Computational Aircraft Prototype Syntheses

Training Session 1
CAPS Overview

ESP v1.26

Marshall Galbraith
galbramc@mit.edu

Bob Haimes
haimes@mit.edu

Massachusetts Institute of Technology

John F. Dannenhoffer, III
jfdannen@syr.edu
Syracuse University

Overview

- ESP and CAPS training
- CAPS and MDAO frameworks
- CAPS Goals
- CAPS Infrastructure
- pyCAPS Interface
- ESP UI
- CAPS Analysis Interface Modules
- CAPS documentation
- CAPS training directory structure

CAPS Download

- CAPS is distributed as part of ESP
- ESP is freely available at acdl.mit.edu/ESP
  - macOS 10.5 (and up) downloads see: DownloadsMAC.txt
  - Windows downloads see: DownloadsWIN.txt
- Available as source or PreBuilt binaries
  - acdl.mit.edu/ESP/ESP.tgz (also need OpenCASCADE)
  - acdl.mit.edu/ESP/PreBuilts
- Training found in: acdl.mit.edu/ESP/archive/CAPStraining_Langley_2024.tgz

ESP Training

- CAPS training assumes participants have taken ESP training or are otherwise familiar with the ESP scripting language

CAPS and MDAO frameworks

- 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

CAPS and MDAO frameworks Cont.

- “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:
  - 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

CAPS Goals

- Augment/enhance MDAO frameworks
  - Augment MDA with richer geometric information via `OpenCSM`
  - Enhance automation by tightly coupling analysis with geometry
  - 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
  - Modifying a geometric parameter invalidates analysis outputs

pyCAPS Overview

- Python c-types 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 data/session00 directory
  - data/session00/template_avl.c
  - data/session00/template_avl.py
- pyCAPS works with Python 3.3+
  - serveESP (pyscript) requires Python 3.8+
- PreBuilt ESP includes Python 3.11 (some AIMs embed Python)
  - Includes minimal packages, e.g. Matplotlib
  - Install additional Python packages with pip

Analysis Interface Module (AIM)

- Interface between CAPS framework and 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 inputs/outputs can be linked (create data flow)
  - Transfer simple or rich data (e.g. meshes) between AIMs
- Facilitate field data interpolation (pressure, displacement)

AIMs Shipped with ESP v1.26

Low Fidelity

- AVL
- AWAVE
- CBAero (TPSSizer/FIAT in progress)
- FRICTION
- MSES (sensitivities)
- TSFoil
- XFoil

Structural Analysis

- Abaqus (in progress)
- ASTROS
- Interference
- masstran (sensitivities)
- MYSTRAN
- NASTRAN
- TACS (sensitivities)
- Sierra SD/SM (in progress)

3D CFD

- Cart3D (sensitivities)
- Fun3D (sensitivities)
- SU^2

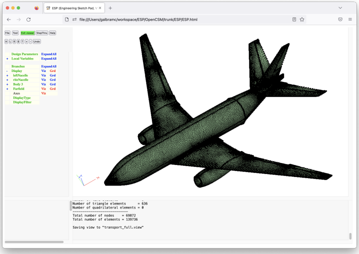
Surface Meshing

- AFLR2
- AFLR4
- Delaundo
- Native EGADS

Volume Meshing

- TetGen
- AFLR3
- Pointwise
- refine (metric adaptation)

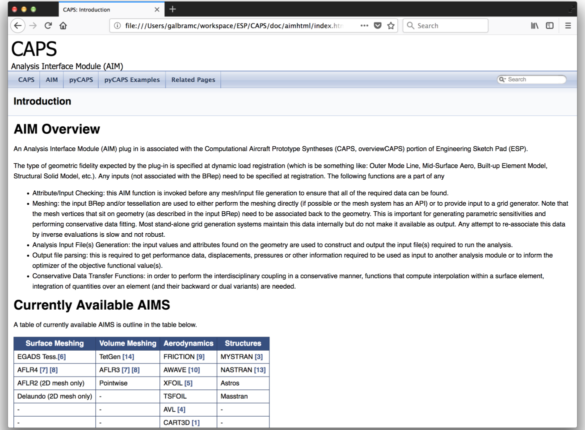
- Used to assist teaching/debugging case setup with CAPS
- Visualize bodies used by CAPS
  - Cannot change parameters or attributes
- Visualize surface meshing AIMs
- Visualize data transfer setup



AIM Documentation

- HTML AIM documentation (doxygen)
- Referenced throughout training

$ESP_ROOT/doc/CAPS/CAPS_Overview.html


CAPS: Introduction

file:///Users/galbramc/workspace/ESP/CAPS/doc/aimhtml/index.htm

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 be 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 mesh/input 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]	AFLR3 [7] [8]	AWAVE [10]	NASTRAN [13]
AFLR2 (2D mesh only)	Pointwise	XFOIL [5]	Astros
Delaundo (2D mesh only)	-	TSFOIL	Masstran
-	-	AVL [4]	-
-	-	CART3D [1]	-

CAPS Training Directory Structure

```
CAPStraining
├── data
│   ├── ESP
│   └── session01, session02,...
├── lectures
│   └── session01.pdf, session02.pdf,...
└── solutions
    └── session01, session02,...
```

- Multi-analysis/fidelity `OpenCSM` files in `ESP` directory
- Lecture slides reference `data` directory
  `session01/1_f118_Geom.py` →
  `CAPStraining/data/session01/1_f118_Geom.py`
- Lecture slides in `lectures` directory
- Possible exercise solutions in `solutions` directory

CAPS Sessions

0	CAPS Overview	What is CAPS?
1	CAPS Geometry	Interacting with geometry
2	CAPS Analysis	Interacting with AIMs
3	CAPS with Optimization	Analysis and Geometric Derivatives
4	Geometry Analysis Views	Geometry for Multi-Analysis
5	Meshing for CFD: AFLR	Surface/Volume meshing
6	CFD: Cart3D, Fun3D, and refine	CFD execution
7	Structures: mAstros, Nastran, and TACS	Structures attributes
8	Data Transfer: Loosely-Coupled Aeroelasticity	