XFOIL Analysis Interface Module (AIM)

Ryan Durscher
AFRL/RQVC

June 12, 2017

Contents

1 Introduction

1.1 xFoil AIM Overview

A module in the Computational Aircraft Prototype Syntheses (CAPS) has been developed to interact (through input files) with the subsonic airfoil analysis tool xFoil [1]. xFoil is an open-source tool and may be freely downloaded from `http://web.mit.edu/drela/Public/web/xfoil/` . At this time only a subsection of xFoil's capabilities are exposed through the AIM. Furthermore, only versions 6.97 and 6.99 of xFoil have been tested against (for Windows only 6.99).

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.

Upon running preAnalysis the AIM generates two files: 1. "xfoilInput.txt" which contains instructions for xFoil to execute and 2. "caps.xfoil" which contains the the geometry to be analyzed.

1.2 Assumptions

xFoil inherently assumes the airfoil cross-section is in the x-y plane, if it isn't an attempt is made to automatically rotate the provided body.

Within **OpenCSM** there are a number of airfoil generation UDPs (User Defined Primitives). These include NACA 4 series, a more general NACA 4/5/6 series generator, Sobieczky's PARSEC parameterization and Kulfan's CST parameterization. All of these UDPs generate **EGADS** *FaceBodies* where the *Face*'s underlying *Surface* is planar and the bounds of the *Face* is a closed set of *Edges* whose underlying *Curves* contain the airfoil shape. In all cases there is a *Node* that represents the *Leading Edge* point and one or two *Nodes* at the *Trailing Edge* – one if the representation is for a sharp TE and the other if the definition is open or blunt. If there are 2 *Nodes* at the back,

then there are 3 *Edges* all together and closed, even though the airfoil definition was left open at the TE. All of this information will be used to automatically fill in the xFoil geometry description.

It should be noted that general construction in either **OpenCSM** or even **EGADS** will be supported as long as the topology described above is used. But care should be taken when constructing the airfoil shape so that a discontinuity (i.e., simply C^0) is not generated at the *Node* representing the *Leading Edge*. This can be done by splining the entire shape as one and then intersecting the single *Edge* to place the LE *Node*.

1.3 Examples

An example problem using the xFoil AIM may be found at xFoil AIM Example .

2 Geometry Representation and Analysis Intent

The geometric representation for the XFOIL AIM requires the airfoil cross-section, though the global attribute capsIntent, be set to ALL or LINEARAERO.

3 AIM Inputs

The following list outlines the xFoil inputs along with their default values available through the AIM interface.

- **Mach = 0.0**
  Mach number.
- **Re = 0.0**
  Reynolds number.
- **Alpha = NULL**
  Angle of attack, either a single value or an array of values ([0.0, 4.0, ...]) may be provided [degree].
- **Alpha_Increment = NULL**
  Angle of attack [degree] sequence - [first value, last value, increment].
- **CL = NULL**
  Prescribed coefficient of lift, either a single value or an array of values ([0.1, 0.5, ...]) may be provided.
- **CL_Increment = NULL**
  Prescribed coefficient of lift sequence - [first value, last value, increment].
- **CL_Inviscid = NULL**
  Prescribed inviscid coefficient of lift, either a single value or an array of values ([0.1, 0.5, ...]) may be provided.
- **Append_PolarFile = False**
  Append the file (xfoilPolar.dat) that polar data is written to.
- **Viscous_Iteration = 20**
  Viscous solution iteration limit. Only set if a Re isn't 0.0 .
- **Write_Cp = False**
  Have xFoil write the coefficient of pressure (Cp) distribution to a file - xfoilCp.dat.

4 AIM Outputs

The following list outlines the xFoil outputs available through the AIM interface.

- **Alpha =** Angle of attack value(s).

- **CL =** Coefficient of lift value(s).
- **CD =** Coefficient of drag value(s).
- **CD_p =** Coefficient of drag value(s), pressure contribution.
- **CM =** Moment coefficient value(s).
- **Cp_Min =** Minimum coefficient of pressure value(s).
- **Transition_Top =** Value(s) of x- transition location on the top of the airfoil.
- **Transition_Bottom =** Value(s) of x- transition location on the bottom of the airfoil.

5 xFoil AIM Example

This is a walkthrough for using xFoil AIM to analyze a single airfoil cross-section.

5.1 Prerequisites

It is presumed that ESP and CAPS have been already installed, as well as xFoil.

5.1.1 Script files

Two scripts are used for this illustration:

1. airfoilSection.csm: Creates geometry, as described in [Part 1] of the next section
2. xfoil_PyTest.py: pyCAPS script for performing analysis, as described in the [Part 2] of the next section.

5.2 Creating Geometry using ESP

In our example $*$.csm file setting up the CAPS fidelity is the first step. If multiple bodies exist in the $*$.csm file the tag, capsIntent, can be used to distinguish what type of analysis the body may be used for. In this example, the geometry model generated can be used for CFD analysis, as shown:

```
attribute capsIntent       LINEARAERO
```

A typical geometry model can be created and interactively modified using design parameters. These design parameters are either design- or geometry- based. In this example a single airfoil cross-section is created using the following design parameters.

```
despmtr   thick     0.12      frac of local chord
despmtr   camber    0.00      frac of local chord
despmtr   area      10.0      Planform area of the full span wing
despmtr   aspect    6.00      Span^2/Area
despmtr   taper     0.60      TipChord/RootChord
```

After our design parameters are defined they are used to setup other local variables (analytically) for the wing.

```
set       span      sqrt(aspect*area)
set       croot     2*area/span/(1+taper)
set       ctip      croot*taper
```

Once all design and locale variables are defined, a single airfoil cross-section is created using the NACA series airfoils (following a scale).

```
udprim    naca      Thickness thick     Camber    camber sharpte 1
scale     ctip
```

5.3 Performing analysis using pyCAPS

The first step in the pyCAPS script is to import the required modules. For this example the following modules are used,

```
from __future__ import print_function

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

In order to create a new capsProblem the pyCAPS module also needs to be imported; on Linux and OSx this is the pyCAPS.so file, while on Windows it is the pyCAPS.pyd file. For convenience, it is recommended that the path to this file is added to the environmental variable PYTHONPATH.

```
from pyCAPS import capsProblem
```

Similarly, local variables used throughout the script may be defined.

```
workDir = "xFoilAnalysisTest"
```

Once the required modules have been loaded, a capsProblem can be instantiated.

```
myProblem = capsProblem()
```

Using the loadCAPS() function, the desired geometry file is then loaded into the problem.

```
myGeometry = myProblem.loadCAPS("./csmData/airfoilSection.csm")
```

Any design parameters available in $*$.csm file are also available within the pyCAPS script. The following snippet changes the despmtrs "area" which will force a rebuild of the geometry that xFoil will now use.

```
myGeometry.setGeometryVal("camber", 0.1)
```

When loading each analysis tools the desired capsIntent needs to be specified. Optionally this may be set at the problem level as follows to avoid redundancy.

```
myProblem.capsIntent = "LINEARAERO"
```

Next the xFoil AIM needs to be loaded.

```
xfoil = myProblem.loadAIM(aim = "xfoilAIM",
                          analysisDir = workDir)
```

Once loaded analysis parameters specific to xFoil need to be set (see AIM Inputs). These parameters are automatically converted into xFoil specific format and transferred into the xFoil configuration file.

```
# Set Mach number, Reynolds number
xfoil.setAnalysisVal("Mach", 0.5 )
xfoil.setAnalysisVal("Re", 1.0E6 )

# Set custom AoA
xfoil.setAnalysisVal("Alpha", [0.0, 3.0, 5.0, 7.0, 9.0, 11, 13, 14, 15.0])

# Set AoA seq
xfoil.setAnalysisVal("Alpha_Increment", [1.0, 2.0, 0.10])

# Set custom Cl
xfoil.setAnalysisVal("CL", 0.1)

# Set Cl seq
xfoil.setAnalysisVal("CL_Increment", [0.8, 3, .25])
```

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

```
xfoil.aimPreAnalysis()
```

At this point the required files necessary run xFoil should have be created and placed in the specified analysis working directory. Next xFoil needs to executed such as through an OS system call like,

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

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

os.system("xfoil < xfoilInput.txt > Info.out"); # Run xfoil via system call

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

After xFoil is finished running aimPostAnalysis needs to be executed.

```
xfoil.aimPostAnalysis()
```

Finally available AIM outputs (see AIM Outputs) may be retrieved, for example:

```
# Retrieve Cl and Cd
Cl = xfoil.getAnalysisOutVal("CL")
print("Cl = ", Cl)

Cd = xfoil.getAnalysisOutVal("CD")
print("Cd = ", Cd)

# Angla of attack
Alpha = xfoil.getAnalysisOutVal("Alpha")
print("Alpha = ", Alpha)

# Transition location
TranX = xfoil.getAnalysisOutVal("Transition_Top")
print("Transition location = ", TranX)
```

results in,

```
Cl =  [0.951, 1.2403, 1.4266, 1.6214, 1.7724, 1.8756, 1.8507, 1.7748, 1.6766, 1.0562,
       1.0665, 1.0767, 1.0869, 1.0969, 1.107, 1.1168, 1.1264, 1.1356, 1.1448, 1.05,
       1.3, 1.55, 1.55, 1.55, 1.55, 1.55, 1.55, 1.55, 1.55, 1.55, 1.55, 1.55]
Cd =  [0.01293, 0.016, 0.01916, 0.02326, 0.03022, 0.04169, 0.06602, 0.08659, 0.11208,
       0.01387, 0.01394, 0.01403, 0.01412, 0.01422, 0.01429, 0.01439, 0.0145, 0.01462,
       0.01476, 0.0139, 0.01704, 0.02139, 0.02139, 0.02139, 0.02139, 0.02139, 0.02139,
       0.02139, 0.02139, 0.02139, 0.02139, 0.02139]
Alpha =  [0.0, 3.0, 5.0, 7.0, 9.0, 11.0, 13.0, 14.0, 15.0, 1.1, 1.2, 1.3, 1.4, 1.5,
          1.6, 1.7, 1.8, 1.9, 2.0, 1.055, 3.66, 6.227, 6.227, 6.227, 6.227, 6.227,
          6.227, 6.227, 6.227, 6.227, 6.227, 6.227]
```

When finally finished with the script, the open CAPS problem should be closed.

```
myProblem.closeCAPS()
```

5.4 Executing pyCAPS script

Issuing the following command executes the script:

```
python xfoil_PyTest.py
```

Below is a representative image obtained by plotting the data presented above:

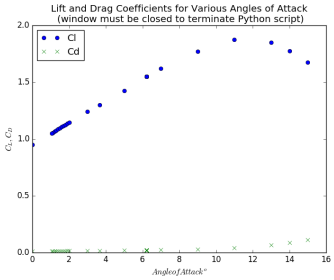

Figure 1: xFoil generated lift and drag coefficients for various angles of attack

References

[1] Mark Drela. *XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils*, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989. 1