

HI-Mach Analysis Interface Module (AIM) Manual

Marshall Galbraith MIT

January 15, 2026

0.1 Introduction	1
0.1.1 FlightStream AIM Overview	1
0.2 Attribution	1
0.3 FlightStream Data Transfer	1
0.3.1 Data transfer to FlightStream (FieldIn)	1
0.3.2 Data transfer from FlightStream (FieldOut)	1
0.4 Geometry Representation The geometric	2
0.5 AIM Inputs	2
0.6 aimInputsCART3D	3
0.7 AIM Outputs	3
0.8 AIM Execution	4
Bibliography	5
Index	7

0.1 Introduction

0.1.1 FlightStream AIM Overview

A module in the Computational Aircraft Prototype Syntheses (CAPS) has been developed to interact (primarily through input files) with Research in Flight's FlightStream [1]. FlightStream predicts the aerodynamic performance of a vehicle via a panel method, which makes it very fast.

An outline of the AIM's inputs and outputs are provided in [AIM Inputs](#) and [AIM Outputs](#), respectively.

Geometric attributes recognized by the AIM are provided in [Attribution](#).

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

0.2 Attribution

The following list of attributes drives the FlightStream geometric definition.

- **capsLength** This attribute defines the length units that the *.csm file is generated in and is not optional for FlightStream. The FlightStream input grid will be scaled to either the default length of METER or the user specified length unit (see aimUnitsHIMACH).
- **capsReferenceChord** and **capsReferenceSpan** [Optional] These attributes may exist on any *Body*. Their value will be used as the reference moment lengths in FlightStream's input file with their units assumed to be consistent with the attribute "capsLength". These values may be alternatively set through an input value, "ReferenceChord" (see [AIM Inputs](#))
- **capsReferenceArea** [Optional] This attribute may exist on any *Body*. Its value will be used as the reference area in FlightStream's input file with its units assumed to be consistent with the attribute "capsLength". This value may be alternatively set through an input value, "ReferenceArea" (see [AIM Inputs](#))
- **capsReferenceX**, **capsReferenceY**, and **capsReferenceZ** [Optional] These attribute may exist on any *Body*. Their value will be used as the reference moment lengths in FlightStream's input file with their units assumed to be consistent with the attribute "capsLength". These values may be alternatively set through an input value, "ReferenceX" (see [AIM Inputs](#))

0.3 FlightStream Data Transfer

0.3.1 Data transfer to FlightStream (FieldIn)

- **"Displacement"**
Retrieves nodal displacements (as from a structural solver) and updates FlightStream surface mesh.

The FlightStream 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.

0.3.2 Data transfer from FlightStream (FieldOut)

- **"Pressure"**
Loads the pressure distribution from FlightStream vtk file. This distribution may be scaled based on Pressure = Pressure_Scale_Factor*Pressure, where "Pressure_Scale_Factor" is an AIM input ([AIM Inputs](#))

0.4 Geometry Representation The geometric

representation for the FlightStream AIM requires that the body be either a solid body (SOLIDBODY) or a manifold sheet body (SHEETBODY).

0.5 AIM Inputs

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

- **ProjName = "himach_CAPS"**
Name for files generated by HI-Mach AIM.
- **HIMach = "himach.exe"**
The name of the HI-Mach executable. May include full path.
- **Mach = NULL**
Mach number
- **Alpha = 0.0 (default)**
Angle of attack [degree]
- **Beta = 0.0 (default)**
Sideslip angle
- **gamma = 1.4 (default)**
Specific heat ratio
- **Nose_Axis = "x-" (default)**
Geometric nose axis direction. Must be one of: "x-", "x+", "y-", "y+", "z-", "z+"
- **Pitch_Axis = "z-" (default)**
Geometric pitch axis direction. Must be one of: "x-", "x+", "y-", "y+", "z-", "z+"
- **ReferenceChord = NULL**
This sets the reference chord for used in force and moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceChord" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).
- **ReferenceSpan = NULL**
This sets the reference span for used in force and moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceSpan" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).
- **ReferenceArea = NULL**
This sets the reference area for used in force and moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceArea" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).
- **ReferenceX = NULL**
This sets the reference X for moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceX" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).
- **ReferenceY = NULL**
This sets the reference Y for moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceY" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).

- **ReferenceX = NULL**

This sets the reference Z for moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceZ" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).

- **ReferenceVelocity = NULL**

This sets the reference velocity /

```
} else if (index == inPressure_Scale_Factor) { ainame = EG_strdup("Pressure_Scale_Factor"); inval->type = Double; inval->vals.real = 1.0;
```

```
/*!
```

- **Windward_Method = "modified-newtonian" (default)**

Solver windward method

- **Windward_Method = NULL**

Solver leeward method

- **Base_Pressure = NULL**

Solver base pressure method

- **Shielding_Effects = false (default)**

Enable shielding effects

- **Design_Variable = NULL**

List of AnalysisIn and/or GeometryIn variable names used to compute sensitivities for optimization, see `cfDesignVariable` for additional details.

- **Mesh_Morph = False**

Project previous surface mesh onto new geometry.

- **Surface_Mesh = NULL**

A Surface_Mesh link.

0.6 aimInputsCART3D

- **Pressure_Scale_Factor = 1.0**

Value to scale Pressure data when transferring data. Data is scaled based on $\text{Pressure} = \text{Pressure_Scale_Factor} \times \text{Pressure}$.

0.7 AIM Outputs

The following list outlines the FlightStream outputs available through the AIM interface. All variables currently correspond to values found in the *.plt file

Aerodynamic coefficients:

- **Cx** = X-force coefficient
- **Cx** = X-force coefficient
- **Cx** = X-force coefficient
- **CL** = Lift coefficient
- **CD** = drag coefficient
- **CMx** = X-moment coefficient
- **CMy** = Y-moment coefficient
- **CMz** = Z-moment coefficient

0.8 AIM Execution

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

```
FlightStream script.txt > flightstreamOut.txt
```

and on Windows with the command:

```
FlightStream -hidden -script script.txt > flightstreamOut.txt
```

In both cases the FlightStream executable is assumed to be in the PATH environment variable.

The analysis can 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 FlightStream AIM object. In this mode, caps_execute and Analysis.runAnalysis can be used to run the analysis, or FlightStream can be executed by calling preAnalysis, system, and posAnalysis as demonstrated below with a pyCAPS example:

```
print ("\n\npreAnalysis.....")
flightstream.preAnalysis()

print ("\n\nRunning.....")
flightstream.system("FlightStream.exe -hidden -script script.txt"); # Run via system call in inputs analysis
    directory

print ("\n\npostAnalysis.....")
flightstream.postAnalysis()
```


Bibliography

[1] Altair flightstream. [1](#)

Index

AIM Execution, [4](#)

AIM Inputs, [2](#)

AIM Outputs, [3](#)

aimInputsCART3D, [3](#)

Attribution, [1](#)

FlightStream Data Transfer, [1](#)

Geometry Representation The geometric, [2](#)

Introduction, [1](#)