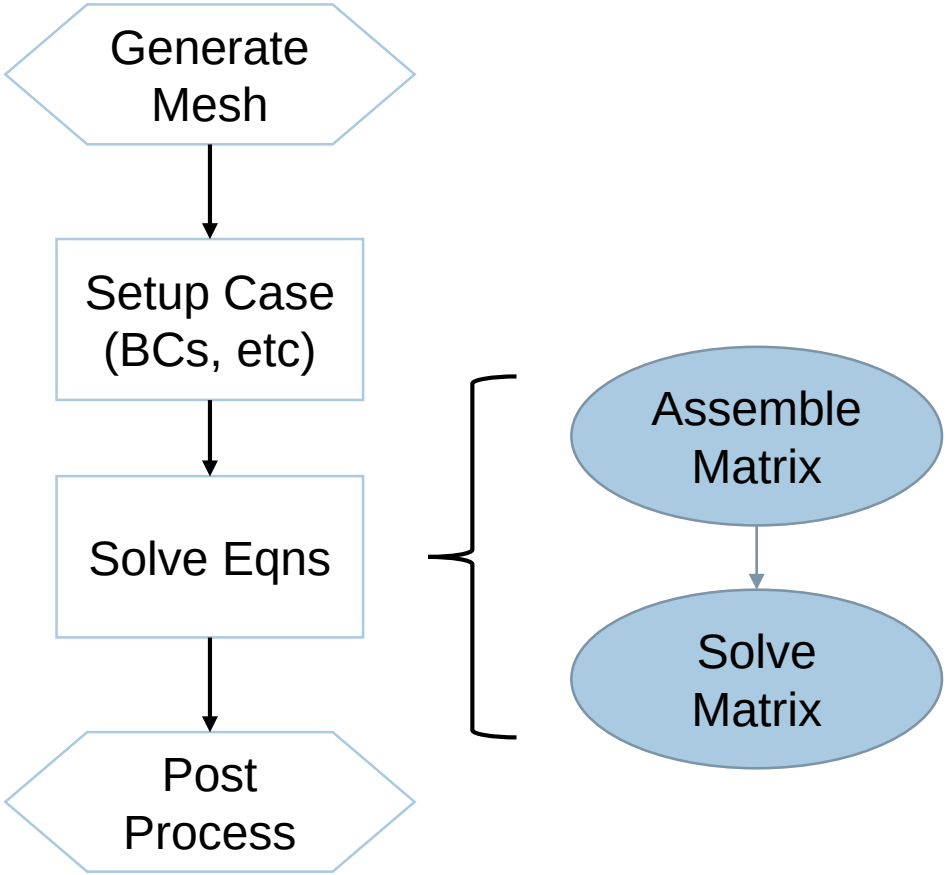
Implementation

- Manipulate matrix and field values of fringe cells
- Templated matrix (momentum, pressure) and field (volume fraction) update functions
- Calls DiRTlib and libSuggar++ to update fringe

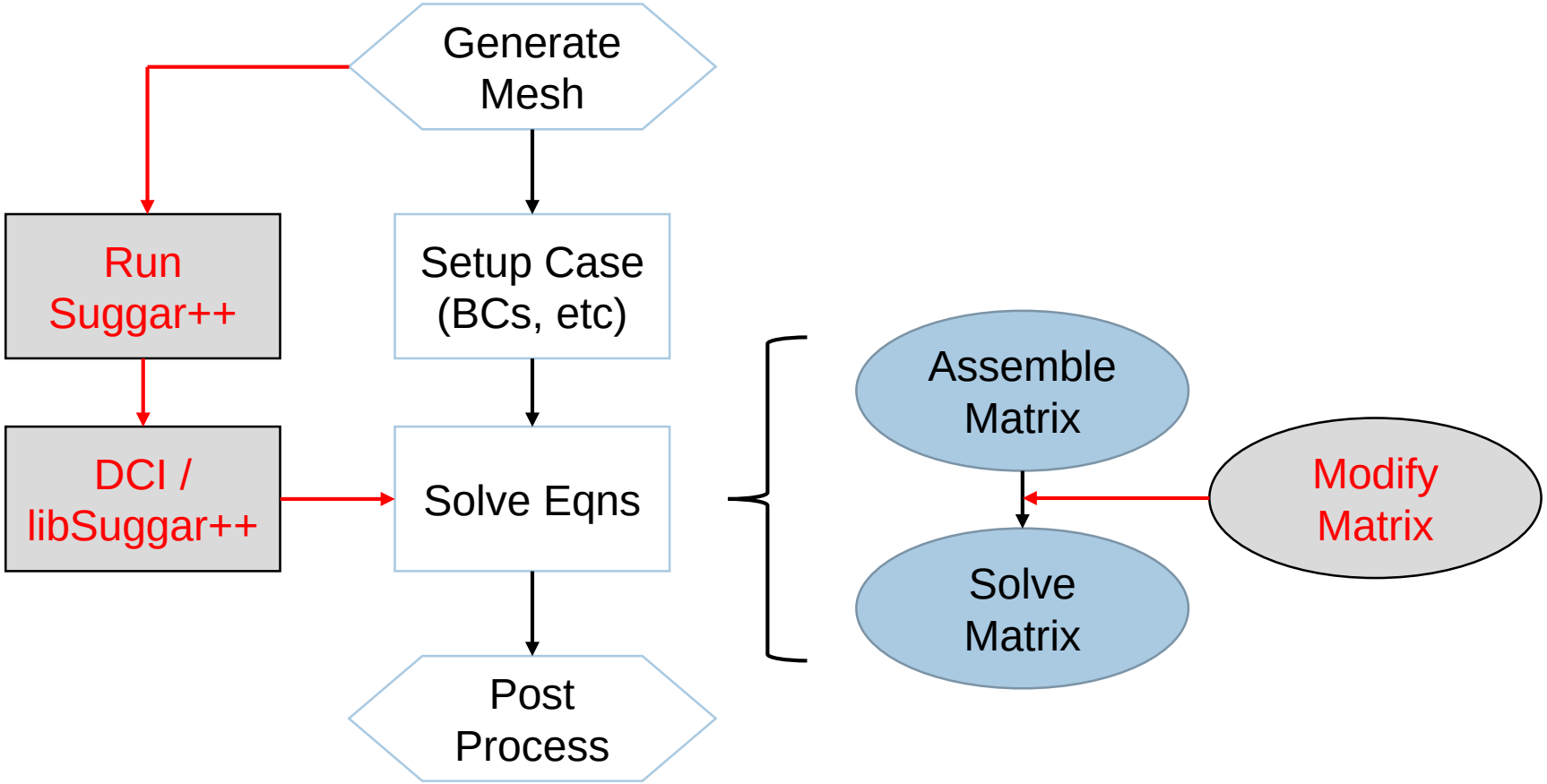
```
// Update matrix implicitly
mesh.updateFvMatrix<vector>(UEqn);
UEqn.solve();
```

```
// Update matrix implicitly
mesh.updateFvMatrix<vector>(UEqn);

// Update field value explicitly
mesh.updateFvField<scalar>(alpha1);
```

Usual Workflow



Overset Workflow



Case Structure

```
├── 0
│   ├── U
│   ├── p
│   ├── k
│   ├── omega
│   └── nut
├── 1
│   ├── U
│   ├── ...
│   ├── polyMesh
│   └── uniform
│   │   └── sixDoFRigidBodyState
├── constant
│   └── polyMesh
├── suggar
│   ├── DCI
│   ├── Grids
│   │   ├── background
│   │   └── sphere
│   ├── Input
│   └── Work
└── system
    ├── controlDict
    ├── fvSchemes
    └── fvSolution
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "0/uniform";
    object      sixDoFRigidBodyState;
}

centreOfRotation ( 4.099396 0 1.650153 );
principalOrientation ( 1 0 0 0 1 0 0 0 1 );
velocity        ( 1.0e+01 0.0 0.0 );
acceleration    ( 0.0 0.0 0.0 );
angularMomentum ( 0.0 0.0 0.0 );
torque          ( 0.0 0.0 0.0 );
```

Features

- Suggar++ for OGA:
  - DCI files (DiRTlib)
  - DCI files or in memory (libSuggar++)
- Static (DiRTlib)
- Prescribed
  - Single body, rotation or translation (DiRTlib)
  - Multi body, more complex prescribed motions (libSuggar++)
- Dynamic
  - Multi body
  - Hierarchical body arrangement (mimics Suggar++ input)
  - 6 degree of freedom solver (w/ constraints, restraints, loads)

Orphan Points

- LSQ nearest neighbor averaging
- “Neighborhood” user selectable: radius and levels
- Remediation fully parallel
- Same approach used for changing iblank (dynamic)

```
oversetFvMeshCoeffs
{
    DiRTlibNumberOfVariables 3;
    dciFilePrefix            "output";
    searchRadiusFactor       3.4;
    maxCellCellRows          4;
    nOrphanIterations        50;
    tolOrphanIterations      0.05;
...
}
```

libSuggar++

- More complex prescribed and dynamic motions need libSuggar++

```
oversetFvMeshCoeffs
{
...
    writeComponentCellSets   false;
    writeDynBodiesCellSets   false;
    useLibSuggar++           true;

    // Only libSuggar++
    DCIMemoryTransfer        true;
    nSuggarRanks             1;
    motionMemoryTransfer     true;
    motionFileOutput         true;
    motionOutputDCIFile      false;
}
```

6 Degrees of Freedom

- Available integrators
  - Symplectic
  - Newmark
  - Crank-Nicholson
- Constraints: point, axis, line, plane, orientation
- Restraints: linear spring, linear and angular damper
- Loads: external force and moment
- Selectable gravity, buoyancy and hydrodynamic forces

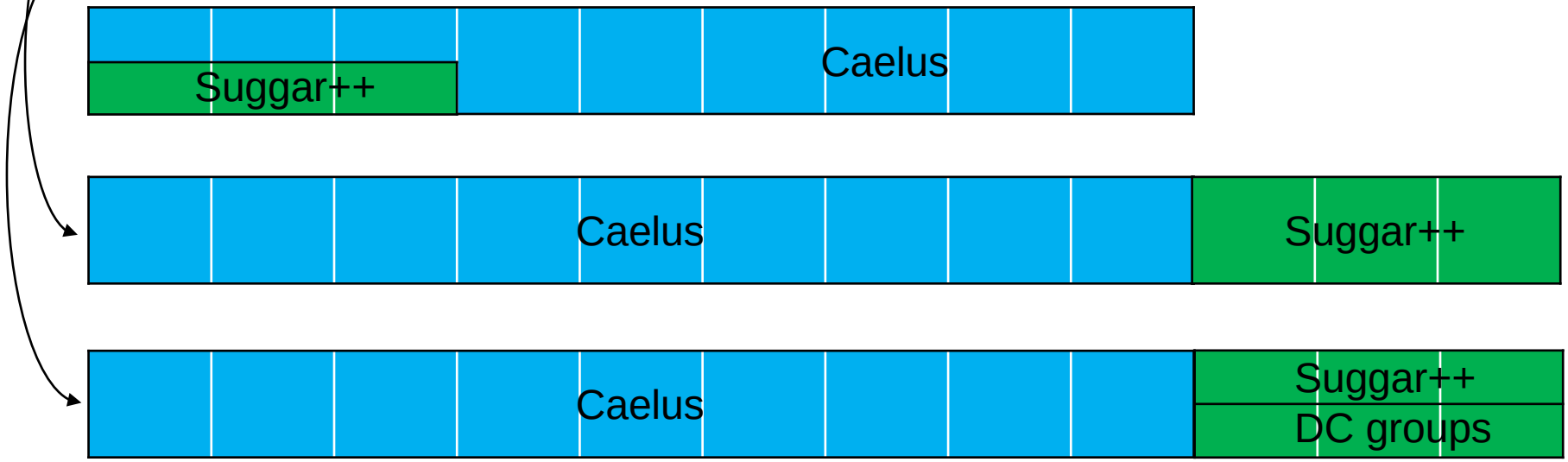
Body Hierarchy

- No limit on number of bodies or levels
- Only limit: no 6DoF-in-6DoF

```
multiOversetBodyMotionFvMeshCoeffs
{
    helicopter
    {
        oversetBodyMotionFunction linearMotion;

        linearMotionCoeffs
        {
        }

        main-rotor
        {
            oversetBodyMotionFunction rotatingMotion;
            rotatingMotionCoeffs
            {
            }

            blade1
            {
                oversetBodyMotionFunction oscillatingRotatingMotion;
                oscillatingRotatingMotionCoeffs
                {
                }
            }
        }
    }
}
```

Parallel

- Solver
  - Usual decomposition methods available
  - No restriction on how or where mesh is decomposed
- Overset (libSuggar++) currently runs in parallel on solver ranks
  - No MPI_COMM splitting (yet)
  - No DC groups (yet)



Solvers

- Will initially provide:
  - Single phase: static and dynamic
  - Multi-phase (VOF): static and dynamic
- Adding solvers “should” be trivial

overSimpleSolver.cpp (C:\Caelus\overset-...lvers\incompressible\simpleSolver) - GVIM2

File Edit Tools Syntax Buffers Window Help

```
    Overset steady-state solver for incompressible, turbulent flow

\*---------------------------------------------------------------------------*/

#include "fvCFD.hpp"
#include "singlePhaseTransportModel.hpp"
#include "RASModel.hpp"
#include "simpleControl.hpp"
#include "fvIOoptionList.hpp"
#include "oversetFvMesh.hpp"

int main(int argc, char *argv[])
{
    #include "setRootCase.hpp"
    #include "createTime.hpp"
    #include "createOversetFvMesh.hpp"
    #include "createFields.hpp"
    #include "createFvOptions.hpp"
    #include "initContinuityErrs.hpp"

    simpleControl simple(mesh);

    Info<< "\nStarting time loop\n" << endl;

    while (simple.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        // --- Pressure-velocity SIMPLE corrector
        {
            #include "UEqn.hpp"
            #include "pEqn.hpp"
        }

        turbulence->correct();

        runTime.write();

        #include "reportTimeStats.hpp"
    }

    Info<< "End\n" << endl;

    return 0;
}
```

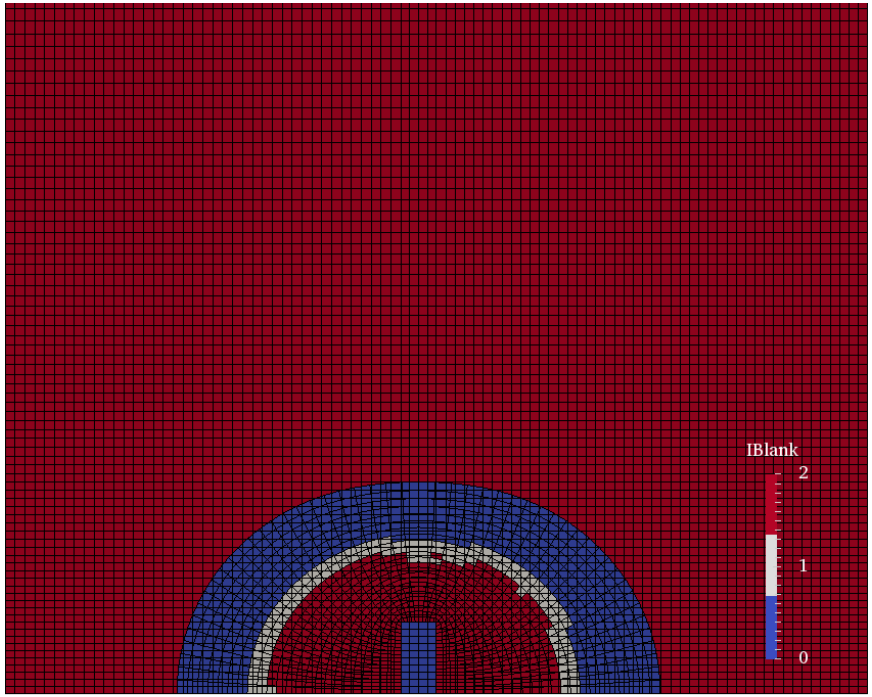
UEqn.hpp + (C:\Caelus\overset-clean\appl...lvers\incompressible\simpleSolver) - GVIM2

File Edit Tools Syntax Buffers Window Help

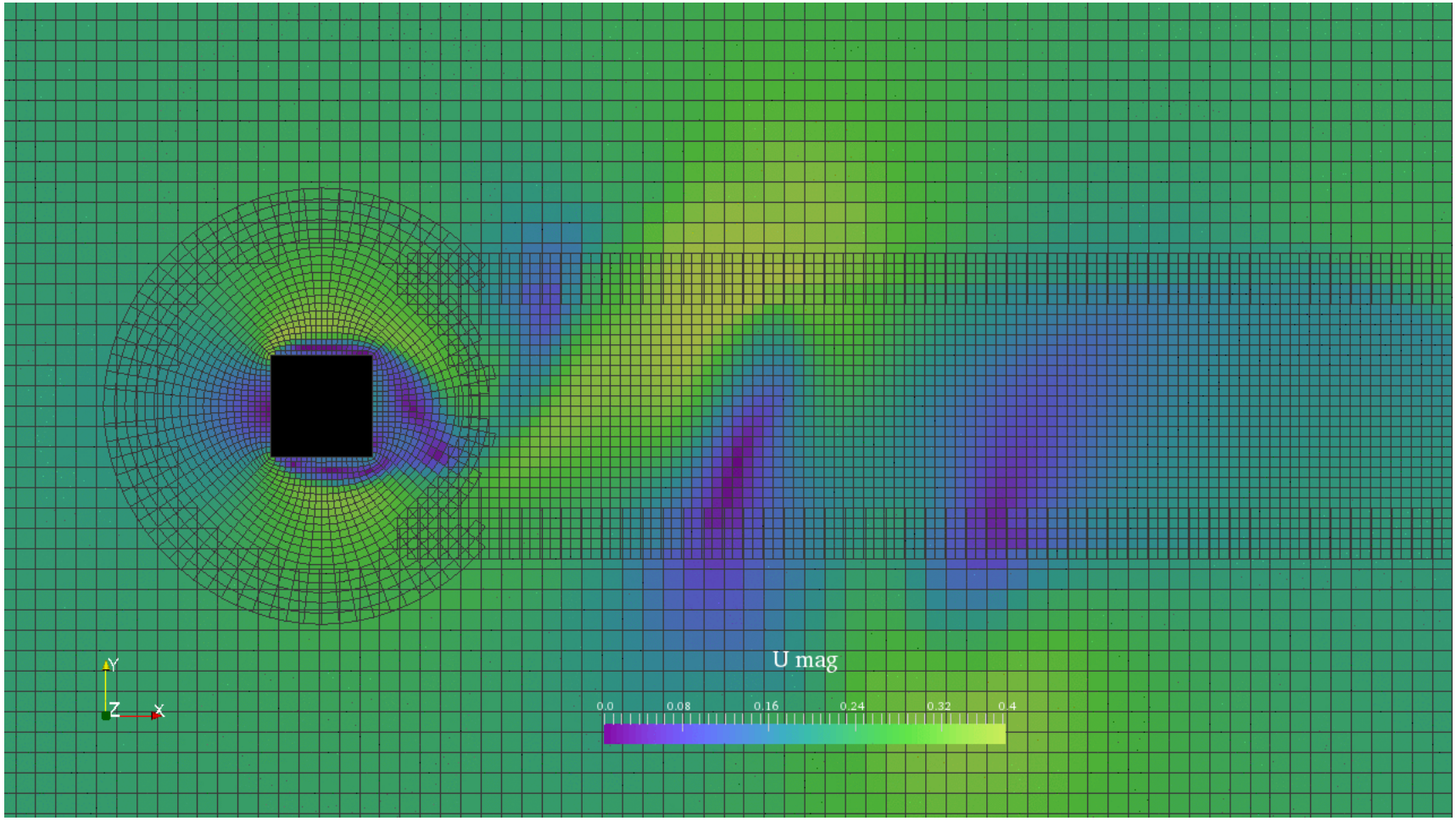
```
    // Momentum predictor

    tmp<fvVectorMatrix> UEqn
    (
        fvm::div(phi, U)
      + turbulence->divDevReff(U)
      ==
        fvOptions(U)
    );

    UEqn().relax();

    fvOptions.constrain(UEqn());

    mesh.updateFvMatrix(UEqn());

    solve(UEqn() == -fvc::grad(p));

    fvOptions.correct(U);

"UEqn.hpp" 19L, 330C
```

Utilities

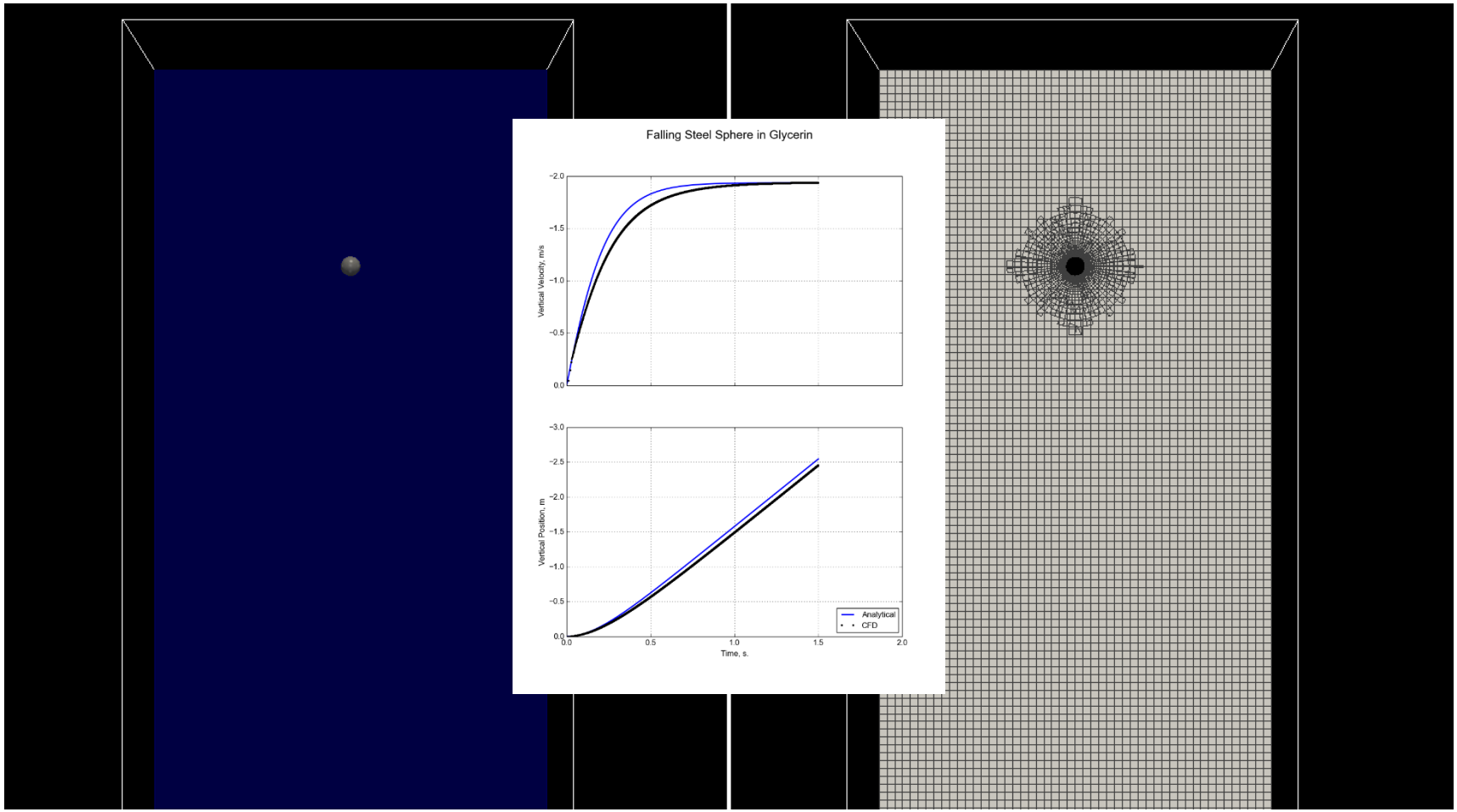
- Move component grids with:
  - Prescribed motion (DCI files)
  - motion.xml
- Create overset data: iblank, component ids
- Compute mass and moment of inertial properties of bodies



Static Validation

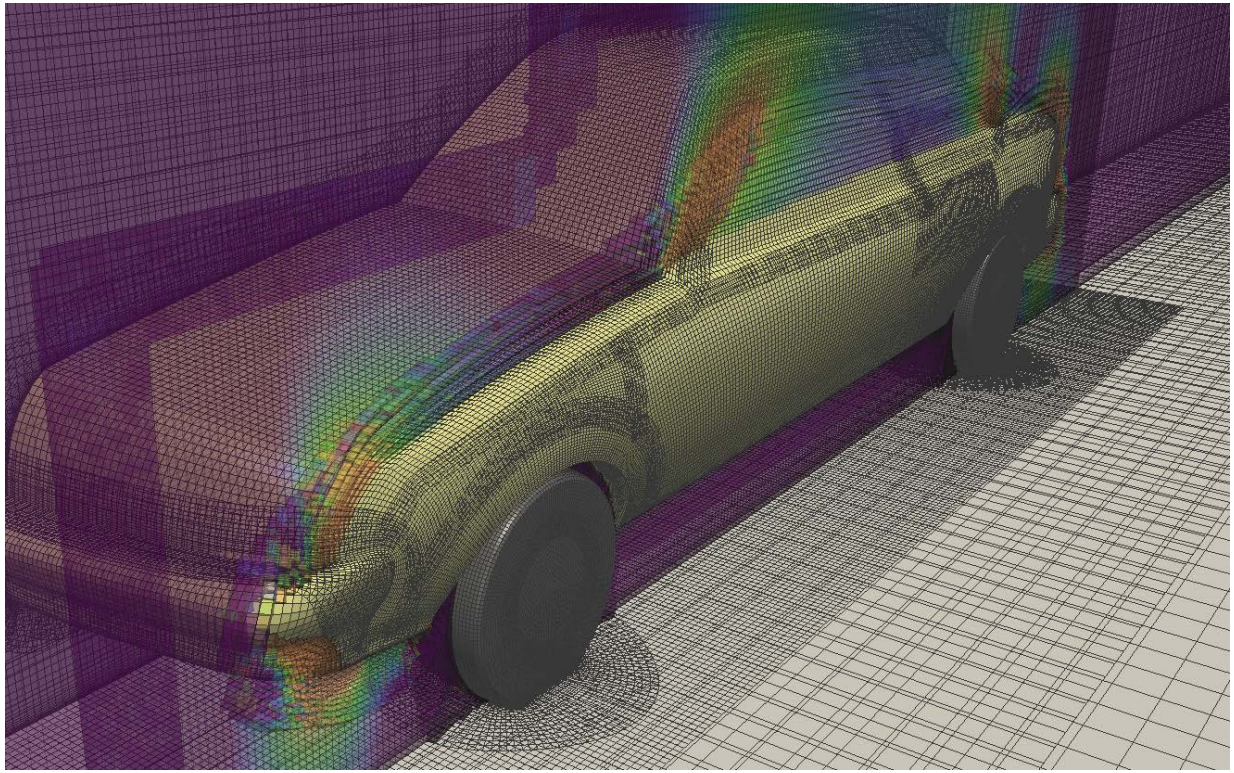


6 DoF Validation



Availability

- Yes
- Open-source but not free (as in beer)



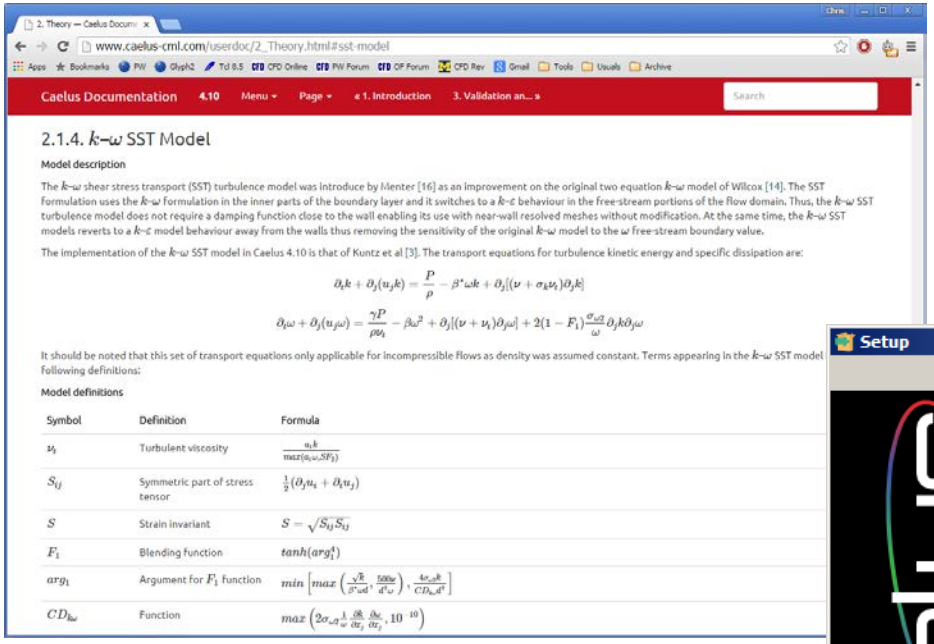
What is caelus ?

Caelus is a derivative of OpenFOAM

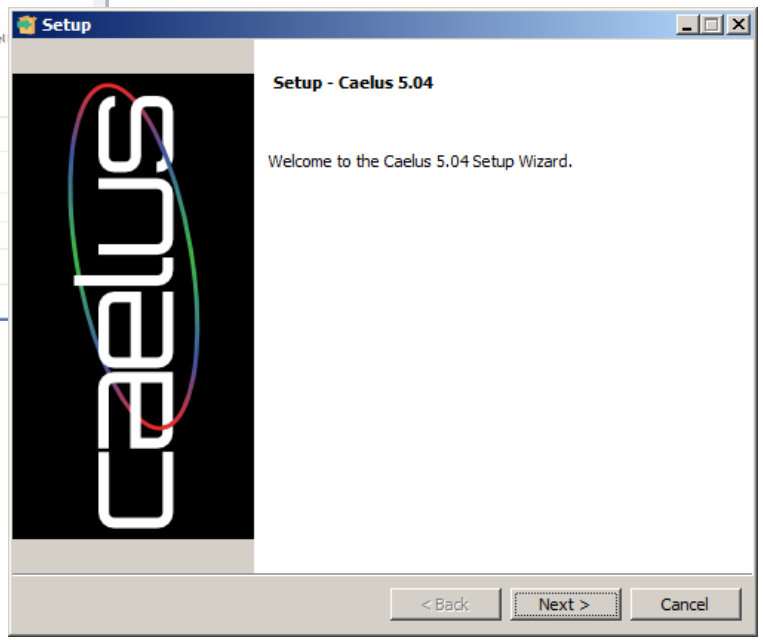
- Open and free (as in beer): www.caelus-cml.com
- Supports multiple platforms (Windows, Linux and Mac)
- Robust compilation: Scons
- Easy to install
- ***Verified*** schemes and ***Validated*** solvers and turbulence models
- Documentation and validation cases
- Improved algorithmic robustness on non-”perfect” meshes
- Improved accuracy on non-”perfect” meshes
- Swak, Python wrapping, etc
- New compressible solvers and turbulence models
- New VLES turbulence model class
- Stable, predictable API

Applied CCM

What is Caelus?



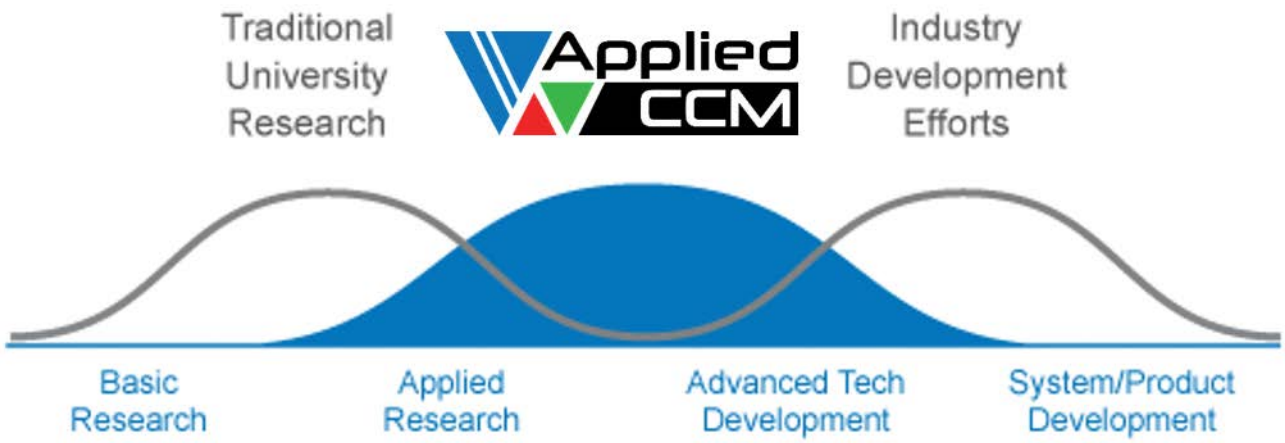
Documentation and validation cases



Easy installation with minimal dependencies (Python and MPI)

What we do

- Specialise in the application, support and development of open source computational mechanics
- Provides customer value by conducting industry-relevant TRL 2-6 R&D
- Engage with customers as their technology partner



Industry Relevant Research

Technology Summary

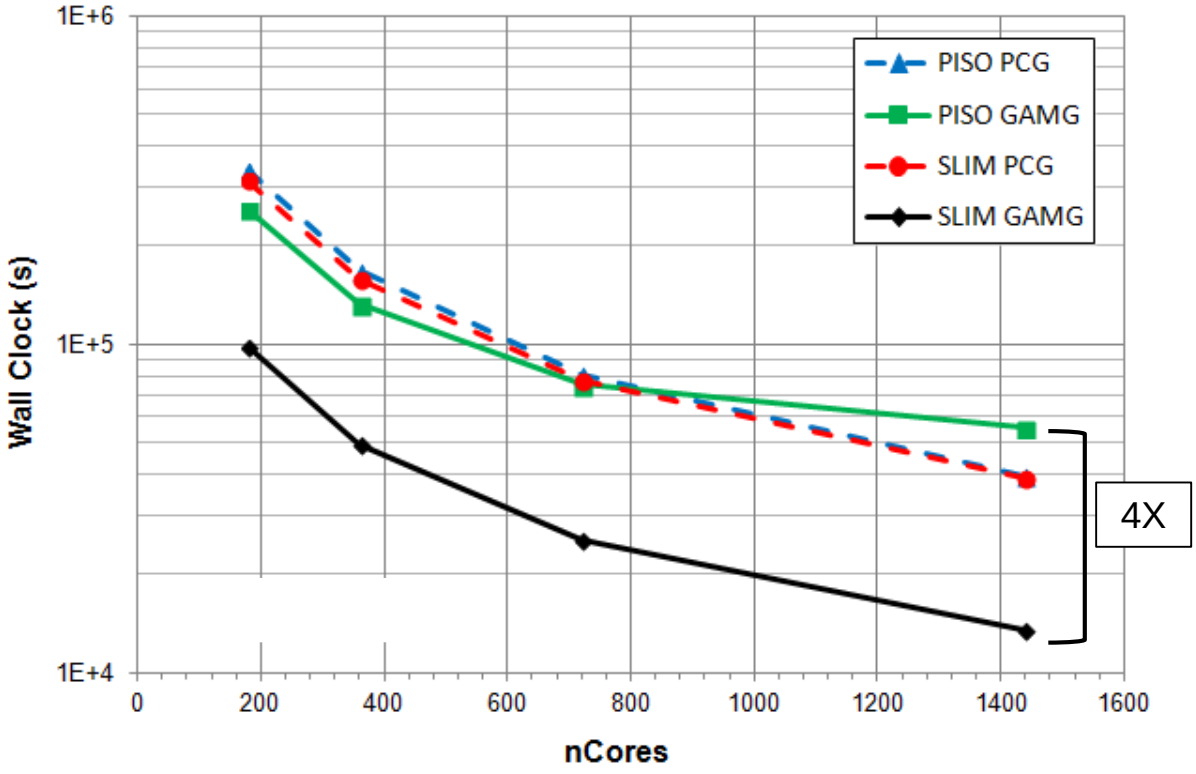
- Solvers
  - Highly efficient incompressible/compressible fractional-step solvers *(AIAA Scitech 2015)*
- Combustion
  - Arrhenius finite-rate chemistry models *(GTP 2015)*
  - Thermo-acoustic instability solvers *(JPP 2015)*
- Sensitivity, Optimization and Uncertainty (SOU)
  - Adjoint *(SAE 2014)*
  - Tangent
  - Uncertainty and error propagation
- Turbulence modelling
  - VLES *(APISAT 2015)*
- Acoustics solvers *(JPP 2015)*
  - Linearized Euler
  - Linearized Perturbed Compressible Equations (LPCE)
- *Overset*

Current Research Projects

- Multi-scale multiphase solver
- Employ adjoint, tangent, polynomial chaos to:
  - Optimize for acoustic phenomena
  - Investigate uncertainty and error propagation of turbulence parameters and quantities (e.g. empirical coefficients)
- Automated vertically integrated CFD workflows:
  - Meshing → Solving → Post-pro → SOU → Meshing …
- Development of generic overset framework (solver agnostic) to enable:
  - Higher quality meshes
  - Multiple, nested dynamic body motions

Semi Linear Implicit Method - SLIM

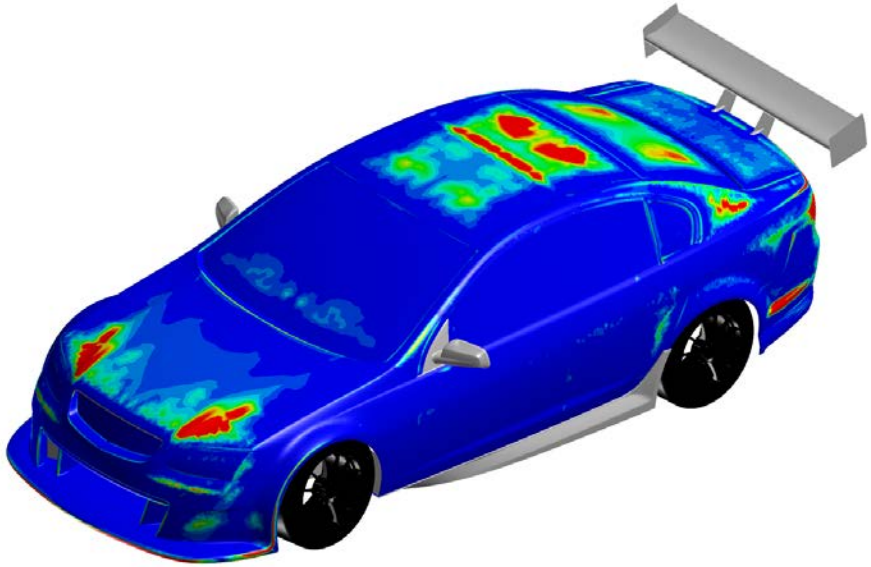
- Fast, low dissipation fractional step solver based on projection method
- Significantly better scalability than traditional implicit transient solvers

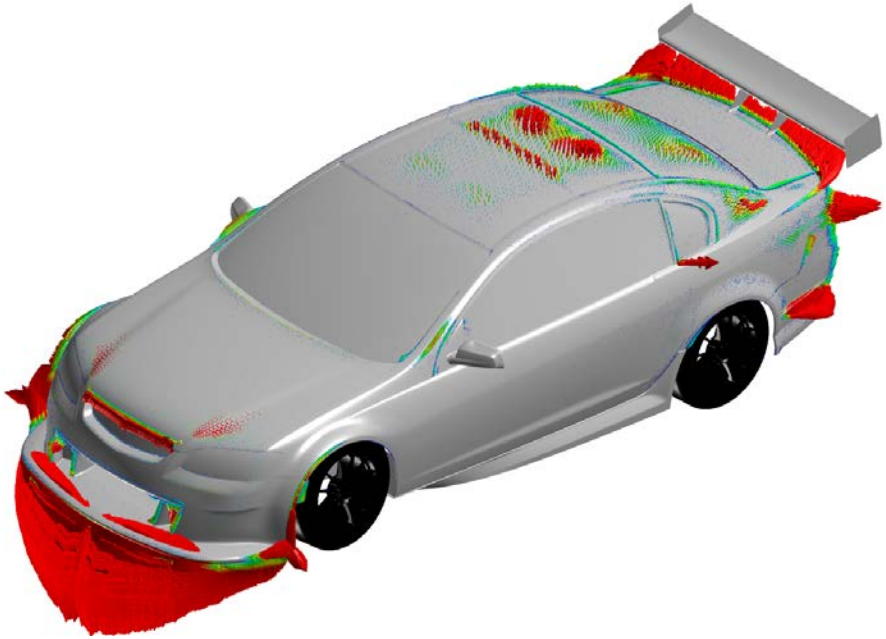


Industrial parallel performance study of 50 million cell ABL simulation

Adjoint

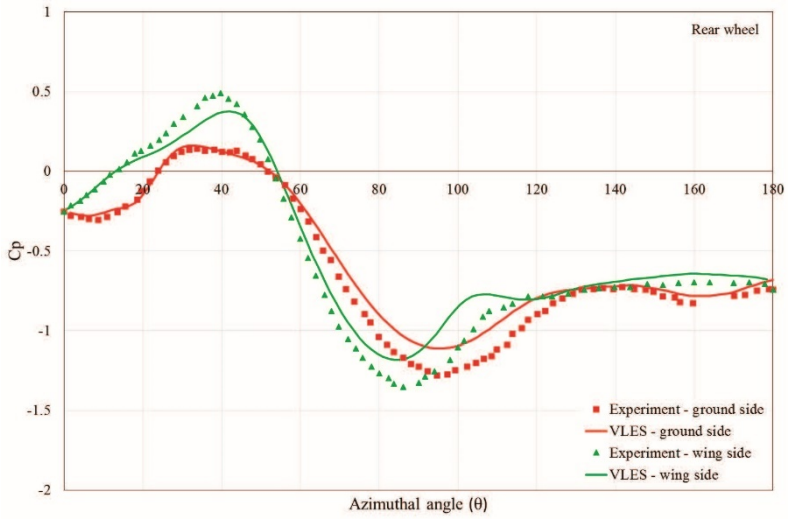
- Time-symmetry preserving semi-discrete surface adjoint:
  - Provides both direction and magnitude
- Tested on meshes up to 60M cells without limitation
- Used to redesign front air dam on V8 supercar



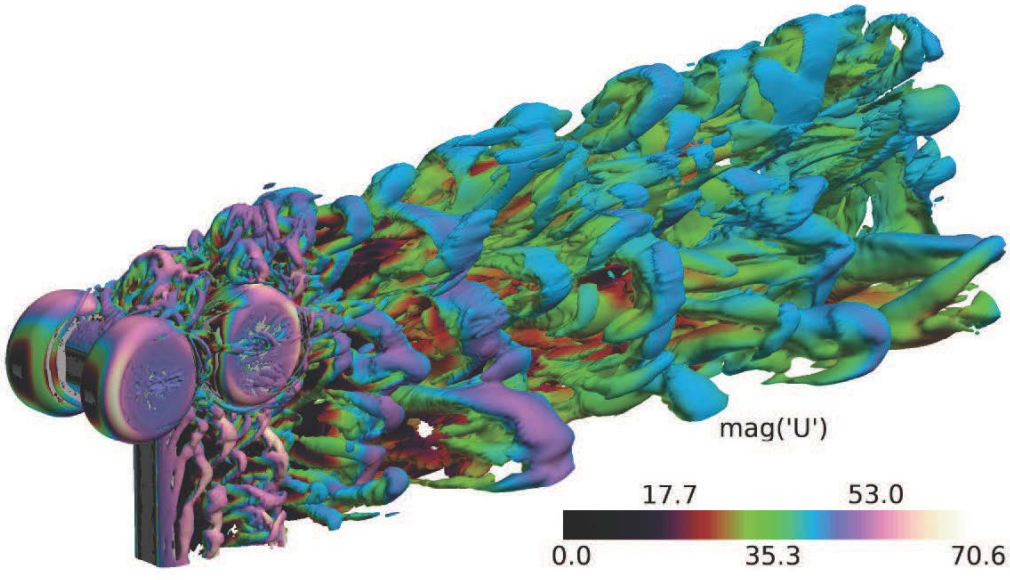


VLES: Hybrid RANS/DES Turbulence Model

- Based off of Speziale, C. G. (1998) “Turbulence modeling for time-dependent RANS and VLES: A review”, *AIAA J.*, **36**, pp. 173–184.
- Use resolution function, F_r, off body to blend RANS and LES



Rear wheel pressure coefficient



Rudimentary Landing Gear , AIAA Aeroacoustics Workshop

Who we are

Darrin Stephens

has an extensive background in modeling a range of complex reacting multiphase flows and led the development of CFD models for a number of industrial unit processes allowing re-design and optimization, improving throughput and efficiencies. He is the managing director and principle research engineer at Applied CCM Pty Ltd., a company providing development, technical support and training for the open source CFD software OpenFOAM. His professional interests include multi-phase algorithms, optimization algorithms, population balance modeling, coupled solvers and the development of next-generation hybrid and multi-physics models. He earned his Ph.D. degrees in mechanical engineering from James Cook University in Townsville, Australia.

Chris Sideroff

is the managing director and principle research engineer at Applied CCM Canada, a company providing development, technical support and training for open source based CFD software such as Caelus and OpenFOAM. His current research interests are automation methods for mesh generation, more robust discretization schemes for highly anisotropic and skewed meshes for finite volume based CFD solvers, work flow automation (pre, solve, post) with interpreted programming languages such as Python, and overset grid technologies. He earned his Ph.D. degree in mechanical and aerospace engineering from Syracuse University in Syracuse, USA.

Aleks Jemcov

has 20 years of experience in computational fluid dynamics and turbomachinery related flows. His research interests include development of numerical algorithms for fluid mechanics, solid-fluid interactions, efficient solver methodologies applicable to both compressible and incompressible flows, thermo-acoustics, turbulence modeling, and linear algebra. Prior experience includes work in the Combustion Dept. at Pratt & Whitney Canada, Development Dept. at Fluent Inc., and the Development Department and Fluids Business Unit at ANSYS, Inc. He currently serves as Associate Director for Computational Sciences at the Notre Dame Turbomachinery Laboratory and is a Research Assistant Professor in the Aerospace and Mechanical Engineering Department at the University of Notre Dame. He earned his Ph.D. in Aerospace Engineering from the University of Belgrade.

Questions

Applied CCM Inc
Dr Chris Sideroff

Phone: +1 613 276 7472
Email: c.sideroff@appliedccm.ca
Web: www.appliedccm.com

Applied CCM Pty Ltd
Dr Darrin Stephens

Phone: +61 03 8376 6962
Email: d.stephens@appliedccm.com.au
Web: www.appliedccm.com

www.caelus-cml.com