SU2 Analysis Interface Module (AIM)

Nitin Bhagat
UDRI

Ryan Durscher
AFRL/RQVC

0.1 Introduction

0.1.1 SU2 AIM Overview

This module can be used to interface with the open-source CFD code, SU2 [1] [2] with geometry in the CAPS system. For SU2 capabilities and related documentation, please refer to `http://su2.stanford.edu/`. SU2 expects a volume mesh file and a corresponding configuration file to perform the analysis.

An outline of the AIM's inputs and outputs are provided in AIM Inputs and AIM Outputs, respectively.

Details on the use of units are outlined in AIM Units.

Details of the AIM's automated data transfer capabilities are outlined in AIM Data Transfer

0.1.1.1 Automatic generation of SU2 Mesh file

The volume mesh file from SU2 AIM is written in native SU2 format ("filename.su2"). The description of the native SU2 mesh can be found SU2 website. For the automatic generation of mesh file, SU2 AIM depends on Mesh AIMs, for example, TetGen or AFLR4/3 AIM.

0.1.1.2 Automatic generation of SU2 Configuration file

The configuration file ("filename.cfg") from SU2 AIM is automatically created by using the flow features and boundary conditions that were set in the driver program as a user input. For the rest of the configuration variables, default set of values are provided for a general execution. If desired, a user has freedom to manually (a) change these variables based on personal preference, or (b) override the configuration file with unique configuration variables.

0.1.2 SU2 Examples

Here is an example that illustrated use of SU2 AIM SU2 AIM Example. Note this AIM uses TetGen AIM for volume mesh generation.

0.2 AIM Units

A unit system may be optionally specified during AIM instance initiation. If a unit system is provided, all AIM input values which have associated units must be specified as well. If no unit system is used, AIM inputs, which otherwise would require units, will be assumed unit consistent. A unit system may be specified via a JSON string dictionary for example: unitSys = "{"mass": "kg", "length": "m", "time":"seconds", "temperature": "K"}"

0.2.1 JSON String Dictionary

The key arguments of the dictionary are described in the following:

- **mass = "None"**
  Mass units - e.g. "kilogram", "k", "slug", ...

- **length = "None"**
  Length units - e.g. "meter", "m", "inch", "in", "mile", ...

- **time = "None"**
  Time units - e.g. "second", "s", "minute", ...

- **temperature = "None"**
  Temperature units - e.g. "Kelvin", "K", "degC", ...

0.3 AIM Inputs

For the description of the configuration variables, associated values, and available options refer to the template configuration file that is distributed with SU2. Note: The configuration file is dependent on the version of SU2 used. This configuration file that will be auto generated is compatible with SU2 4.1.1. (Cardinal), 5.0.0 (Raven), 6.2.0 (Falcon) or 7.4.0 (Blackbird - Default)

- **Proj_Name = "su2_CAPS"**
  This corresponds to the project "root" name.

- **Mach = NULL**
  Mach number; this corresponds to the MACH_NUMBER keyword in the configuration file.

- **Re = NULL**
  Reynolds number; this corresponds to the REYNOLDS_NUMBER keyword in the configuration file.

- **Math_Problem = "Direct"**
  Math problem type; this corresponds to the MATH_PROBLEM keyword in the configuration file. Options: DIRECT, CONTINUOUS_ADJOINT, DISCRETE_ADJOINT, ... see SU2 template for additional options.

- **Physical_Problem = "Euler"**
  Physical problem type; this corresponds to the PHYSICAL_PROBLEM keyword in the configuration file. Options: Euler, Navier_Stokes, Wave_Equation, ... see SU2 template for additional options.

- **Equation_Type = "Compressible"**
  Equation regime type; this corresponds to the REGIME_TYPE keyword in the configuration file. Options: Compressible or Incompressible.

- **Turbulence_Model = "SA_NEG"**
  RANS turbulence model; this corresponds to the KIND_TURB_MODEL keyword in the configuration file. Options: NONE, SA, SA_NEG, SST, SA_E, SA_COMP, SA_E_COMP, SST_SUST

- **Alpha = 0.0**
  Angle of attack [degree]; this corresponds to the AoA keyword in the configuration file.

- **Beta = 0.0**
  Side slip angle [degree]; this corresponds to the SIDESLIP_ANGLE keyword in the configuration file.

- **Init_Option = "Reynolds"**
  Init option to choose between Reynolds (default) or thermodynamics quantities for initializing the solution (REYNOLDS, TD_CONDITIONS); this corresponds to the INIT_OPTION keyword in the configuration file.

- **Overwrite_CFG = True**
  Provides permission to overwrite configuration file. If set to False a new configuration file won't be generated.

- **Num_Iter = 9999**
  Number of total iterations; this corresponds to the EXT_ITER keyword in the configuration file.

- **CFL_Number = 10.0**
  Courant–Friedrichs–Lewy number; this corresponds to the CFL_NUMBER keyword in the configuration file.

- **Boundary_Condition = NULL**
  See CFD Boundary Conditions for additional details.

- **MultiGrid_Level = 2**
  Number of multi-grid levels; this corresponds to the MGLEVEL keyword in the configuration file.

- **Residual_Reduction = 6**
  Residual reduction (order of magnitude with respect to the initial value); this corresponds to the RESIDUAL↩ _REDUCTION keyword in the configuration file.

- **Unit_System = "SI"**
  Measurement unit system; this corresponds to the SYSTEM_MEASUREMENTS keyword in the configuration file. See SU2 template for additional details.

- **Reference_Dimensionalization = NULL**
  Reference dimensionalization; this corresponds to the REF_DIMENSIONALIZATION keyword in the configuration file. See SU2 template for additional details.

- **Freestream_Pressure = NULL**
  Freestream reference pressure; this corresponds to the FREESTREAM_PRESSURE keyword in the configuration file. See SU2 template for additional details.

- **Freestream_Temperature = NULL**
  Freestream reference temperature; this corresponds to the FREESTREAM_TEMPERATURE keyword in the configuration file. See SU2 template for additional details.

- **Freestream_Density = NULL**
  Freestream reference density; this corresponds to the FREESTREAM_DENSITY keyword in the configuration file. See SU2 template for additional details.

- **Freestream_Velocity = NULL**
  Freestream reference velocity; this corresponds to the FREESTREAM_VELOCITY keyword in the configuration file. See SU2 template for additional details.

- **Freestream_Viscosity = NULL**
  Freestream reference viscosity; this corresponds to the FREESTREAM_VISCOSITY keyword in the configuration file. See SU2 template for additional details.

- **Moment_Center = NULL, [0.0, 0.0, 0.0]**
  Array values correspond to the x_moment_center, y_moment_center, and z_moment_center variables; which correspond to the REF_ORIGIN_MOMENT_X, REF_ORIGIN_MOMENT_Y, and REF_ORIGIN_MOMENT_Z variables respectively in the SU2 configuration script. Alternatively, the geometry (body) attributes "capsReferenceX", "capsReferenceY", and "capsReferenceZ" may be used to specify the x-, y-, and z- moment centers, respectively (note: values set through the AIM input will supersede the attribution values).

- **Moment_Length = NULL, 1.0**
  Reference length for pitching, rolling, and yawing non-dimensional; which correspond to the REF_LENGTH_MOMENT. Alternatively, the geometry (body) attribute "capsReferenceSpan" may be used to specify the x-, y-, and z- moment lengths, respectively (note: values set through the AIM input will supersede the attribution values).

- **Reference_Area = NULL**
  This sets the reference area for used in force and moment calculations; this corresponds to the REF_AREA keyword in the configuration file. Alternatively, the geometry (body) attribute "capsReferenceArea" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).

- **Pressure_Scale_Factor = 1.0**
  Value to scale Cp or Pressure data when transferring data. Data is scaled based on Pressure = Pressure_Scale_Factor∗Cp + Pressure_Scale_Offset.

- **Pressure_Scale_Offset = 0.0**
  Value to offset Cp or Pressure data when transferring data. Data is scaled based on Pressure = Pressure_Scale_Factor∗Cp + Pressure_Scale_Offset.

- **Output_Format = "Paraview"**
  Output file format; this corresponds to the OUTPUT_FORMAT keyword in the configuration file. See SU2 template for additional details.

- **Two_Dimensional = False**
  Run SU2 in 2D mode.

- **Convective_Flux = "Roe"**
  Numerical method for convective (inviscid) flux construction; this corresponds to the CONV_NUM_METHOD_FLOW keyword in the configuration file. See SU2 template for additional details.

- **SU2_Version = "Blackbird"**
  SU2 version to generate specific configuration file. Options: "Cardinal(4.0)", "Raven(5.0)", "Falcon(6.2)" or "Blackbird(7.4.0)".

- **Surface_Monitor = NULL**
  Array of surface names where the non-dimensional coefficients are evaluated

- **Surface_Deform = NULL**
  Array of surface names that should be deformed. Defaults to all inviscid and viscous surfaces.

- **Input_String = NULL**
  Array of input strings that will be written as is to the end of the SU2 cfg file.

- **Mesh = NULL**
  A Area_Mesh or Volume_Mesh link for 2D and 3D calculations respectively.

0.4 AIM Outputs

After successful completion, SU2 writes results in various files. The data from these files can be directly viewed, visualized, and or used for further postprocessing.

One of the files is ("forces_breakdown.dat") which summarizes convergence including flow properties, numerical parameters, and resulting force and moment values. As an AIM output, this file is parsed for force and moment coefficients, and printed as closing remarks.

Net Forces - Pressure + Viscous:

- **CLtot** = The lift coefficient.

- **CDtot** = The drag coefficient.

- **CSFtot** = The skin friction coefficient.

- **CMXtot** = The moment coefficient about the x-axis.

- **CMYtot** = The moment coefficient about the y-axis.

- **CMZtot** = The moment coefficient about the z-axis.

- **CXtot** = The force coefficient about the x-axis.

- **CYtot** = The force coefficient about the y-axis.

- **CZtot** = The force coefficient about the z-axis.

Pressure Forces:

- **CLtot_p** = The lift coefficient - pressure contribution only.

- **CDtot_p** = The drag coefficient - pressure contribution only.

- **CSFtot_p** = The skin friction coefficient - pressure contribution only.

- **CMXtot_p** = The moment coefficient about the x-axis - pressure contribution only.

- **CMYtot_p** = The moment coefficient about the y-axis - pressure contribution only.

- **CMZtot_p** = The moment coefficient about the z-axis - pressure contribution only.

- **CXtot_p** = The force coefficient about the x-axis - pressure contribution only.

- **CYtot_p** = The force coefficient about the y-axis - pressure contribution only.

- **CZtot_p** = The force coefficient about the z-axis - pressure contribution only.

Viscous Forces:

- **CLtot_p** = The lift coefficient - viscous contribution only.

- **CDtot_p** = The drag coefficient - viscous contribution only.

- **CSFtot_p** = The skin friction coefficient - viscous contribution only.

- **CMXtot_p** = The moment coefficient about the x-axis - viscous contribution only.

- **CMYtot_p** = The moment coefficient about the y-axis - viscous contribution only.

- **CMZtot_p** = The moment coefficient about the z-axis - viscous contribution only.

- **CXtot_p** = The force coefficient about the x-axis - viscous contribution only.

- **CYtot_p** = The force coefficient about the y-axis - viscous contribution only.

- **CZtot_p** = The force coefficient about the z-axis - viscous contribution only.

0.5 AIM Data Transfer

The SU2 AIM has the ability to transfer surface data (e.g. pressure distributions) to and from the AIM using the conservative and interpolative data transfer schemes in CAPS. Currently these transfers may only take place on triangular meshes.

0.5.1 Data transfer from SU2 (FieldOut)

- **"Cp", or "CoefficientOfPressure"**
  Loads the coefficient of pressure distribution from surface_flow_[project_name].cvs file. This distribution may be scaled based on Pressure = Pressure_Scale_Factor$*$Cp + Pressure_Scale_Offset, where "Pressure_$\leftarrow$ Scale_Factor" and "Pressure_Scale_Offset" are AIM inputs (AIM Inputs)

- **"Pressure" or "P"**
  Loads the pressure distribution from surface_flow_[project_name].cvs file. This distribution may be scaled based on Pressure = Pressure_Scale_Factor$*$Pressure + Pressure_Scale_Offset, where "Pressure_Scale$\leftarrow$ _Factor" and "Pressure_Scale_Offset" are AIM inputs (AIM Inputs)

0.5.2 Data transfer to SU2 (FieldIn)

- **"Displacement"**
  Retrieves nodal displacements (as from a structural solver) and updates SU2's surface mesh; a new [project$\leftarrow$ _name]_motion.dat file is written out which may be loaded into SU2 to update the surface mesh/move the volume mesh.

0.6 CFD Boundary Conditions

Structure for the boundary condition tuple = ("CAPS Group Name", "Value"). "CAPS Group Name" defines the capsGroup on which the boundary condition should be applied. The "Value" can either be a JSON String dictionary (see Section JSON String Dictionary) or a single string keyword string (see Section Single Value String)

0.6.1 JSON String Dictionary

If "Value" is a JSON string dictionary (eg. "Value" = {"bcType": "Viscous", "wallTemperature": 1.1}) the following keywords (= default values) may be used:

- **bcType = "Inviscid"**
  Boundary condition type. Options:

    - Inviscid

    - Viscous

    - Farfield

    - Freestream

    - BackPressure

    - Symmetry

    - SubsonicInflow

    - SubsonicOutflow

    - Internal

0.6.1.1 Wall Properties

- **wallTemperature = 0.0**
  Dimensional wall temperature for inviscid and viscous surfaces

- **wallHeatFlux = 0.0**
  Heat flux on viscous surfaces.

0.6.1.2 Stagnation Properties

- **totalPressure = 0.0**
  Dimensional total pressure on a boundary surface.

- **totalTemperature = 0.0**
  Dimensional total temperature on a boundary surface.

0.6.1.3 Static Properties

- **staticPressure = 0.0**
  Dimensional static pressure on a boundary surface.

0.6.1.4 Velocity Components

0.6.1.5 Massflow Properties

0.6.2 Single Value String

If "Value" is a single string the following options maybe used:

- "Inviscid" (default)

- "Viscous"

- "Farfield"

- "Freestream"

- "SymmetryX"

- "SymmetryY"

- "SymmetryZ"

- "Internal"

0.7 CFD Functional

Structure for the design functional tuple = {"Functional Name": "Expression"}. "Functional Name" must be one of the SU2 pre-defined functionals:
DRAG, LIFT, SIDEFORCE,
MOMENT_X, MOMENT_Y, MOMENT_Z,
EFFICIENCY, BUFFET,
EQUIVALENT_AREA, NEARFIELD_PRESSURE,
FORCE_X, FORCE_Y, FORCE_Z, THRUST,
TORQUE, TOTAL_HEATFLUX, CUSTOM_OBJFUNC,
MAXIMUM_HEATFLUX, INVERSE_DESIGN_PRESSURE,
INVERSE_DESIGN_HEATFLUX, SURFACE_TOTAL_PRESSURE,
SURFACE_MASSFLOW, SURFACE_STATIC_PRESSURE, SURFACE_MACH

and "Expression" must be an empty string.

If "Functional Name" is "CUSTOM_OBJFUNC" then "Expression" defines the objective which is parsed symbolically, e.g.:

{"CUSTOM_OBJFUNC": 'DRAG + 10 $*$ pow(fmax(0.4-LIFT, 0), 2)'}

0.8 SU2 AIM Example

This is a walkthrough for using SU2 AIM to analyze and three-dimensional two-wing configuration.

0.8.1   Prerequisites

It is presumed that ESP and CAPS have been already installed. In addition, appropriate meshing and analysis packages were available for CAPS to link them to the corresponding AIMS.

In this example, TetGen is used for volume mesh generation, and SU2 is used for flow analysis. During the CAPS/pyCAPS build process, the location of TetGen source code needs to be set in ESPenv.sh to build TetGen AIM.

0.8.1.1   Script files

Two scripts are used for this illustration:

1. cfdMultiBody.csm: Creates geometry, as described in [Part 1] of the next section

2. su2_and_Tetgen_PyTest.py: pyCAPS script for performing analysis, as described in the [Part 2] of the next section.

0.8.2   pyCAPS walkthrough using SU2

This is a two-part process. The first part consists of creating a geometry model. The second part is setting the AIMs that uses this geometry model to perform desired analysis. In this walkthrough, the geometry model is created using ESP script and setup for AIMs is done using pyCAPS.

0.8.2.1   Part 1: Creating Geometry using ESP

Step 1: The CSM script generates Bodies which are designed to be used by specific AIMs. The AIMs that the Body is designed for is communicated to the CAPS framework via the "capsAIM" string attribute. This is a semicolon-separated string with the list of AIM names. Thus, the CSM author can give a clear indication to which AIMs should use the Body. In this example, the list contains the list of mesh generators and CFD solvers that can consume the body:

```
ATTRIBUTE capsAIM $fun3dAIM;su2AIM;egadsTessAIM;aflr4AIM;pointwiseAIM;tetgenAIM;aflr3AIM #CFD Analysis
```

Step 2: A typical geometry model can be created and interactively modified using design parameters. These design parameters are either design-variable based, or geometry-variables based. In this example a two-wing configuration is created using following design parameters,

```
DESPMTR    area          40.00000
DESPMTR    aspect        5.00000
DESPMTR    taper         0.50000
DESPMTR    twist         15.00000
DESPMTR    lesweep       30.00000
DESPMTR    dihedral      1.00000
```

as well as the following configuration design paramters. Configuration quantities cannot be used with sensitivities.

```
CFGPMTR    series            8412
CFGPMTR    series2           0020
CFGPMTR    sharpte           0
CFGPMTR    wake              1
```

Step 3: Set CAPS internal attributes
```
# Set reference values
ATTRIBUTE capsReferenceArea     area
ATTRIBUTE capsReferenceChord    sqrt(area/aspect)
ATTRIBUTE capsReferenceSpan     sqrt(area/aspect)*aspect
```

Step 4: Local analytical formulation for geometry creation
```
SET        cmean     sqrt(area/aspect)
SET        span      cmean*aspect
```

```
SET      sspan      span/2
SET      croot      2*cmean/(1+taper)
SET      ctip       croot*taper
SET      xtip       sspan*tand(lesweep)
SET      ytip       sspan*tand(dihedral)
SET      ybot       -0.1*croot
SET      ytop       +0.2*croot+ytip
SET      extend     0.02*cmean
```

Step 5: Building solid model. Once all design and locale variables are defined, a half span, solid model is created by "ruling" together to NACA series airfoils (following a series of scales, rotations, and translations).

```
MARK
   UDPRIM   naca   Series   series    sharpte sharpte
      SCALE     croot
   UDPRIM   naca   Series   series2   sharpte sharpte
      SCALE     ctip
      ROTATEZ   -twist   0   0
      TRANSLATE xtip   ytip   -sspan
RULE
```

A full span model is then created by mirroring and joining the half-span model.
```
# Store half of wing and keep a copy on the stack
STORE     HalfWing 0 1
# Restore and mirror the half wing
RESTORE   HalfWing 0
   MIRROR    0   0   1   0
# Combine halfs into a whole
JOIN      0
```

Once the desired model obtained it needs to be rotated so that it is in the expected aero-coordinated system (y- out the right wing, x- in the flow direction, and +z- up).
```
# Get body into a typical aero-system
ROTATEX 90 0 0
```

Step 6: An attribute is then placed in the geometry so that the geometry components may be reference by the SU2 AIM
```
# Store the wing
STORE     Wing 0 0
# Wing 1 - Restore
RESTORE   Wing 0
   ATTRIBUTE capsGroup     $Wing1
   ATTRIBUTE capsMesh      $Wing1
   ATTRIBUTE _name         $Wing1
   ATTRIBUTE AFLR4_Cmp_ID 1
   ATTRIBUTE AFLR4_Edge_Refinement_Weight 1
```

Next a second wing is created and scaled using the store/restore operations.
```
# Wing 2 -  Restore and scale, translate
RESTORE   Wing 0
   ATTRIBUTE capsGroup         $Wing2
   ATTRIBUTE capsMesh          $Wing2
   ATTRIBUTE _name             $Wing2
   ATTRIBUTE AFLR4_Scale_Factor 10
   ATTRIBUTE AFLR4_Cmp_ID       2
   SCALE     0.4
   TRANSLATE 10   0   0
```

Step 7: For three-dimensional CFD analysis with the FUN3D AIM a "farfield" or "bounding box" boundary need to be also provided. In this example a simple sphere is created, and designated as such using the capsGroup attribute.
```
SPHERE    0   0   0   80
   ATTRIBUTE capsGroup   $Farfield
   ATTRIBUTE capsMesh    $Farfield
   ATTRIBUTE _name       $Farfield
   ATTRIBUTE AFLR_GBC    $FARFIELD_UG3_GBC
   ATTRIBUTE AFLR4_Cmp_ID 4
   ATTRIBUTE capsMeshLength  cmean #Charachteristic length for meshing
   ATTRIBUTE .tParam "30.;5.;30;"
```

Step 8: Close the ESP script
```
END
```

0.8.2.2 Part 2: Performing analysis using CAPS/pyCAPS

Step 1: Import (py)CAPS environment.
```
# Import pyCAPS module
import pyCAPS
# Import os module
import os
import argparse
# Import SU2 Python interface module
from parallel_computation import parallel_computation as su2Run
```

Step 2: Load the geometry
```
geometryScript = os.path.join("..","csmData","cfdMultiBody.csm")
myProblem = pyCAPS.Problem(problemName=workDir,
                           capsFile=geometryScript,
                           outLevel=args.outLevel)
```

Step 3: Make a list of design parameters available to interact with geometry model
```
# Change a design parameter - area in the geometry and no wake (TetGen does not support the wake)
myProblem.geometry.despmtr.area = 50
myProblem.geometry.cfgpmtr.wake = 0
```

Step 4: Load required AIMS. A typical high-fidelity CFD analysis requires mesh AIMs and an analysis AIM. For surface meshing, the face tessellation from ESP geometry can be directly used as a surface mesh. If the face tessellation is not satisfactory, an additional step of using surface AIM for external surface mesh generator will be required. Here, the face tessellation is used as surface mesh. For volume mesh generation, TetGen is used. For this, TetGen AIM is used.
```
# Load EGADS Tess aim
mySurfMesh = myProblem.analysis.create(aim = "egadsTessAIM",
                                       name = "tess")
# Load Tetgen aim
myMesh = myProblem.analysis.create(aim = "tetgenAIM",
                                   name = "myMesh")
```

Provide appropriate inputs to mesh generator required to generate mesh with adequate mesh quality and any additional features, such as boundary layer and local refinement. Refer TetGen AIM documentation for the list of all the available options.
```
# Set project name so a mesh file is generated
mySurfMesh.input.Proj_Name = "egadsTessMesh"
# Set new EGADS body tessellation parameters
mySurfMesh.input.Tess_Params = [0.5, 0.1, 20.0]
# Set output grid format since a project name is being supplied - Tecplot file
mySurfMesh.input.Mesh_Format = "Tecplot"
# Link surface mesh from EGADS to TetGen
myMesh.input["Surface_Mesh"].link(mySurfMesh.output["Surface_Mesh"])
# Preserve surface mesh while meshing
myMesh.input.Preserve_Surf_Mesh = True
```

Note that both EGADS and TetGen mesh generates are executed automaticall by CAPS:

Load and SU2 AIM and link it to TetGen AIM as a parent. This step is required so that the volume mesh generated by TetGen AIM can be used for SU2 analysis. This step allows the volume mesh generated by TetGen AIM to be converted into SU2 native format.
```
# Load SU2 aim - child of Tetgen AIM
myAnalysis = myProblem.analysis.create(aim = "su2AIM", name = "su2")
```

Set analysis parameters specific to SU2. These parameters are automatically converted into SU2 specific format and transferred into the SU2 configuration file. See AIM Inputs for the available options.
```
# Link the Mesh to the TetGen Volume_Mesh
myAnalysis.input["Mesh"].link(myMesh.output["Volume_Mesh"])
# Set SU2 Version
myAnalysis.input.SU2_Version = "Blackbird"
# Set project name
myAnalysis.input.Proj_Name = "pyCAPS_SU2_Tetgen"
# Set AoA number
myAnalysis.input.Alpha = 1.0
# Set Mach number
myAnalysis.input.Mach = 0.5901
# Set equation type
myAnalysis.input.Equation_Type = "Compressible"
# Set number of iterations
myAnalysis.input.Num_Iter = 5
# Specifcy the boundares used to compute forces
myAnalysis.input.Surface_Monitor = ["Wing1", "Wing2"]
```

Set the boundary conditions using attribution. These boundary tags and associated boundary conditions are converted into SU2 specific boundary conditions and set in the SU2 configuration file.

```
# Set boundary conditions
inviscidBC1 = {"bcType" : "Inviscid"}
inviscidBC2 = {"bcType" : "Inviscid"}
myAnalysis.input.Boundary_Condition = {"Wing1"  : inviscidBC1,
                                        "Wing2"  : inviscidBC2,
                                        "Farfield": "farfield"}
```

After all desired options are set aimPreAnalysis needs to be executed.

```
myAnalysis.preAnalysis()
```

Execute SU2 flow solver. Here the python module for parallel computation is used. The configuration file (and the corresponding mesh file) is provided automatically for executing SU2. The number of processors can be set by user, depending on the available computational resources.

```
print ("\n\nRunning SU2......")
currentDirectory = os.getcwd() # Get our current working directory
os.chdir(myAnalysis.analysisDir) # Move into test directory
su2Run(myAnalysis.input.Proj_Name + ".cfg", args.numberProc) # Run SU2
os.chdir(currentDirectory) # Move back to top directory
```

Perform post analysis task, such as parsing the final values from SU2 execution.

```
myAnalysis.postAnalysis()
```

Step 5: Print the post analysis values.

```
# Get force results
print("Total Force - Pressure + Viscous")
# Get Lift and Drag coefficients
print("Cl = " , myAnalysis.output.CLtot,
      "Cd = " , myAnalysis.output.CDtot)
# Get Cmx, Cmy, and Cmz coefficients
print("Cmx = " , myAnalysis.output.CMXtot,
      "Cmy = " , myAnalysis.output.CMYtot,
      "Cmz = " , myAnalysis.output.CMZtot)
# Get Cx, Cy, Cz coefficients
print("Cx = " , myAnalysis.output.CXtot,
      "Cy = " , myAnalysis.output.CYtot,
      "Cz = " , myAnalysis.output.CZtot)
```

0.8.3 Executing pyCAPS script

Issuing the following command executes the script:

```
python su2_and_Tetgen_PyTest.py
```

Bibliography

[1] F. Palacios, M. R. Colonno, A. C. Aranake, A. Campos, S. R. Copeland, T. D. Economon, A. K. Lonkar, T. W. Lukaczyk, T. W. R. Taylor, and J. J. Alonso. Stanford university unstructured (su2): An open-source integrated computational environment for multi-physics simulation and design. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, number 2013-0287. American Institute of Aeronautics and Astronautics, Jan. 2013. 1

[2] F. Palacios, T. D. Economon, A. C. Aranake, S. R. Copeland, A. K. Lonkar, T. W. Lukaczyk, D. E. Manosalvas, K. R. Naik, A. S. Padron, B. Tracey, A. Variyar, and J. J. Alonso. Stanford university unstructured (su2): Open-source analysis and design technology for turbulent flows. In *52nd Aerospace Sciences Meeting*, number 2014-0243. American Institute of Aeronautics and Astronautics, Jan. 2014. 1