pyCAPS - A Python Interface to CAPS

Ryan Durscher
AFRL/RQVC

November 20, 2015

1 Introduction

pyCAPS provides a simple means to interact with Computational Aircraft Prototype Syntheses (CAPS) routines in the Python environment. pyCAPS primarily consists of two files, pyCAPS.c and pyCAPS.py. The first file contains simplified wrapper functions to CAPS where all inputs are typical c-types (int, int *, double, etc.), so the user doesn't have explicit access to the CAPS objects. However, when compiled into a shared object having all the inputs be typical c-types allows for easy integration into Python using the "ctypes" module. The primary propose of the second file, pyCAPS.py, is to further simplify the inputs to pyCAPS.c for the user, i.e. the user doesn't need to worry about using "ctypes" type conversions such c_char_p, c_int, etc. as the native Python types are cast automatically.

- Is this the best approach to integrate CAPS into Python? No, however it is the simplest.
- What would be a better option? Using Cython would be a better option, though looking through the Cython documentation it was not clear how to cythonize a wrapper that had derived types from multiple libraries.
- What are the consequences of using "ctypes"? None that I am currently aware of, but I am sure there are some.

2 Using pyCAPS in Python

In the Python environment pyCAPS is broken down into a series of classes - capsProblem, capsGeometry, and capsAnalysis. As currently designed caps-

Geometry and capsAnalysis are meant to be children of a capsProblem instance.

Substantiations of the capsGeometry and capsAnalysis classes are accessed through the capsProblem.geometry and capsProblem.analysis variables, respectively. Note that capsProblem.analysis is actually a dictionary so multiple AIMs may be loaded for problem.

3 pyCAPs Functions

3.1 capsProblem

The following list outlines the current, primary functions in the pyCAPS capsProblem class.

- __init__(**kwargs)
  - "libDir" = Directory where the pyCAPS.so resides (should be the same as the AIM shared objects).
  - "capsFile" = *.csm, *.caps, or *.egads file to load for the problem.
- loadCAPS(fname = None) - Loads an *.csm, *.caps, or *.egads file.
- closeCAPS() - Close a CAPS problem.
- saveCAPS(fname = None) - Save a CAPS problem (default: fname = CAPS_File.caps).
- loadAIM(**kwargs) - Load an AIM
  - "aim" = Name of requested AIM.
  - "capsFidelity" = Fidelity in which to initialize the AIM with.
  - "analysisDir" = Working directory for the AIM
  - "altName" = Alternative reference name for the AIM. Needed if the same AIM will be loaded multiple times. If not set, "altName" and the name of AIM become one in the same.
  - "parents" = A list of parent AIMs to initilize the AIM with (use the "altName" of the AIMs)

After a call to loadCAPS() the geometry class is now substantiated. Functions in this class can be accessed through capsProblem.geometry.(function) (please see Section 3.2).

Similarly after a call to loadAIM() the analysis class is now substantiated. Functions in this class can be accessed through capsProblem.analysis["AIM Name (altName)"].(function) (please see Section 3.3).

Additional (potentially usefully) class variables that may be set are as follows.

- capsFidelity - Shortcut to set the capsFidelity keyword argument when loading AIMs (i.e. one can just set this once and not have to enter it when using loadAIM() - Note entering a value for the capsFidelity keyword argument in loadAIM() sets this value).
- analysisDir - Shortcut to set the analysisDir keyword argument when loading AIMs (same caveat as capsFidelity).
- capsFile - Set the *.csm, *.caps, or *.egads file that will be loaded.

3.2 capsGeometry

The following list outlines the current, primary functions in the pyCAPS capsGeometry class.

- setGeometryVal(varname = None, value=None) - Set a geometry parameter of name = "varname" and value = "value". Typecasting (Python to C) is handled automatically.

3.3 capsAnalysis

The following list outlines the current, primary functions in the pyCAPS capsAnalysis class.

- setAnalysisVal(varname = None, value=None) - Set a AIM input parameter of name = "varname" and value = "value". Typecasting (Python to C specified type in the AIM) is handled automatically. If the AIM input requires a fixed size/shape (set within the AIM), the size of the value provided to the function is checked for consistency. Furthermore, if the AIM allows the input variable to change size/shape, pyCAPS executes the appropriate CAPS's routines to modify the shape/size based on the function input value. (NOTE: This currently only supports scalars and vectors - NO matrices).
- getAnalysisInfo() - Get analysis information (print to screen).

- aimPreAnalysis() - Run AIM pre-analysis.
- aimPostAnalysis() - Run AIM post-analysis.

4 Example

The following python code uses CAPS to setup an analysis project and run the pre-analysis. (this file is provided with pyCAPS - fun3d_PyTest.py)

```
# Import pyCAPS class file
from pyCAPS import capsProblem as pyCAPS

# Initialize pyCAPS object - directory where the pyCAPS.so resides
pyCAPS = pyCAPS("/home/dursch/Desktop/CAPS/EngSketchPad/lib/")

# Load CSM file
pyCAPS.loadCAPS("../data/CAPS/fun3dAIM.csm")

# Set fidelity - could also be set individually during loadAIM()
pyCAPS.capsFidelity = 20

# Change a design parameter - area in the geometry
pyCAPS.geometry.setGeometryVal("area", 50)

# Load desired aim
pyCAPS.loadAIM(aim = "tetgenAIM", analysisDir= ".")

# Set new tessellation parameters
pyCAPS.analysis["tetgenAIM"].setAnalysisVal("Tess_Params", [0.05, .001,
20.0])

# Set project name
pyCAPS.analysis["tetgenAIM"].setAnalysisVal("Proj_Name", "pyCAPS_FUN3D_Test")

# Get analysis info
pyCAPS.analysis["tetgenAIM"].getAnalysisInfo()

# Run AIM pre-analysis
pyCAPS.analysis["tetgenAIM"].aimPreAnalysis()

# Run AIM post-analysis
pyCAPS.analysis["tetgenAIM"].aimPostAnalysis()
```

```
# Load desired aim - using an alternative name
pyCAPS.loadAIM(aim = "fun3dAIM", altName = "fun3d",
analysisDir = ".", parents = ["tetgenAIM"])

# Set AoA number
pyCAPS.analysis["fun3d"].setAnalysisVal("Alpha", 1.0)

# Set Mach number
pyCAPS.analysis["fun3d"].setAnalysisVal("Mach", 0.4)

# Run AIM pre-analysis
pyCAPS.analysis["fun3d"].aimPreAnalysis()

# Run AIM post-analysis
pyCAPS.analysis["fun3d"].aimPostAnalysis()

# Close CAPS
pyCAPS.closeCAPS()
```

5 Current issues

- When setting an analysis or geometry variables to specified values only scalars and vectors are currently supported, NO matrices