Computational Aircraft Prototype Syntheses

Training Session 8
CFD Analysis: Fun3D/SU2

ESP v1.19

Marshall Galbraith
galbramc@mit.edu

Bob Haimes
haimes@mit.edu

Massachusetts Institute of Technology

John F. Dannenhoffer, III
jfdannen@syr.edu
Syracuse University

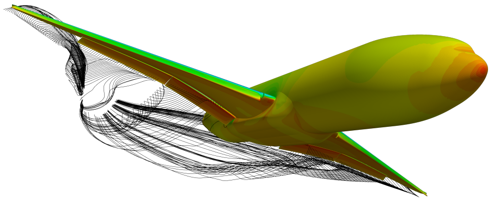
Overview

- Unstructured CFD analysis
  - SU2
  - FUN3D

- CFD analysis setup
  - CFD execution

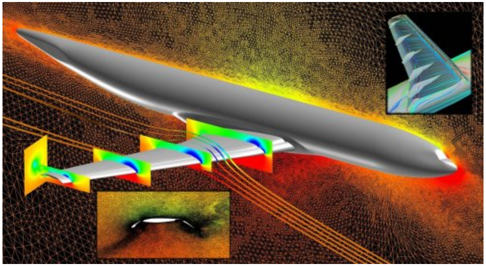
SU2 Overview

- Open-source CFD solver (https://su2code.github.io/)
- Unstructured meshes
- Mesh deformation
- Adjoint based optimization
- CAPS: 4.1.1. (Cardinal), 5.0.0 (Raven), 6.2.0 (Falcon), 7.1.1 (Blackbird)



Fun3D Overview

- Developed at NASA Langley since late 1980s
- Unstructured meshes
- Mesh deformation
- Adjoint based optimization
- Much more
- CAPS: Fun3D 13.4



CFD Simulation Outline

- Load geometry
- Generate mesh (the most difficult CFD input to generate)
- Load CFD AIM
  - Set CFD analysis inputs
- Execute CFD
- Extract analysis outputs

session08/su2_1_InviscidWing.py
session08/fun3d_2_InviscidWing.py

SU2 AIM Analysis Inputs

session08/su2_1_InviscidWing.py

```
# Create SU2 AIM
su2 = myProblem.analysis.create(aim  = "su2AIM",
                                name = "su2")

# Link the aflr3 Volume_Mesh as input to su2
su2.input["Mesh"].link(aflr3.output["Volume_Mesh"])

# Set project name. Files written to analysisDir will have this name
su2.input.Proj_Name = "inviscidWing"

su2.input.Alpha = 1.0                    # AoA
su2.input.Mach = 0.5                     # Mach number
su2.input.Equation_Type = "Compressible" # Equation type
su2.input.Num_Iter = 5                   # Number of iterations

# Set boundary conditions via capsGroup
inviscidBC = {"bcType" : "Inviscid"}
su2.input.Boundary_Condition = {"Wing"    : inviscidBC,
                                "Farfield": "farfield"}

# Specifcy the boundares used to compute forces
su2.input.Surface_Monitor = ["Wing"]

# Set SU2 Version
su2.input.SU2_Version = "Blackbird"
```

SU2 Execution

- Execute using SU2 python interface

session08/su2_1_InviscidWing.py

```python
# Import SU2 python environment
from parallel_computation import parallel_computation as su2Run
```

```python
# Run AIM pre-analysis
su2.preAnalysis()

####### Run SU2 ######################
print ("\n\nRunning SU2......")
currentDirectory = os.getcwd() # Get our current working directory

os.chdir(su2.analysisDir) # Move into test directory

# Run SU2 with specified number of partitions
su2Run(su2.input.Proj_Name + ".cfg", partitions = 1)

os.chdir(currentDirectory) # Move back to top directory
#######################################

# Run AIM post-analysis
su2.postAnalysis()
```

SU2 Extracting Analysis Output

- Retrieve forces and moments

session08/su2_1_InviscidWing.py

```
print ("\n==> Total Forces and Moments")
# Get Lift and Drag coefficients
print ("--> Cl = ", su2.output.CLtot,
           "Cd = ", su2.output.CDtot)

# Get Cmx, Cmy, and Cmz coefficients
print ("--> Cmx = ", su2.output.CMXtot,
           "Cmy = ", su2.output.CMYtot,
           "Cmz = ", su2.output.CMZtot)

# Get Cx, Cy, Cz coefficients
print ("--> Cx = ", su2.output.CXtot,
           "Cy = ", su2.output.CYtot,
           "Cz = ", su2.output.CZtot)
```

Fun3D AIM Analysis Inputs

session08/fun3d_2_InviscidWing.py

```
# Create Fun3D AIM
fun3d = myProblem.analysis.create(aim  = "fun3dAIM",
                                  name = "fun3d")

# Link the aflr3 Volume_Mesh as input to fun3d
fun3d.input["Mesh"].link(aflr3.output["Volume_Mesh"])

# Set project name. Files written to analysisDir will have this name
projectName = "inviscidWing"
fun3d.input.Proj_Name = projectName

fun3d.input.Alpha = 1.0                     # AoA
fun3d.input.Mach = 0.5                      # Mach number
fun3d.input.Equation_Type = "Compressible" # Equation type
fun3d.input.Num_Iter = 5                    # Number of iterations
fun3d.input.Restart_Read = 'off'            # Do not read restart
fun3d.input.Viscous = "inviscid"            # Inviscid calculation

# Set boundary conditions via capsGroup
inviscidBC = {"bcType" : "Inviscid"}
fun3d.input.Boundary_Condition = {"Wing"    : inviscidBC,
                                  "Farfield":"farfield"}
```

Fun3D NML Analysis Inputs

- fun3d.nml is very large (and changes with Fun3D versions)
- Not all inputs implemented in AIM
- f90nml used to write directly to fun3d.nml
  - NOTE: Circumvents CLEAN/DIRTY process
  - Always use AIM inputs when available

session08/fun3d_2_InviscidWing.py

```
# Use python to add inputs to fun3d.nml file
fun3d.input.Use_Python_NML = True

# Write boundary output variables to the fun3d.nml file directly
fun3dnml = f90nml.Namelist()
fun3dnml['boundary_output_variables'] = f90nml.Namelist()
fun3dnml['boundary_output_variables']['mach'] = True
fun3dnml['boundary_output_variables']['cp'] = True
fun3dnml['boundary_output_variables']['average_velocity'] = True

fun3dnml.write(os.path.join(fun3d.analysisDir,"fun3d.nml"), force=True)
```

Fun3D Execution

- Execute Fun3D using system call

session08/fun3d_2_InviscidWing.py

```
# Run AIM pre-analysis
fun3d.preAnalysis()

####### Run FUN3D #####################
print ("\n==> Running FUN3D......")
currentDirectory = os.getcwd() # Get our current working directory

os.chdir(fun3d.analysisDir) # Move into test directory

# Run fun3d via system call
os.system("nodet_mpi --animation_freq -1 --write_aero_loads_to_file > Info.out")

os.chdir(currentDirectory) # Move back to top directory
#######################################

# Run AIM post-analysis
fun3d.postAnalysis()
```

Fun3D Extracting Analysis Output

- Retrieve forces and moments

session08/fun3d_2_InviscidWing.py

```
# Get force results
print ("\n==> Total Forces and Moments")
# Get Lift and Drag coefficients
print ("--> Cl = ", fun3d.output.CLtot,
           "Cd = ", fun3d.output.CDtot)

# Get Cmx, Cmy, and Cmz coefficients
print ("--> Cmx = ", fun3d.output.CMXtot,
           "Cmy = ", fun3d.output.CMYtot,
           "Cmz = ", fun3d.output.CMZtot)

# Get Cx, Cy, Cz coefficients
print ("--> Cx = ", fun3d.output.CXtot,
           "Cy = ", fun3d.output.CYtot,
           "Cz = ", fun3d.output.CZtot)
```

Future CFD Simulation Outline

Current Process

- Load geometry
- Generate mesh
- Load CFD AIM
- Execute CFD
- Extract analysis outputs

Future Process

- Load geometry
- Load CFD AIM
- for_each DESPMTR
    - Generate mesh
    - for_each Anlysis_Input
        - Generate CFD input files
- Execute all CFD jobs
- Extract analysis outputs for all jobs