FRICTION Analysis Interface Module (AIM) Manual

Ed Alyanak and Ryan Durscher
AFRL/RQVC

November 28, 2025

0.1 Introduction

0.1.1 FRICTION AIM Overview

FRICTION provides an estimate of laminar and turbulent skin friction and form drag suitable for use in aircraft preliminary design [1]. Taken from the FRICTION manual: "The program has its roots in a program by Ron Hendrickson at Grumman. It runs on any computer. The input requires geometric information and either the Mach and altitude combination, or the Mach and Reynolds number at which the results are desired. It uses standard flat plate skin friction formulas. The compressibility effects on skin friction are found using the Eckert Reference Temperature method for laminar flow and the van Driest II formula for turbulent flow. The basic formulas are valid from subsonic to hypersonic speeds, but the implementation makes assumptions that limit the validity to moderate supersonic speeds (about Mach 3). The key assumption is that the vehicle surface is at the adiabatic wall temperature (the user can easily modify this assumption). Form factors are used to estimate the effect of thickness on drag, and a composite formula is used to include the effect of a partial run of laminar flow."

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

The AIM requires dimensional units which are described in AIM Units.

Friction drag is estimated based on laminar (White, Viscous Fluid Flow, 1974 ed., pg 589-590.) and turbulent (an Driest II Method, NASA TN D-6945) flat plate skin friction. The reference length for lifting surfaces is estimated as the average segment chord length between two airfoil sections, and the total drag is an wetted area weighted sum of the segments. For bodies of revolution, the references length is the total length of body (nose-to-tail). The form factor for lifting surfaces is based on the average thickness to chord ratio for each segment, and the maximum diameter (estimated as $d = 2 * \sqrt{(A_{sec}/\pi)}$) to length ratio.

0.1.2 FRICTION Modifications

While FRICTION is available from, `FRICTION download`, the AIM has re-implemented the calculations in memory. This has enabled the additional inputs not available in the original Fortran software.

The AIM previously used FRICTION source is still provided as part of CAPS, and contains a few modifications from the original source code. These modificiations allows for longer input and output file name lengths, as well as other I/O modifications. This modified version of FRICTION, friction_eja_mod.f, is supplied and built with the AIM. During the compilation the source code is compiled into an executable with the name *friction* (Linux and OSX) or *friction.exe* (Windows).

0.1.3 Examples

An example problem using the FRICTION AIM may be found at FRICTION AIM Examples.

0.2 AIM Attributes

The following list of attributes drives the FRICTION geometric definition. Aircraft components are defined as cross sections in the low fidelity geometry definition. To be able to logically group the cross sections into wings, tails, fuselage, etc they must be given a grouping attribute. This attribute defines a logical group along with identifying a set of cross sections as a lifting surface or a body of revolution. The format is as follows.

- **capsGroup** This string attribute labels the *FaceBody* as to which type the section is assigned. This information is also used to logically group sections together by type to create wings, tails, stores, etc.
- **capsReferenceArea** [Optional: Default 1.0] This attribute may exist on any *Body*. Its value will be used as the SREF entry in the FRICTION input.
- **capsLength** This attribute defines the length units that the $*$.csm file is generated in. Friction input MUST be in units of feet. The AIM handles all unit conversion internally based on this input.

0.3 Friction Surface

Structure for the Friction Surface tuple = ("Name of Surface", "Value"). "Name of surface defines the name of the surface in which the data should be applied. The "Value" must be be a JSON String dictionary. @section json↵ StringSurface JSON String Dictionary If "Value" is a JSON string dictionary (eg. "Value" = {"BL_Transition": 0.1}) the following keywords (= default values) may be used: <ul> <B>groupName = "(no default)"</B>
 Single or list of capsType names used to define the surface (e.g. "Name1" or ["Name1","Name2",...]. If no groupName variable is provided an attempted will be made to use the tuple name instead;   <B>BL_Transition = NULL</B>
 Specify transition location in range [0-1] for the surface. Defaults to global BL_Transition input. \iline 861 <ul> <B>TwTaw = 1.0</B>
 Tw/Taw, the wall temperature ratio. \iline 877 @section keyStringVLMSurface Single Value String If "Value" is a single string the following options maybe used:

- (NONE Currently)

0.4 AIM Units

A unit system may be specified as "SI" or "US". For "SI", the internal calculations will be in meters, for "US" in feet. The default is "US".

0.5 AIM Inputs

The following list outlines the FRICTION inputs along with their default values available through the AIM interface. All inputs to the FRICTION AIM are variable length arrays. **All inputs must be the same length** .

- **Mach = double**
  OR
- **Mach = [double, ... , double]**
  Mach number.
  Must have the same number of entries as Altitude.
- **Altitude = double**
  OR
- **Altitude = [double, ... , double]**
  Altitude in units of ft for 'US' units and m for 'SI' units. Must have the same number of entries as Mach.
- **BL_Transition = double [0.1 default]**
  Boundary layer laminar to turbulent transition percentage [0.0 turbulent to 1.0 laminar] location for all surfaces.
- **TwTaw = double [1.0 default]**
  Tw/Taw, the wall temperature ratio for all surfaces.
- **LiftSurface = NULL**
  Defines lifting surfaces. See Friction Surface for additional details.
- **RevolveSurface = NULL**
  Defines revolution surfaces. See Friction Surface for additional details.

0.6 AIM Outputs

Total, Form, and Friction drag components:

- **CDtotal =** Total Drag Coefficient [CDform + CDfric] for each analysis case.
- **CDform =** Total Form Drag Coefficient for each analysis case.
- **CDfric =** Total Friction Drag Coefficient for each analysis case.
- **Swet =** Total wetted area.

Geometric component are also available as dynamic outputs. In addition to the above these include:

- **Swet =** Component wetted area
- **RefL =** Reference length (average chord for lifting surface, total length for body of revolution)
- **ReL =** Component Reynolds number based on RefL
- **ToC =** Wetted area averaged Thickness to Chord ratio for lifting surfaces (max Diameter to Length for body of revolution).
- **FormFactor =** The Form Factor for the surface (area averaged for lifting surfaces)

0.7 FRICTION AIM Examples

This is a walkthrough for using FRICTION AIM to analyze a wing, tail, fuselage configuration.

0.7.1 Prerequisites

It is presumed that ESP and CAPS have been already installed, as well as FRICTION. Furthermore, a user should have knowledge on the generation of parametric geometry in Engineering Sketch Pad (ESP) before attempting to integrate with any AIM. Specifically this example makes use of Design Parameters, Set Parameters, User Defined Primitive (UDP) and attributes in ESP.

0.7.1.1 Script files

Two scripts are used for this illustration:

1. frictionWingTailFuselage.csm: Creates geometry, as described in the following section.
2. friction_PyTest.py: pyCAPS script for performing analysis, as described in Performing analysis using pyCAPS.

0.7.2 Creating Geometry using ESP

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 only the frictionAIM:

```
attribute capsAIM $frictionAIM
```

FRICTION input is always in feet, to enable automatic conversion, the geometric attribute **capsLength** may be used to define the units the geometry (*.csm) file is in.

```
attribute capsLength $m
```

Next we will define the design parameters to define the wing cross section and planform.

```
despmtr   thick     0.12      frac of local chord
despmtr   camber    0.04      frac of loacl chord
despmtr   tlen      5.00      length from wing LE to Tail LE
despmtr   toff      0.5       tail offset

despmtr   area      10.0
despmtr   aspect    6.00
despmtr   taper     0.60
despmtr   sweep     20.0      deg (of c/4)

despmtr   washout  -5.00      deg (down at tip)
despmtr   dihedral  4.00      deg
```

The design parameters will then be used to set parameters for use internally to create geometry.

```
set       span      sqrt(aspect*area)
set       croot     2*area/span/(1+taper)
set       ctip      croot*taper
set       dxtip     (croot-ctip)/4+span/2*tand(sweep)
set       dztip     span/2*tand(dihedral)
```

Next the Wing, Vertical and Horizontal tails are created using the *naca* User Defined Primitive (UDP). The inputs used for this example to the UDP are Thickness and Camber. The naca sections generated are in the X-Y plane and are rotated to the X-Z plane. They are then translated to the appropriate position based on the design and set parameters defined above. Finally reference area can be given to the FRICTION AIM by using the **caps↩ ReferenceArea** attribute. If this attribute exists on any body that value is used otherwise the default is 1.0.

In addition, each section has a **capsGroup** attribute. This is used to logically group sections together. More information on this can be found in the AIM Attributes section.

```
# right tip
udprim    naca      Thickness thick     Camber    camber
attribute capsReferenceArea area
attribute capsGroup   $Wing
scale     ctip
rotatex   90        0         0
rotatey   washout   0         ctip/4
translate dxtip    -span/2    dztip

# root
udprim    naca      Thickness thick     Camber    camber
attribute capsGroup   $Wing
rotatex   90        0         0
scale     croot

# left tip
udprim    naca      Thickness thick     Camber    camber
attribute capsGroup  $Wing
scale     ctip
rotatex   90        0         0
rotatey   washout   0         ctip/4
translate dxtip     span/2    dztip
```

Vertical Tail definition

```
# tip
udprim    naca      Thickness thick
attribute capsGroup  $VTail
scale     0.75*ctip
translate tlen+0.75*(croot-ctip) 0.0 ctip+toff

# root
udprim    naca      Thickness thick
attribute capsGroup  $VTail
scale     0.75*croot
```

```
translate tlen 0.0 toff
```

Horizontal Tail definition

```
# tip left
udprim    naca      Thickness thick
attribute capsGroup  $HTail
scale     0.75*ctip
rotatex   90         0         0
translate tlen+0.75*(croot-ctip) -ctip toff

# tip right
udprim    naca      Thickness thick
attribute capsGroup  $HTail
scale     0.75*ctip
rotatex   90         0         0
translate tlen+0.75*(croot-ctip) ctip toff
```

Fuselage definition. Notice the use of the ellipse UDP. In this case, only translation is required to move the cross section into the desired location.

```
point -0.4*tlen 0.0 0.0
attribute capsGroup $Fuse

udprim  ellipse ry 0.5*croot rz 0.2*croot
attribute capsGroup $Fuse
translate 0.0 0.0 0.0

udprim  ellipse ry 0.4*croot rz 0.1*croot
attribute capsGroup $Fuse
translate croot 0.0 0.0

udprim  ellipse ry 0.1*croot rz 0.1*croot
attribute capsGroup $Fuse
translate tlen 0.0 toff

udprim  ellipse ry 0.01*croot rz 0.01*croot
attribute capsGroup $Fuse
translate tlen+0.75*croot 0.0 toff
```

0.7.3 Performing analysis using pyCAPS

An example pyCAPS script that uses the above *.csm file to run FRICTION is as follows.

First the pyCAPS and os module needs to be imported.

```
# Import pyCAPS module
import pyCAPS

# Import os module
import os
import argparse
```

Local variables used throughout the script are defined.

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

Once the modules have been loaded the problem needs to be initiated with the *.csm file, and design parameter is changed - area in the geometry. Any despmtr from the frictionWingTailFuselage.csm file is available inside the pyCAPS script. They are: thick, camber, area, aspect, taper, sweep, washout, dihedral...

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

capsProblem.geometry.despmtr.area = 10.0
```

The FRICTION AIM is then loaded with:

```
friction = capsProblem.analysis.create( aim = "frictionAIM", unitSystem = "US" )
```

After the AIM is loaded, the Mach number and Altitude are set (see AIM Inputs for additional inputs). The FRICTION AIM supports variable length inputs. For example 1 or 10 or more, Mach and Altitude pairs can be entered. The example below shows two inputs. Note that the length of the Mach and Altitude inputs must be the same.

```
friction.input.Mach = [0.5, 1.5]

# Note: friction wants ft (defined in the AIM) - Automatic unit conversion to ft
```

```
friction.input.Altitude = [9000, 18200.0]*pyCAPS.Unit("m")

# Specify transition location applied to all surfaces
friction.input.BL_Transition = 0.1

# Specify lifting surfaces
friction.input.LiftSurface = {"Wing":{"BL_Transition":0.2},
                              "HTail":{},
                              "VTail":{}}

# Specify body of revolution surfaces
friction.input.RevolveSurface = {"Fuse":{"BL_Transition":0.05}}
```

Once all the inputs have been set, outputs can be directly requested. The FRICTION analysis will be automatically executed just-in-time (aimExecuteFRICTION).

Any of the AIM's output variables (AIM Outputs) are readily available; for example,

```
Cdtotal  = friction.output.CDtotal
CdForm   = friction.output.CDform
CdFric   = friction.output.CDfric
```

Printing the above variables results in,

```
Total drag = [0.01321, 0.01227]
Form drag = [0.00331, 0.00308]
Friction drag = [0.0099, 0.00919]
```

Bibliography

[1] W. H. Mason. *FRICTION - Skin Friction and Form Drag Program*, Jan. 2006. Available from http://www.dept.aoe.vt.edu/ mason/Mason_f/MRsoft.html. 1

Index