

## SU2 Analysis Interface Module (AIM)

Nitin Bhagat  
UDRI

Ryan Durscher  
AFRL/RQVC

December 22, 2017

## Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                                      | <b>1</b>  |
| 1.1      | SU2 AIM Overview . . . . .                               | 1         |
| 1.1.1    | Automatic generation of SU2 Mesh file . . . . .          | 2         |
| 1.1.2    | Automatic generation of SU2 Configuration file . . . . . | 2         |
| 1.2      | SU2 Examples . . . . .                                   | 2         |
| <b>2</b> | <b>Geometry Representation and Analysis Intent</b>       | <b>2</b>  |
| <b>3</b> | <b>AIM Inputs</b>  | <b>2</b>  |
| <b>4</b> | <b>AIM Shareable Data</b>                                | <b>4</b>  |
| <b>5</b> | <b>AIM Outputs</b>                                       | <b>4</b>  |
| <b>6</b> | <b>SU2 Data Transfer</b>                                 | <b>5</b>  |
| 6.1      | Data transfer from SU2 . . . . .                         | 5         |
| 6.2      | Data transfer to SU2 . . . . .                           | 6         |
| <b>7</b> | <b>CFD Boundary Conditions</b>                           | <b>6</b>  |
| 7.1      | JSON String Dictionary . . . . .                         | 6         |
| 7.1.1    | Wall Properties . . . . .                                | 6         |
| 7.1.2    | Stagnation Properties . . . . .                          | 6         |
| 7.1.3    | Static Properties . . . . .                              | 6         |
| 7.1.4    | Velocity Components . . . . .                            | 7         |
| 7.1.5    | Massflow Properties . . . . .                            | 7         |
| 7.2      | Single Value String . . . . .                            | 7         |
| <b>8</b> | <b>SU2 AIM Example</b>                                   | <b>7</b>  |
| 8.1      | Prerequisites . . . . .                                  | 7         |
| 8.1.1    | Script files . . . . .                                   | 7         |
| 8.2      | pyCAPS walkthrough using SU2 . . . . .                   | 7         |
| 8.2.1    | Part 1: Creating Geometry using ESP . . . . .            | 7         |
| 8.2.2    | Part 2: Performing analysis using CAPS/pyCAPS . . . . .  | 9         |
| 8.3      | Executing pyCAPS script . . . . .                        | 11        |
|          | <b>Bibliography</b>                                      | <b>12</b> |

## 1 Introduction

### 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.

The accepted and expected geometric representation and analysis intentions are detailed in [Geometry Representation and Analysis Intent](#).

Details of the AIM's shareable data structures are outlined in [AIM Shareable Data](#) if connecting this AIM to other AIMS in a parent-child like manner.

Details of the AIM's automated data transfer capabilities are outlined in [SU2 Data Transfer](#)

### 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.

### 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.

## 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.

## 2 Geometry Representation and Analysis Intent

The geometric representation for the SU2 AIM requires that the body(ies) be either a face body(ies) (FACEBODY), solid body(ies) (SOLIDBODY) or non-manifold sheet body(ies) (SHEETBODY). For 2D simulations the attribute capsIntent should be set to CFD with the geometric representation being either a planar SHEETBODY or FACEBODY, while for 3D simulations it should again be set to CFD, but both SOLIDBODY(ies) or SHEETBODY(ies) are acceptable.

## 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) or 5.0.0 (Raven - Default)

- **Proj\_Name = "su2\_CAPS"**  
This corresponds to the project "root" name.
- **Mach = 0.8**  
Mach number; this corresponds to the MACH\_NUMBER keyword in the configuration file.
- **Re = 6.5E6**  
Reynolds number; this corresponds to the REYNOLDS\_NUMBER keyword in the configuration file.
- **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.
- **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.
- **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  $\text{Pressure} = \text{Pressure}_{\leftrightarrow} \text{Scale\_Factor} * \text{Cp} + \text{Pressure\_Scale\_Offset}$ .
- **Pressure\_Scale\_Offset = 0.0**  
Value to offset Cp or Pressure data when transferring data. Data is scaled based on  $\text{Pressure} = \text{Pressure}_{\leftrightarrow} \text{Scale\_Factor} * \text{Cp} + \text{Pressure\_Scale\_Offset}$ .
- **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\_METHO↔\_D\_FLOW keyword in the configuration file. See SU2 template for additional details.
- **SU2\_Version = "Raven"**  
SU2 version to generate specific configuration file for. Options: "Cardinal" or "Raven".

## 4 AIM Shareable Data

Currently the SU2 AIM does not have any shareable data types or values. It will try, however, to inherit a "Volume\_↔\_Mesh" (for 3D simulations) or a "Surface\_Mesh" (for 2D simulations) from any parent AIMs. Note that the inheritance of the volume/surface mesh variable is required if the SU2 aim is to write a suitable grid file.

## 5 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.

## 6 SU2 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.

### 6.1 Data transfer from SU2

- **"Cp", or "CoefficientofPressure"**  
Loads the coefficient of pressure distribution from surface\_flow\_[project\_name].cvs file. This distribution may be scaled based on  $\text{Pressure} = \text{Pressure\_Scale\_Factor} * \text{Cp} + \text{Pressure\_Scale\_Offset}$ , where "Pressure\_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  $\text{Pressure} = \text{Pressure\_Scale\_Factor} * \text{Pressure} + \text{Pressure\_Scale\_Offset}$ , where "Pressure\_Scale\_Factor" and "Pressure\_Scale\_Offset" are AIM inputs ([AIM Inputs](#))

## 6.2 Data transfer to SU2

- **"Displacement"**

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

## 7 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](#))

### 7.1 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

#### 7.1.1 Wall Properties

- **wallTemperature = 0.0**

Dimensional wall temperature for inviscid and viscous surfaces

- **wallHeatFlux = 0.0**

Heat flux on viscous surfaces.

#### 7.1.2 Stagnation Properties

- **totalPressure = 0.0**

Dimensional total pressure on a boundary surface.

- **totalTemperature = 0.0**

Dimensional total temperature on a boundary surface.

#### 7.1.3 Static Properties

- **staticPressure = 0.0**

Dimensional static pressure on a boundary surface.

### 7.1.4 Velocity Components

### 7.1.5 Massflow Properties

## 7.2 Single Value String

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

- "Inviscid" (default)
- "Viscous"
- "Farfield"
- "Freestream"
- "SymmetryX"
- "SymmetryY"
- "SymmetryZ"

## 8 SU2 AIM Example

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

### 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 C<sub>++</sub> APS/pyCAPS build process, the location of TetGen source code needs to be set in ESPenv.sh to build TetGen AIM.

#### 8.1.1 Script files

Two scripts are used for this illustration:

1. `cfMultiBody.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.

## 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.

### 8.2.1 Part 1: Creating Geometry using ESP

Step 1: Setting up the CAPS Fidelity is the first step. This is required to provide CAPS information about all the geometries models available for analysis. In this example, the geometry model generated can be used for CFD analysis as shown:



```
attribute capsIntent CFD
```

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 series[1,1]      8412.00000
despmtr series2[1,1]     20.00000
despmtr area[1,1]        40.00000
despmtr aspect[1,1]      5.00000
despmtr taper[1,1]       0.50000
despmtr twist[1,1]       15.00000
despmtr lesweep[1,1]     30.00000
despmtr dihedral[1,1]    1.00000
```

### 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
  scale croot
  udprim naca Series series2
  scale ctip
  rotatez -twist 0 0
  translate xtip ytip -sspan
rule
  0
```

A full span model is then created by mirroring and joining the half-span model.

```
# Store half of wing
store HalfWing 0 0

# Restore the half wing
restore HalfWing 0

# Restore and mirror the half wing
restore HalfWing 0
mirror 0 0 1 0

# Combine halves 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
```

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

          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
```

Step 8: Close the ESP script

```
end
```

## 8.2.2 Part 2: Performing analysis using CAPS/pyCAPS

Step 1: Import (py)CAPS environment.

```
from __future__ import print_function

# Import pyCAPS class file
try:
    import pyCAPS
except:
    print ("Unable to import pyCAPS module")
    raise SystemError

# Import os module
try:
    import os
except:
    print ("Unable to import os module")
    raise SystemError

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

Step 2: Initialize

```
myProblem = pyCAPS.capsProblem()
```

Step 3: Load the geometry

```
myProblem.loadCAPS("./csmData/cfdMultiBody.csm")
```

Step 4: Extract the geometry model specific to the desired analysis

```
myProblem.capsIntent = "CFD"
```

Step 5: Make a list of design parameters available to interact with geometry model

```
# Change a design parameter - area in the geometry
myProblem.geometry.setGeometryVal("area", 50)
```

Step 6: 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.

```
myMesh = myProblem.loadAIM(aim = "tetgenAIM", analysisDir= ".")
```

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 new EGADS body tessellation parameters
myMesh.setAnalysisVal("Tess_Params", [.05, 0.01, 20.0])

# Preserve surface mesh while meshing
myMesh.setAnalysisVal("Preserve_Surf_Mesh", True)
```

After options are set aimPreAnalysis needs to be executed:

```
myMesh.aimPreAnalysis()
```

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
su2 = myProblem.loadAIM(aim = "su2AIM",
                        altName = "su2",
                        analysisDir = "SU2TetgenAnalysisTest", parents = ["tetgenAIM"])
```

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.

```
# Set SU2 Version
su2.setAnalysisVal("SU2_Version", "Raven")

projectName = "pyCAPS_SU2_Tetgen"
# Set project name
su2.setAnalysisVal("Proj_Name", projectName)

# Set AoA number
su2.setAnalysisVal("Alpha", 1.0)

# Set Mach number
su2.setAnalysisVal("Mach", 0.5901)

# Set equation type
su2.setAnalysisVal("Equation_Type", "Compressible")

# Set number of iterations
su2.setAnalysisVal("Num_Iter", 10)
```

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.

```
inviscidBC1 = {"bcType" : "Inviscid"}
inviscidBC2 = {"bcType" : "Inviscid"}
su2.setAnalysisVal("Boundary_Condition", [(("Wing1", inviscidBC1),
                                           ("Wing2", inviscidBC2),
                                           ("Farfield", "farfield"))])
```

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

```
su2.aimPreAnalysis()
```

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(su2.analysisDir) # Move into test directory

numberProc = 4;
su2Run(projectName + ".cfg", numberProc) # Run SU2

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

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

```
su2.aimPostAnalysis()
```

Step 7: Print the post analysis values.

```
# Get force results
print ("Total Force - Pressure + Viscous")
# Get Lift and Drag coefficients
print ("Cl = " , su2.getAnalysisOutVal("CLtot"), \
      "Cd = " , su2.getAnalysisOutVal("CDtot"))

# Get Cmx, Cmy, and Cmc coefficients
print ("Cmx = " , su2.getAnalysisOutVal("CMXtot"), \
      "Cmy = " , su2.getAnalysisOutVal("CMYtot"), \
      "Cmc = " , su2.getAnalysisOutVal("CMZtot"))

# Get Cx, Cy, Cz coefficients
print ("Cx = " , su2.getAnalysisOutVal("CXtot"), \
      "Cy = " , su2.getAnalysisOutVal("CYtot"), \
      "Cz = " , su2.getAnalysisOutVal("CZtot"))
```

Step 8: Close the script upon normal termination.

```
myProblem.closeCAPS()
```

### 8.3 Executing pyCAPS script

Issuing the following command executes the script:

```
python su2_and_Tetgen_PyTest.py
```

## References

- [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](#)