Computational Aircraft Prototype Syntheses

Training Session 1
CAPS Overview

ESP v1.19

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 and ParaView
- CAPS with Pointwise
- CAPS training directory structure
- Muddy cards
- Analysis tools covered by this training

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: `OSXcatalina.txt`
  - Apple M1 MACs see: `AppleM1.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/Training`
  - Required: *overlay* for ESP Rev 1.19 based on specific architecture
  - Follow instructions in `TrainingUpdate.txt`

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

- “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

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
  - Transfer simple or rich data (e.g. meshes) between AIMs

CAPS Infrastructure – Multidisciplinary Coupling

User

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

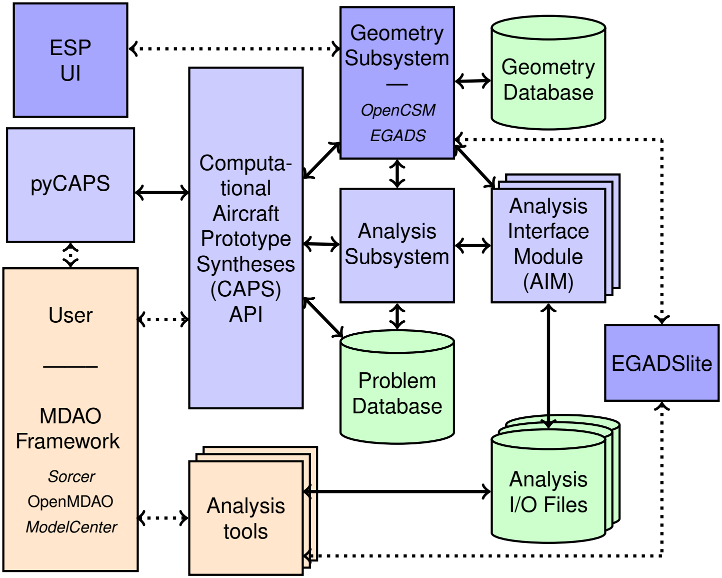
AIM Developer

- 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

ESP with the CAPS Infrastructure



CAPS Infrastructure – Objects

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

Object	Description
capsProblem	Top-level *container* for a single mission/geometry
capsValue	Data *container* 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

pyCAPS Overview

- Python interface to CAPS API
- pyCAPS objects ≈ CAPS API objects
  - Nearly 1-to-1 match between interfaces
  - Some aspects "pythonized"
  - New interface with ESP 1.19 (deprecated interface still functions)
- 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
- pyCAPS works with Python 3.3+
- PreBuilt ESP has Python 3.8 (some AIMs embed Python)
  - Includes minimal packages, e.g. Matplotlib
  - Install additional Python packages with pip

Analysis Scenarios

- MDAO framework/User has complete control over execution process

Simple

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

Database Construction

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

AIMs Discussed in the Training

Low Fidelity

- AWAVE
- FRICTION
- AVL
- XFoil

Structural Analysis

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

3D CFD

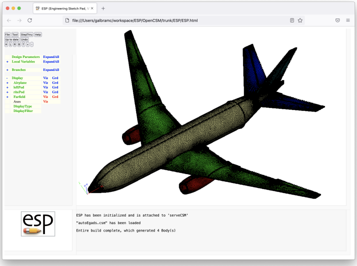
- Cart3D
- Fun3D
- SU^2

Meshing

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

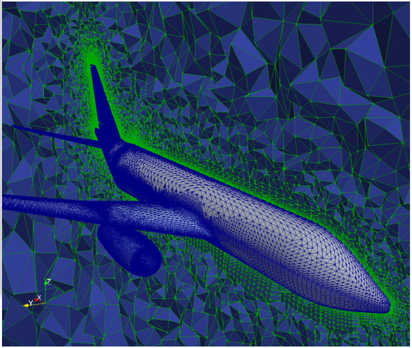
ESP UI With CAPS

- 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 and significant improvements in future release



Volume Mesh Visualization with ParaView

- Volume mesh visualization not supported in `ESP`
- `ParaView` freely available visualizer
  Download at `paraview.org`
- Basic tutorial for mesh visualization: lectures/basic_paraview.pdf



Mesh Generation with Pointwise

Download Pointwise

- `pointwise.com/downloads/pointwise.html`
  Do not need License Manager
- Pointwise training license: CAPS_training_jul2021.lic
- Requires admin to install on macOS and Windows (not on Linux)

- Must tell `ESP` where pointwise is installed with ESPenv
- macOS: ESP119/EngSketchPad/ESPenv.sh
  export PATH=$PATH:/Applications/Pointwise/PointwiseV18.4R4
- Linux: ESP119/EngSketchPad/ESPenv.sh
  export PATH=$PATH:/path/to/PointwiseV18.4R4/
- Windows: ESP119\EngSketchPad\ESPenv.bat
  set PW_HOME="C:\Program Files\Pointwise\PointwiseV18.4R4"
  set PATH="%PW_HOME%\win64\bin";%PATH%

CFD calculations with SU2

Download SU2

- su2code.github.io/download.html
- Available in source or pre-compile binaries
- Must tell ESP where SU2 is installed with ESPenv

Linux and MacOS: ESP119/EngSketchPad/ESPenv.sh

- su2code.github.io/docs_v7/SU2-Linux-MacOS

```
export SU2_RUN=/path/to/SU2/bin
export PATH=$SU2_RUN:$PATH
export PYTHONPATH=$SU2_RUN:$PYTHONPATH
```

Windows: ESP119\EngSketchPad\ESPenv.bat

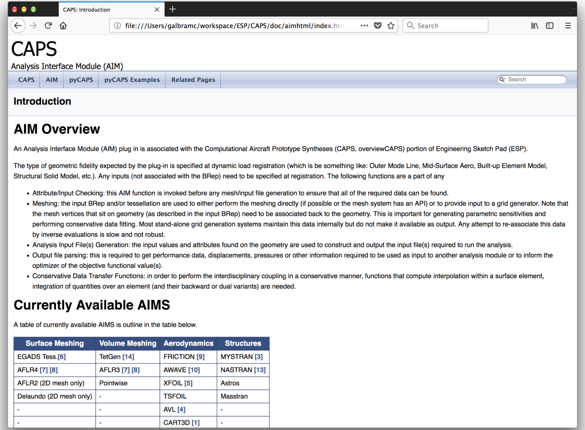
- su2code.github.io/docs_v7/SU2-Windows

```
set SU2_RUN="C:\path\to\SU2\bin"
set PATH=%SU2_RUN%;%PATH%
set PYTHONPATH=%SU2_RUN%;%PYTHONPATH%
```

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

```
$ESP_ROOT/training/CAPS
 ├── EGADS
 ├── ESP
 ├── data
 │    └── session01, session02,...
 ├── lectures:  session01.pdf, session02.pdf,...
 └── solutions
      └── session01, session02,...
```

- Files for meshing in `EGADS` directory
- Multi-analysis/fidelity `OpenCSM` files in `ESP` directory
- Lecture slides reference `data` directory
  session01/template_avl.py →
  $ESP_ROOT/training/CAPS/data/session01/template_avl.py
- Lecture slides in `lectures` directory
- Possible exercise solutions in `solutions` directory

Training Material

Python Language

- Participants are expected to have some programming experience
- All of `CAPS` training uses simple Python scripts
- Limited Python basics will be covered during the `CAPS` training
  - Good resource for more in depth tutorials
    `www.w3schools.com/python`

Relative to 2020 Training

- Training material covers similar topics as the 2020 training
- pyCAPS interface has been refactored as part of EnCAPS

Enhanced CAPS Project

- Follow-on project funded by AFRL to enhance CAPS
  - Breaking changes in C-API (unavoidable)
  - Existing pyCAPS scripts continue to function

Current Status

- Problem directory structure containing all Analysis I/O Files
- Parent/Child replaced with explicit links
- Improved error handling and error messages (in progress)
- Refactor pyCAPS to support coming restart/recycling capabilities

Future tasks

- Restarting the same script recycles previous data
- Deprecate `capsIgnore` in lieu of explicit geometry removal
- Support for analysis execution
- Single UI (and integrated editor) for Geometry and Analysis

CAPS Training Sessions

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	Surface/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	