XFOIL Analysis Interface Module (AIM) Manual

Ryan Durscher
AFRL/RQVC

September 17, 2025

0.1 Introduction

0.1.1 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 version 6.99 of xFoil have been tested against.

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

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 geometry to be analyzed.

The xFoil AIM can automatically execute xfoil, with details provided in AIM Execution.

0.1.2 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.

0.1.2.1 Airfoils in ESP

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.

Important Airfoil Geometry Assumptions

- There must be a *Node* that represents the *Leading Edge* point
- For a sharp trailing edge, there must be a *Nodes* at the *Trailing Edge*
- For a blunt trailing edge, the airfoil curve may be open, or closed by a single *Edge* connecting the upper/lower *Nodes*
- For a *FaceBody*, the airfoil coordinates traverse counter-clockwise around the *Face* normal. The **OpenCSM** *REORDER* operation may be used to flip the *Face* normal.
- For a *WireBody*, the airfoil coordinates traverse in the order of the loop

Note: Additional spurious *Nodes* on the upper and lower *Edges* of the airfoil are acceptable.

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*.

0.1.3 Examples

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

0.2 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 [degree], either a single value or an array of values ([0.0, 4.0, ...]) may be provided.
- **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 = 100**
  Viscous solution iteration limit. Only set if a Re isn't 0.0 .
- **Num_Panel = 200**
  Number of discrete panels.
- **LETE_Panel_Density_Ratio = 0.25**
  Panel density ratio between LE/TE.
- **Write_Cp = False**
  Have xFoil write the coefficient of pressure (Cp) distribution to a file - xfoilCp.dat.

0.3 AIM Execution

If auto execution is enabled when creating an xFoil AIM, the AIM will execute xFoil just-in-time with the command line:

```
xfoil < xfoilInput.txt > xfoilOutput.txt
```

where preAnalysis generated the file "xFoilInput.txt" which contains the input information.

The analysis can be also be explicitly executed with caps_execute in the C-API or via Analysis.runAnalysis in the pyCAPS API.

Calling preAnalysis and postAnalysis is NOT allowed when auto execution is enabled.

Auto execution can also be disabled when creating an xFoil AIM object. In this mode, caps_execute and Analysis.↩ runAnalysis can be used to run the analysis, or xFoil can be executed by calling preAnalysis, system call, and posAnalysis as demonstrated below with a pyCAPS example:

```
print ("\n\preAnalysis......")
xfoil.preAnalysis()

print ("\n\nRunning......")
xfoil.system("xfoil < xfoilInput.txt > xfoilOutput.txt"); # Run via system call

print ("\n\postAnalysis......")
xfoil.postAnalysis()
```

0.4 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.
- **LE_Rbar =** Leading edge radius of the airfoil, normalized by chord.
- **Chord =** Input airfoil chord length.

0.5 xFoil AIM Example

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

0.5.1 Prerequisites

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

0.5.1.1 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.

0.5.2 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
attribute capsAIM $xfoilAIM;tsfoilAIM;msesAIM
```

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
```

0.5.3 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,

```
# Import pyCAPS module
import pyCAPS

import os
import argparse
```

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

```
# Create working directory variable
workDir = "xFoilAnalysisTest"
workDir = os.path.join(str(args.workDir[0]), workDir)
```

Once the required modules have been loaded, a pyCAPS.Problem can be instantiated with the desired geometry file.

```
geometryScript = os.path.join("..","csmData","airfoilSection.csm")
myProblem = pyCAPS.Problem(problemName=workDir,
                           capsFile=geometryScript,
                           outLevel=args.outLevel)
```

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

```
# Change a design parameter - area in the geometry
myProblem.geometry.despmtr.camber = 0.1
```

Next the xFoil AIM needs to be loaded.

```
# Load xfoil aim
xfoil = myProblem.analysis.create(aim = "xfoilAIM")
```

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.input.Mach = 0.5
xfoil.input.Re   = 1.0e6

# Set custom AoA
xfoil.input.Alpha = [0.0, 3.0, 5.0, 7.0, 9.0, 11, 13, 14, 15.0]

# Set AoA seq
#xfoil.input.Alpha_Increment = [1.0, 2.0, 0.10]

# Set custom Cl
#xfoil.input.CL = 0.1

# Set Cl seq
#xfoil.input.CL_Increment = [0.8, 3, .25]

# Append the polar file if it already exists - otherwise the AIM will delete the file
xfoil.input.Append_PolarFile = True
```

The xFoil AIM auto executes xfoil just-in-time when oututs are requested (AIM Execution).

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

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

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

# Angle of attack
Alpha = xfoil.output.Alpha
print("Alpha = ", Alpha)

# Transition location
TranX = xfoil.output.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]
```

0.5.4 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:

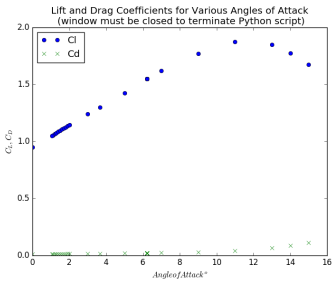

Figure 1 xFoil generated lift and drag coefficients for various angles of attack

Bibliography

[1] Mark Drela. Xfoil: An analysis and design system for low reynolds number airfoils. In Thomas J. Mueller, editor, *Low Reynolds Number Aerodynamics: Proceedings of the Conference Notre Dame, Indiana, USA, 5–7 June 1989*, pages 1–12, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg. 1

Index