Engineering Sketch Pad: An Update on Computational Aircraft Prototype Syntheses

ESP v1.24
https://acdl.mit.edu/ESP

Marshall Galbraith
galbramc@mit.edu
Massachusetts Institute of Technology

AIAA SciTech Forum January 2024

Acknowledgments

- **CAPS started in 2014**
- Primarily funded by AFRL

A Team Effort

- Bob Haimes, Massachusetts Institute of Technology
- John F. Dannenhoffer, III, Syracuse University
- Nitin Bhagat, University of Dayton Research Institute
- David Marcum, Mississippi State
- Ed Alyanak, AFRL
- Dean Bryson, AFRL
- Ryan Durscher, AFRL
- Richard Snyder, AFRL
- and many more...

Outline

CAPS Intrastructure

- CAPS and MDAO frameworks
- CAPS Goals and API
- Analysis Interface Module (AIM)

Analysis and Optimization

- Analysis Scenarios
- Multi-fidelity/-disciplinary

Design *Phases*

- Conceptional/Preliminary Design

Conclusion

Outline

CAPS Intrastructure

- CAPS and MDAO frameworks
- CAPS Goals and API
- Analysis Interface Module (AIM)

Analysis and Optimization

- Analysis Scenarios
- Multi-fidelity/-disciplinary

Design *Phases*

- Conceptional/Preliminary Design

Conclusion

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” in current MDAO frameworks 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 rich geometric information via `OpenCSM`
  - Enhance automation by tightly coupling analysis with geometry (via attribution)
  - 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
- 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
- Geometric and Analysis parametric sensitivities
  - For gradient based optimization

CAPS Infrastructure – API

CAPS API

- The entry point to CAPS system is the C/C++ API
- Direct interface for MDAO framework or User
- 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 API

- c-types interface to CAPS API
- Python script to setup and orchestrate analysis

CAPS Infrastructure – AIMs

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 inputs and outputs (derivatives)
  - 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
- Interpolate "field" data (e.g. pressure/displacement)
  - Loosely coupled Fluid Structure Interaction (FSI)

AIMs Shipped with ESP v1.24

Low Fidelity

- AVL
- AWAVE
- 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-based adaptation)

Outline

CAPS Intrastructure

- CAPS and MDAO frameworks
- CAPS Goals and API
- Analysis Interface Module (AIM)

Analysis and Optimization

- Analysis Scenarios
- Multi-fidelity/-disciplinary

Design *Phases*

- Conceptional/Preliminary Design

Conclusion

Analysis Scenarios

- MDAO framework/User has complete control over execution process
  - C-API or pyCAPS script
  - Many examples in EngSketchPad/CAPSexamples

Simple

- Load Geometry
- Create AIMs
- Set Geometry Parameter
- Set Analysis Parameter
- Generate mesh (automated) (most difficult input to generate)
- Execute Analysis
- Retrieve Analysis Outputs

DOE Database Construction

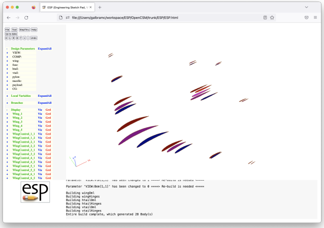
- Load Geometry
- Create AIMs
- for_each Geometry Parameter
  - Set Geometry Parameter
  - Generate mesh (automated)
  - for_each Analysis Parameter
    - Set Analysis Parameter
    - Execute Analysis
    - Retrieve Analysis Outputs

Multi-fidelity/-disciplinary Geometric Analysis Views

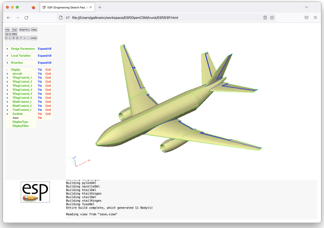
- Single set of geometric parameters → multi-fidelity/-disciplinary analysis
- Views construct analysis specific geometry
- Implemented as user-defined components (UDCs)

		viewVlm.udc	→	avlAIM
transport.csm	→	viewCfdViscous.udc	→	su2AIM
		viewCantilever.udc	→	nastranAIM

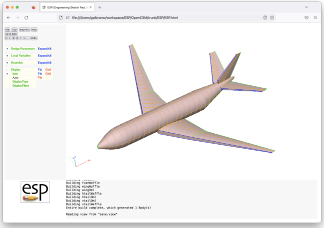
- View attributes geometry with suitable `CAPS` attributes



viewVlm.udc



viewCfdViscous.udc

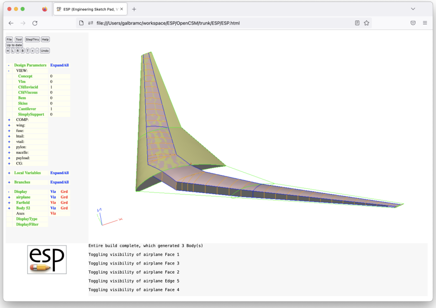


viewCantilever.udc

Coupled Analysis

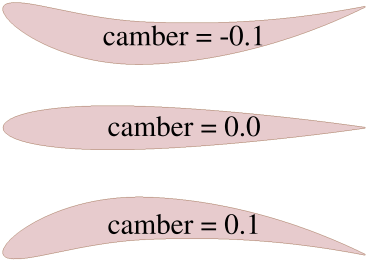
- Coupled analysis (FSI) requires multiple simultaneous analysis geometries
  - Achieved with multiple active views
- `CAPS` facilitates loosely coupled data transfer between AIMs

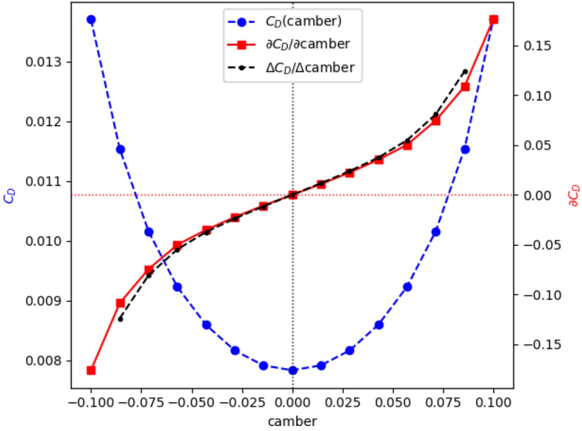
		viewCfdViscous.udc	→	su2AIM
transport.csm	→			↕
		viewCantilever.udc	→	nastranAIM



- Gradient based optimization using OpenMDAO and MSES
  - NACA airfoil, $\alpha = 0°$

$$\min_{\text{camber}} \quad C_D$$





Outline

CAPS Infrastructure

CAPS and MDAO frameworks

CAPS Goals and API

Analysis Interface Module (AIM)

Analysis and Optimization

Analysis Scenarios

Multi-fidelity/-disciplinary

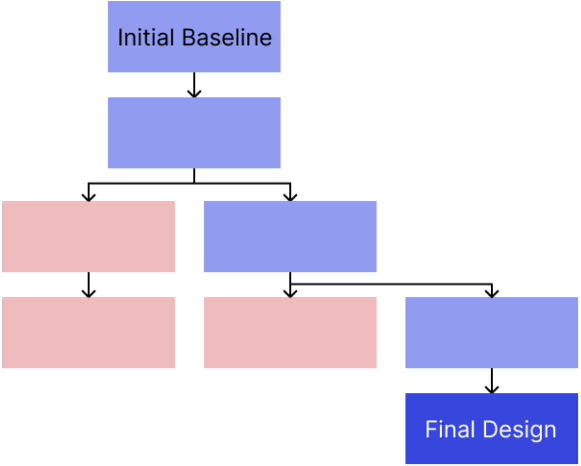
Design *Phases*

Conceptional/Preliminary Design

Conclusion

Phasing Approach to Design[2]

- Each Phase:
  - is a stepping stone to another
  - can branch to multiple new phases
  - can support differing analysis and geometry
- Encapsulated in multiple `pyCAPS` scripts

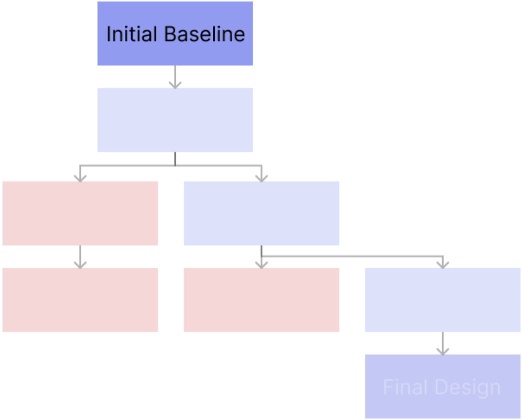


Design as a **growing decision tree**

[1] AIAA 2023-1162

Initial Baseline

- Analytic models
  - $L = \frac{1}{2}\rho V^2 S C_L$
  - $C_D = C_{D_p} + \frac{C_L^2}{\pi A R e}$
  - Beam model
- Surrogate models
  - Data fits (e.g. XFOIL fit C_{D_p})
  - Heuristics (e.g. Tail volume)
  - Empirical models (e.g. Payload fraction)
- Limited Geometry
  - NACA 4-Series airfoil

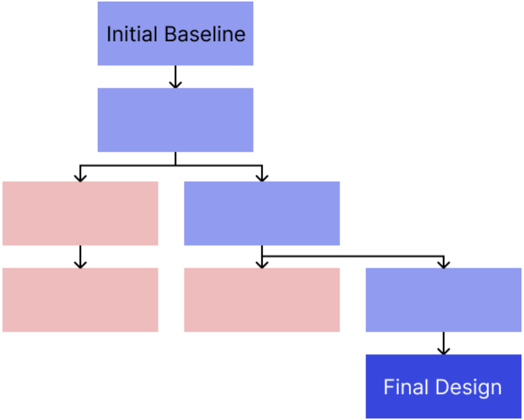


Easy to solve, but **high uncertainty**

Phasing Approach to Design

Increase Fidelity

- Replace surrogate models
  - C_{D_p} data fit → MSES, SU2...
  - Tail volume → AVL
  - Beam model → TACS
- Expand geometry design space
  - Increase number of design parameters
  - NACA → Kulfan family airfoil



Decreasing uncertainty, but increased computational cost

Conclusion

Summary

- ESP/CAPS provides a geometric centric framework for gradient based MDAO
- *Phasing* enables design as a growing decision tree

Current/Future Tasks

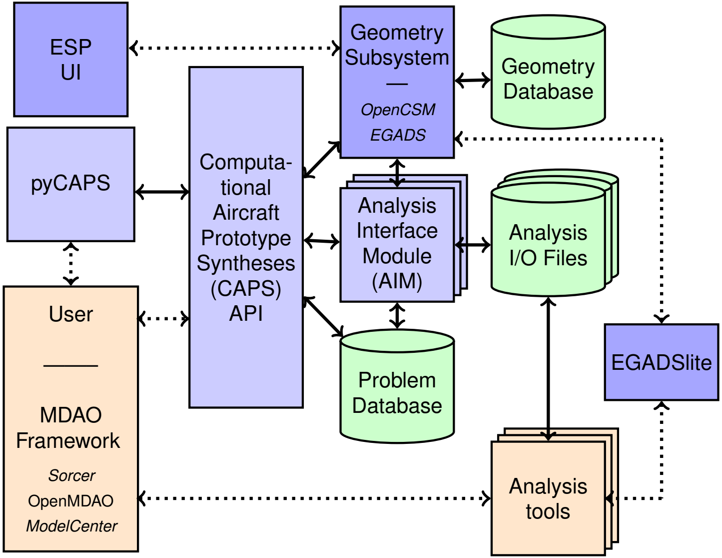
- AIM for FlightStream panel solver by Research In Flight
- Tightly coupled medium fidelity (panel/full potential+shell model+integral boundary layer)
  - Medium fidelity TASOPT
- Metric-based mesh adaptation for structures
- AIMs written in Python

- ESP is freely available for download from acdl.mit.edu/ESP
- Based upon user requests, new and improved features are added continually
- Send bug reports (and success stories) to galbramc@mit.edu or jfdannen@syr.edu

Thank you!
Questions?

This work was funded by the EnCAPS project, AFRL Contract FA8650–20–2–2002: "EnCAPS: Enhanced Computational Aircraft Prototype Syntheses", with Dr. Richard Snyder as the Technical Monitor.

ESP with the CAPS Infrastructure



Plotting and verifying output "camber" derivatives

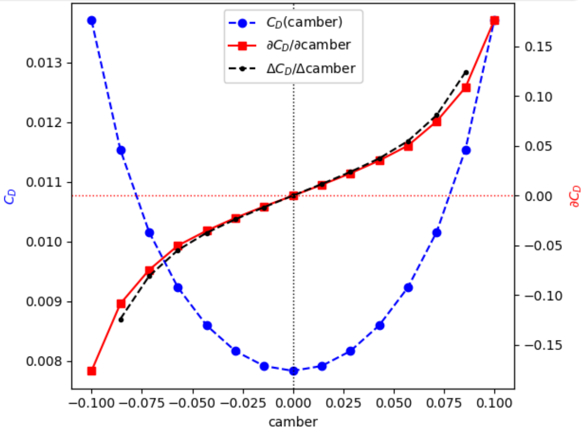
- Analytic vs. Central Difference derivatives
- Plot vs. camber
    - C_d, $\partial C_d/\partial$camber, $\Delta C_d/\Delta$camber

```
# Use camber as the design variable
mses.input.Design_Variable = {"camber":{}}

# Plot the functional and gradient
cambers = npy.linspace(-0.1, 0.1, 15)
CD = []
CD_camber = []
for camber in cambers:
    print("--> camber", camber)
    capsProblem.geometry.despmtr.camber = camber

    CD.append(       mses.output["CD"].value )
    CD_camber.append(mses.output["CD"].deriv("camber") )

# Compute central difference derivatives
dCD_camber = []
for i in range(1,len(cambers)-1):
    dCD_camber.append( \
      (CD[i+1]-CD[i-1])/(cambers[i+1]-cambers[i-1]))
```



CAPS Infrastructure – Objects

- CAPS API has 6 Object types and ∼70 functions
- MDAO framework/User manipulate objects 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

CAPS Infrastructure – Multidisciplinary Coupling

User

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

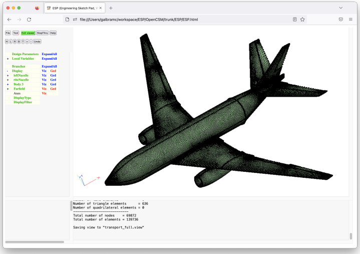
AIM Developer

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

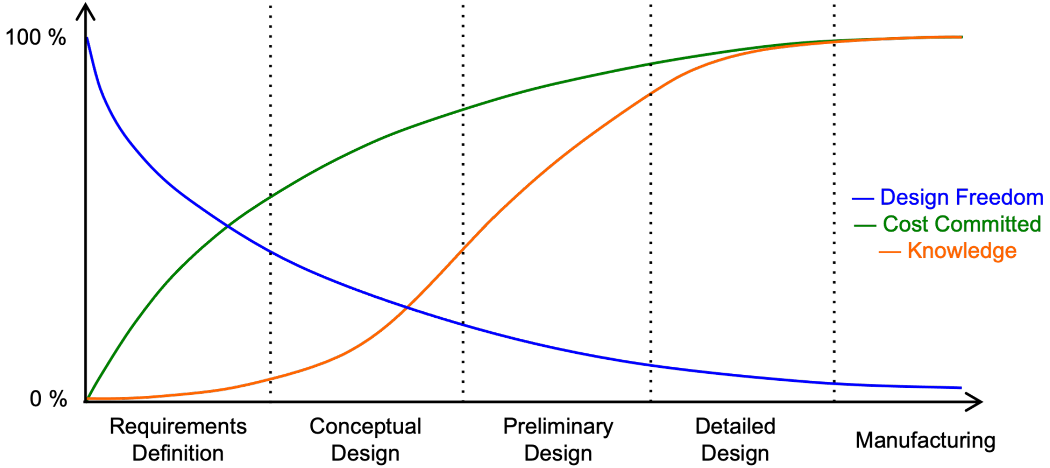
CAPS Framework

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

- Assist teaching/debugging case setup with CAPS
  - Edit/Execute python scripts with `ESP Pyscript`
- Visualize bodies used by CAPS
  - Cannot change parameters or attributes
- Visualize surface meshing AIMs
- Visualize data transfer setup



Traditional Engineering Design[1]



[3]Blanchard and Fabrycky, “Systems Engineering and Analysis”, 1990