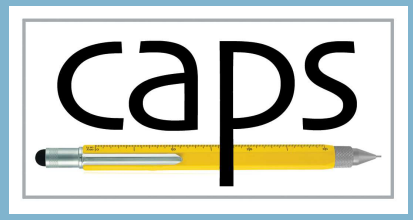
Computational Aircraft Prototype Syntheses



Training Session 1
CAPS Overview

ESP v1.18

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

ESP and CAPS Training

CAPS Download

- CAPS is distributed as part of ESP
- ESP is freely available at acdl.mit.edu/ESP
  - macOS Catalina download instructions OSXcatalina.txt
- Available as source or PreBuilt binaries
  - acdl.mit.edu/ESP/ESP.tgz
  - acdl.mit.edu/ESP/PreBuilts
- Training Patches: acdl.mit.edu/ESP/Training
  - Follow instructions in TrainingUpdate.txt

ESP Training

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

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/fix 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 Infrastructure – API

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 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 can be hierarchical
  - Parent analysis objects specified at CAPS analysis load
  - Parent and child AIMs can directly communicate

CAPS Infrastructure – Multidisciplinary Coupling

User

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

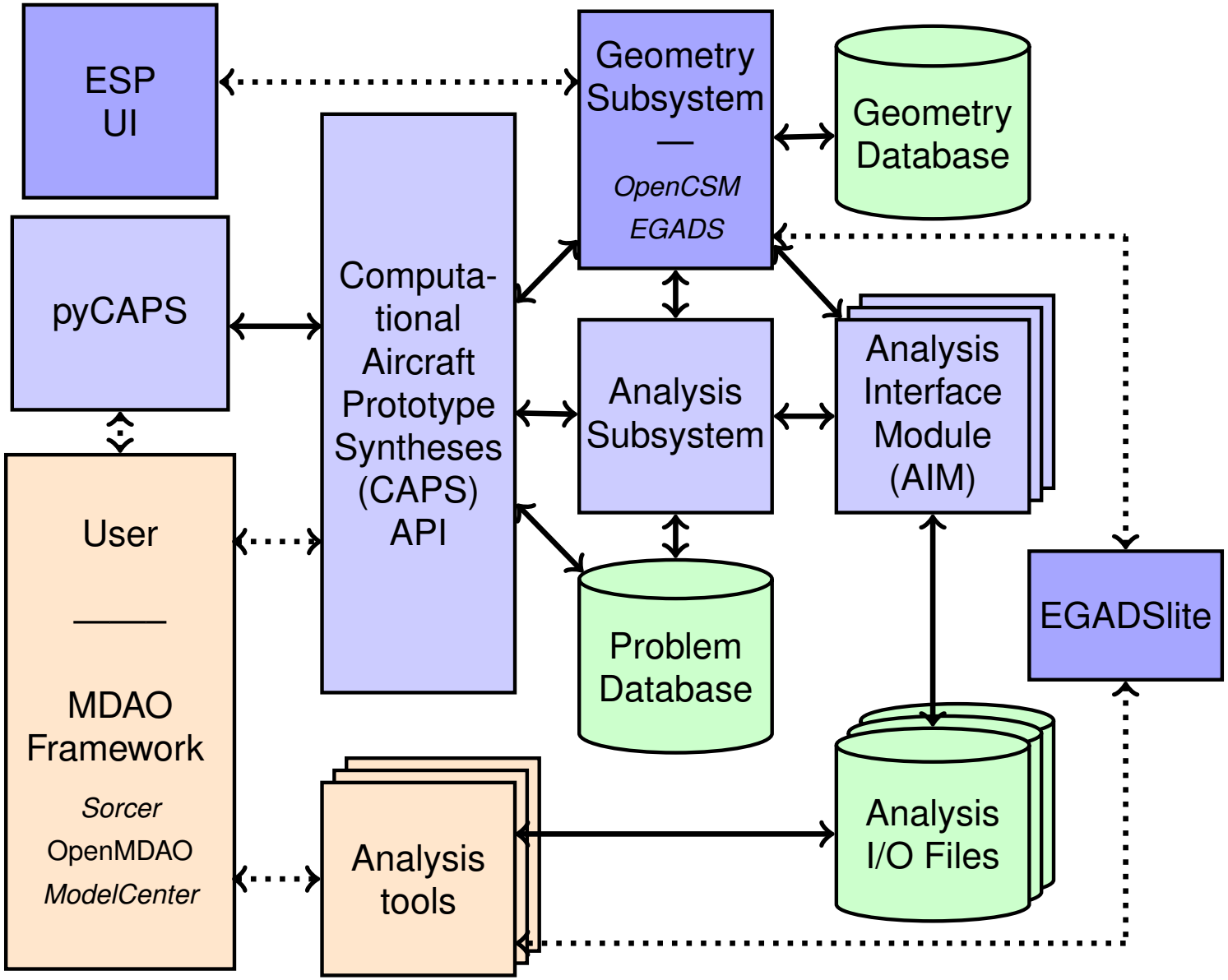
AIM Developer

- 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 when data transfer is requested

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"
- 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
- PreBuild pyCAPS only works with ESP PreBuild Python 3.7
  - Includes minimal packages, e.g. Matplotlib
  - Install additional Python packages with pip
- Build from source is required for other Python installs

Analysis Scenarios

- 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

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

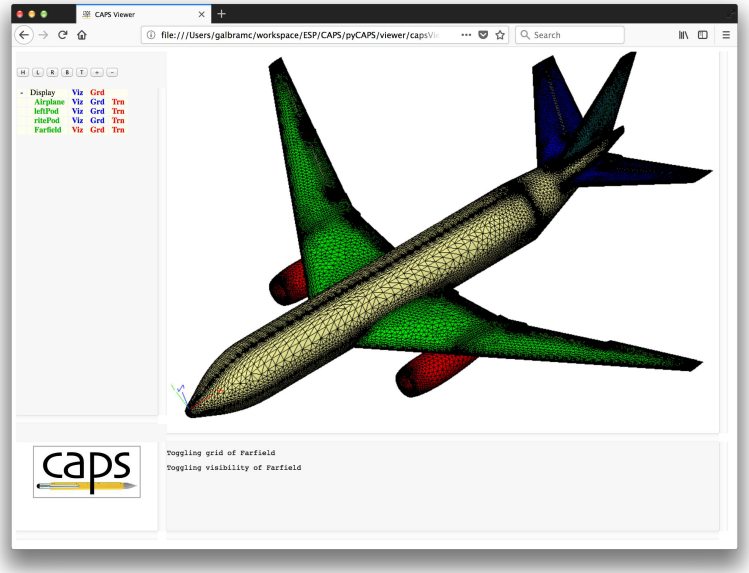
- Cart3D
- Fun3D
- SU^2

Meshing

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

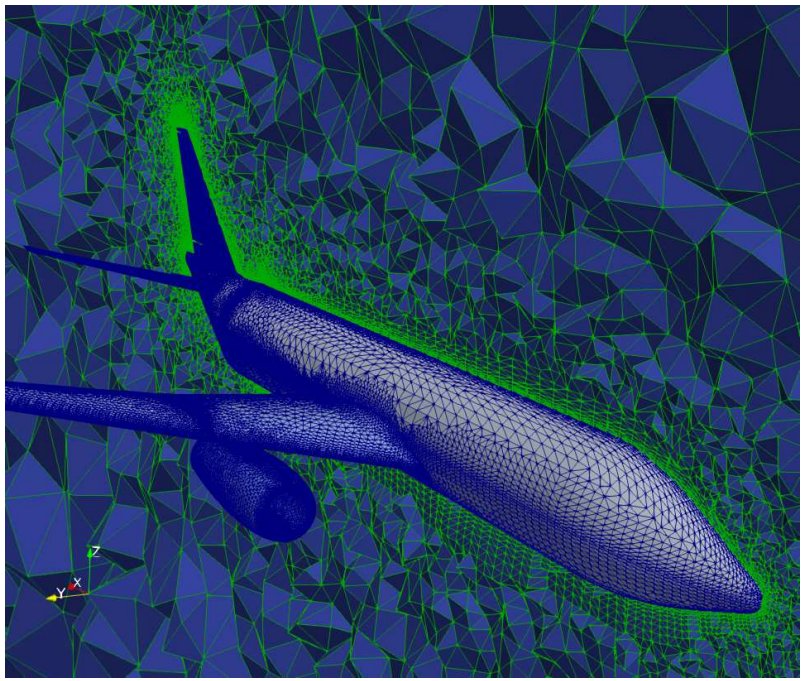
capsViewer

- Used 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 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

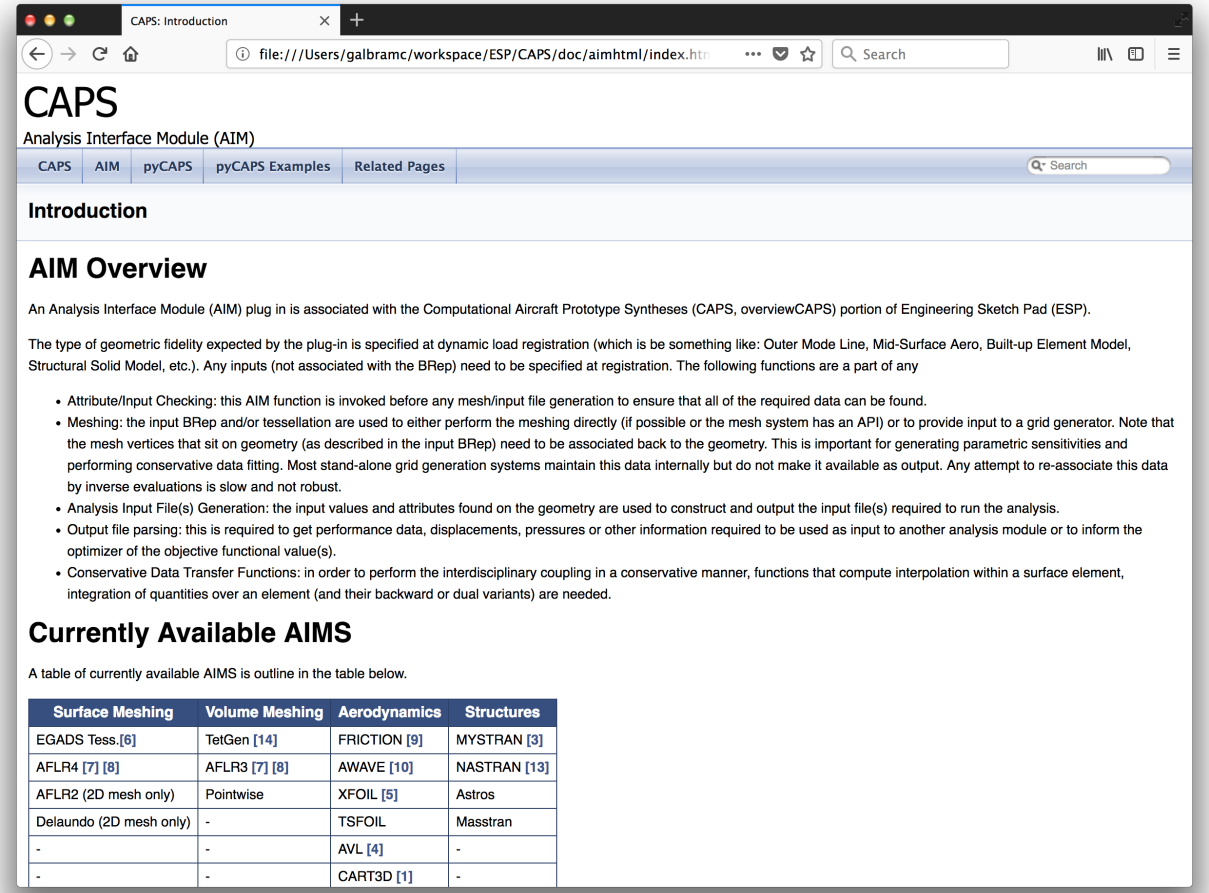
- `pointwise.com/downloads/pointwise.html`
  Do not need License Manager
- Pointwise training license: CAPS_training_sep2020.lic
- Requires admin to install on macOS and Windows (not on Linux)

- Must tell `ESP` where pointwise is installed with ESPenv
- macOS: ESP118/EngSketchPad/ESPenv.sh
  export PATH=$PATH:/Applications/Pointwise/PointwiseV18.3R2
- Linux: ESP118/EngSketchPad/ESPenv.sh
  export PATH=$PATH:/path/to/PointwiseV18.3R2/
- Windows: ESP118\EngSketchPad\ESPenv.bat
  set PW_HOME="C:\Program Files (x86)\Pointwise\PointwiseV18.3R2"

AIM Documentation

- HTML AIM documentation (doxygen)
- Referenced throughout training

$ESP_ROOT/doc/CAPSdoc/html/index.html



CAPS Training Directory Structure

```
$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

Training Material

Python Language

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

Relative to 2019 Training

- Training material cover the same topics as the 2019 training
- Some details have change (e.g. AIM inputs)
- 2020 `CAPS` training includes more exercises

Muddy Cards

- Opportunity to provide anonymous immediate “feedback” for anything not clear (e.g. muddy)
- Any questions about presentation material, critique of sample problems, . . .
- E-mail questions to `galbramc@mit.edu`
- Questions will be answered at next session

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	