

# Computational Aircraft Prototype Syntheses



## Training Session 1

### CAPS Overview

ESP v1.18

**Marshall Galbraith**

[galbramc@mit.edu](mailto:galbramc@mit.edu)

Massachusetts Institute of Technology

**Bob Haimes**

[haimes@mit.edu](mailto:haimes@mit.edu)

**John F. Dannenhoffer, III**

[jfdannen@syr.edu](mailto:jfdannen@syr.edu)

Syracuse University

- CAPS and MDAO frameworks
- CAPS Goals
- CAPS Infrastructure
- pyCAPS Interface
- capsViewer
- Analysis tools covered by this training

- Several MDAO frameworks/environments have been developed over the last couple of decades
- These tend to focus on:
  - automating overall analysis process by creating “data flows”
  - between user-supplied analyses
  - scheduling and dispatching of analysis execution
  - generation of suitable candidate designs via DOE,...
  - visualization of design spaces
  - improvements of designs via optimization
  - techniques for assessing and improving the robustness of designs

- “Data” that current MDAO frameworks handle are “point” quantities (possible in “small” arrays)
  - geometric parameters: length, thickness, camber,...
  - operating conditions: speed, load,...
  - performance values: cost, efficiency, range,...
- No current framework handles “field” data directly example associated operations (consistent with the source):
  - copy (same as for “point” data)
  - interpolate/evaluate
  - integrate
  - supply the derivative
- Multi-disciplinary coupling in current frameworks require that user supplies custom pairwise coupling routines

- Augment/fix MDAO frameworks
  - Augment MDA with richer geometric information via OpenCSM
  - Allow interdisciplinary analysis with “field” data transfer
  - Not replacing optimization algorithms
- Provide the tools & techniques for generalizing analysis coupling
  - multidisciplinary coupling: aeroelastic, FSI
  - multi-fidelity coupling: conceptual and preliminary design
- Provide the tools & techniques for rigorously dealing with geometry (single and multi-fidelity) in a design framework / process
  - OpenCSM connects design parameters to geometry
  - CAPS connects geometry to analysis tools
- Input and attribution driven automated (not automatic) meshing

## CAPS API

- The main entry point to CAPS system is the C/C++ API
- Direct interface for MDAO framework or User
  - `pyCAPS`: Python interface to CAPS API
- C-Object based (not object oriented)
- Facilitates modification of Geometry/Analysis parameters
  - Geometry parameters defined with `OpenCSM`
  - Analysis parameters defined by `AIMs`
- Tracks parameter modification and dependencies  
e.g. modification of geometric parameter invalidates analysis outputs

## Analysis Interface Module (AIM)

- Interface between CAPS framework in analysis tools
  - Hides all of the individual analysis details (and peculiarities)
  - Does not make analysis tool a “black box”
- Shared libraries written in C/C++
  - Loaded at runtime as plugins
- Defines analysis input parameters and outputs
  - Inputs include attributed BRep with geometric-based information
- AIMs can be hierarchical
  - Parent analysis objects specified at CAPS analysis load
  - Parent and child AIMs can directly communicate

## User

- Defines “Bounds” on geometry to connect “field” data
- Defines which AIMs instances “field” are coupled
- Defines iteration loop

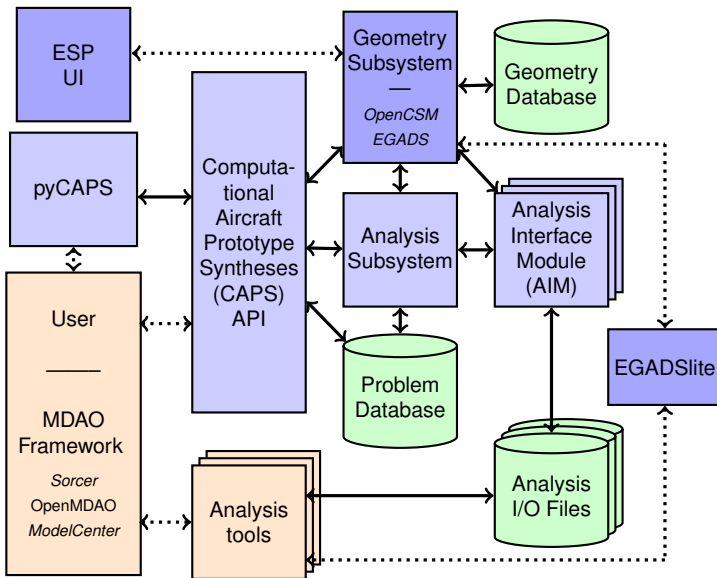
## AIM Developer

- Functions to Interpolate and/or Integrate discrete data (consistent with solver)
- Functions to *reverse* differentiated Interpolate and Integrate to facilitate conservative transfer optimization

## CAPS Framework

- Performs the “field” data transfer (interpolate or conservative)
- Automatically initiated in a *lazy* manner when data transfer is requested





- CAPS API has 6 Object types and 56 functions
- MDAO framework/User manipulate these via CAPS API functions

Object	Description
capsProblem	Top-level <i>container</i> for a single mission/geometry
capsValue	Data <i>container</i> for parameters (scalar/vector/matrix)
capsAnalysis	Instance of an AIM
capsBound	Logical grouping of BRep Objects for data transfer
capsVertexSet	Discrete representation of capsBound
capsDataSet	“Field” data related to a capsVertexSet

- Python interface to CAPS API
- pyCAPS objects  $\approx$  CAPS API objects
  - Nearly 1-to-1 match between interfaces
  - Some aspects “pythonized”
- Training examples for CAPS sessions written with pyCAPS
  - Every example could be written in ANSI C
- Equivalent C/pyCAPS example in session01 directory
  - session01/template\_avl.c
  - session01/template\_avl.py
- Using PreBuild pyCAPS with differing Python versions does not work
- ESP PreBuild comes with Python
  - Matplotlib
  - numpy
  - scipy
- Build from source for other Python installs

- MDAO framework/User has complete control over execution process

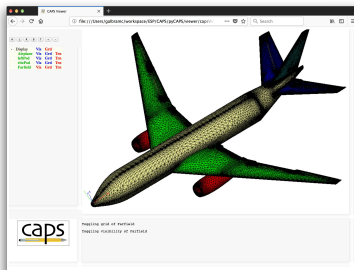
## Simple

- Load Geometry
- Load AIM
- Set Geometry Parameter
- Set Analysis Parameter
- Execute Analysis
- Retrieve Analysis Outputs

## Database Construction

- Load Geometry
- Load AIM
- for\_each Geometry Parameter
  - Set Geometry Parameter
  - for\_each Analysis Parameter
    - Set Analysis Parameter
    - Execute Analysis
    - Retrieve Analysis Outputs

- Recent addition to ESP to assist teaching/debugging with CAPS
- Similar “look and feel” to ESP UI
- Visualize bodies used by CAPS
- Visualize surface meshing AIMs
- Limited capabilities:
  - Only view BODY (no FACE/EDGE/NODE)
  - Cannot change parameters
  - No attribute information
- Visualize data transfer setup in future release



## Low Fidelity

- AWAVE
- Friction
- AVL
- XFoil

## Structural Analysis

- masstran
- mySTRAN
- NASTRAN
- ASTROS
  - linear static & modal analysis
  - support for composites, optimization & aeroelasticity

## 3D CFD

- Cart3D
- Fun3D
- SU<sup>2</sup>

## Meshing

- Surface
  - Native EGADS
  - AFLR4
- Volume
  - TetGen
  - AFLR3
  - Pointwise

- HTML AIM documentation (doxygen)
- Referenced throughout training

\$ESP\_ROOT/doc/CAPSdoc/html/index.html

**CAPS**  
Analysis Interface Module (AIM)

CAPS | AIM | pyCAPS | pyCAPS Examples | Related Pages

## Introduction

### AIM Overview

An Analysis Interface Module (AIM) plug-in is associated with the Computational Aircraft Prototype Syntheses (CAPS, overviewCAPS) portion of Engineering Sketch Pad (ESP).

The type of geometric fidelity expected by the plug-in is specified at dynamic load registration (which is something like: Outer Mode Line, Mid-Surface Aero, Built-up Element Model, Structural Solid Model, etc.). Any inputs (not associated with the BRep) need to be specified at registration. The following functions are a part of any

- Attribute/Input Checking: this AIM function is invoked before any meshInput file generation to ensure that all of the required data can be found.
- Meshing: the input BRep and/or tessellation are used to either perform the meshing directly (if possible or the mesh system has an API) or to provide input to a grid generator. Note that the mesh vertices that sit on geometry (as described in the input BRep) need to be associated back to the geometry. This is important for generating parametric sensitivities and performing conservative data fitting. Most stand-alone grid generation systems maintain this data internally but do not make it available as output. Any attempt to re-associate this data by inverse evaluations is slow and not robust.
- Analysis Input File(s) Generation: the input values and attributes found on the geometry are used to construct and output the input file(s) required to run the analysis.
- Output file parsing: this is required to get performance data, displacements, pressures or other information required to be used as input to another analysis module or to inform the optimizer of the objective functional value(s).
- Conservative Data Transfer Functions: in order to perform the interdisciplinary coupling in a conservative manner, functions that compute interpolation within a surface element, integration of quantities over an element (and their backward or dual variants) are needed.

### Currently Available AIMS

A table of currently available AIMS is outline in the table below.

Surface Meshing	Volume Meshing	Aerodynamics	Structures
EGADS Tess [6]	TetGen [14]	FRICTION [9]	MYSTRAN [3]
AFLR4 [7] [8]	AFLRS [7] [8]	AWAVE [10]	NASTRAN [13]
AFLR2 (2D mesh only)	Pointwise	XFOIL [5]	Astros
Delaundo (2D mesh only)	-	TSFOIL	Masstran
-	-	AVL [4]	-
-	-	CART3D [1]	-

```
$ESP_ROOT/training/CAPS
```

```
|
├── EGADS
├── ESP
├── data
│   ├── session01, session02, ...
├── lectures: session01.pdf, session02.pdf, ...
├── solutions
│   ├── session01, session02, ...
```

- Lecture slides in `lectures` directory
- Lecture slides reference `data` directory  
`session01/template_avl.py` →  
`$ESP_ROOT/training/CAPS/data/session01/template_avl.py`
- Possible exercise solutions in `solutions` directory



1 CAPS Overview	What is CAPS?
2 CAPS Geometry	Interacting with geometry via CAPS
3 CAPS Analysis	Interacting with AIMs
4 Geometry Analysis Views	Geometry for Analysis
5 Aero Modeling	Using multiple AIMs
6 Meshing for CFD I: AFLR	Surface/Volume meshing
7 Meshing for CFD II: Pointwise	Volume meshing
8 CFD Analysis: Fun3D and SU2	CFD execution
9 Meshing for Structures: EGADS	Surface meshing
10 Structures Analysis	Structures attributes
11 Data Transfer: Loosely-Coupled Aeroelasticity	