Tutorial 2:
Python Scripts

Dr. Heather L. Kline
National Institute of Aerospace
August 9th, 2019

Agenda

- Get the tutorial files
- Start the simulation (generate a drag polar)
- Introduction to python scripts distributed with SU2
  - Drag polar
  - Shape optimization
- Anatomy of a python script
- Results of the simulation

Acknowledgments
The files for this tutorial are based on a test case for the compute_polar.py script developed by E. Arad

Tutorial Files And Required Settings

- Set PYTHONPATH (if not already done):
  export SU2_RUN=<...../bin/> (path to SU2_CFD, etc)
  export PYTHONPATH=$PYTHONPATH:$SU2_RUN
  - Python scripts require the path in order to find all the functions that are defined in subfolders.
  - Python scripts can now be called from any folder without moving the scripts.
- Get and extract configuration, mesh and solution files:

- Move to the new directory:
  cd WorkshopTutorial2/
  - Similar to files needed for SU2_CFD analysis.
  - Additional ‘ctrl’ file for polar computation definition
- The files for this tutorial are based on a test case for the compute_polar.py script developed by E. Arad.
- Modify to use paraview if needed.

Starting the Simulation

compute_polar.py -c polarCtrl.in -n 2 -i 1000 >& out.txt &

To verify the location of the script:
$ which compute_polar.py

To check the number of available processors:
$ nproc

To follow the output to opt.out:
$ tail -f out.txt

Starting the Simulation

```
compute_polar.py -c polarCtrl.in -n 2 -i 1000 >& out.txt &
```

compute_polar.py: Python script located in SU2/bin/

-c <filename>
Specify the control file

-n <np>
Specifies the number of processors

-i <niter>
Specify the number of iterations for each CFD simulation

>& out.txt &: Redirecting output

To verify the location of the script:
$ which compute_polar.py

To check the number of available processors:
$ nproc

To follow the output to opt.out:
$ tail -f out.txt

More about compute_polar.py

- compute_polar.py -h
- PolarCtrl.in file
- open compute_polar.py in a text editor

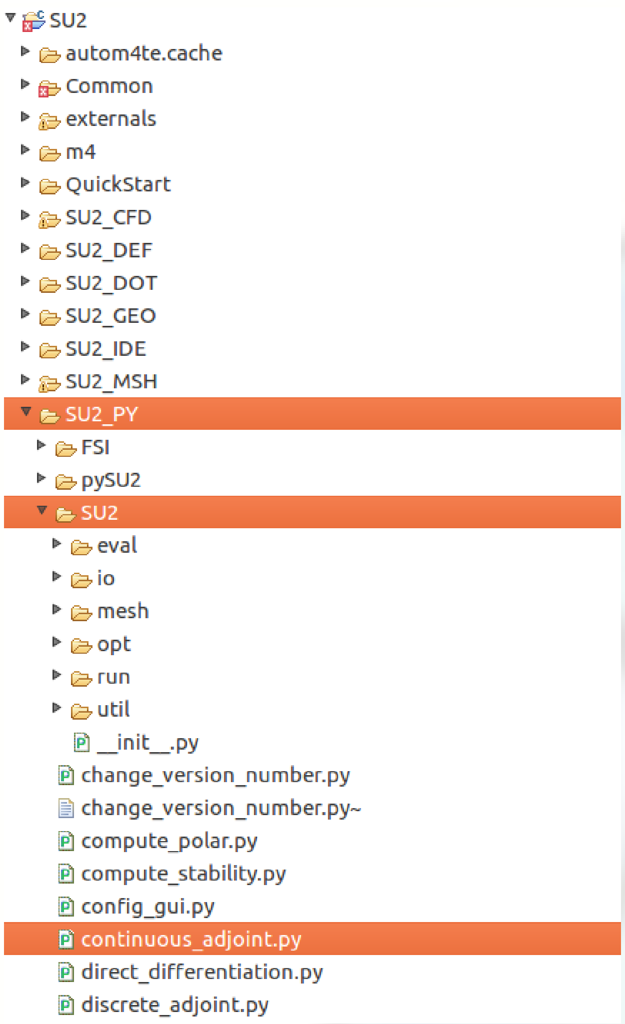
compute_polar.py -h

```
Usage: compute_polar.py [options]

Options:
  -h, --help            show this help message and exit
  -c FILE, --ctrl=FILE  reads polar control parameters from FILE
                        (default:polarCtrl.in)
  -n PARTITIONS, --partitions=PARTITIONS
                        number of PARTITIONS
  -i ITERATIONS, --iterations=ITERATIONS
                        number of ITERATIONS
  -d geomDim, --dimmension=geomDim
                        Geometry dimension (2 or 3)
  -w, --Wind            Wind system (default is body system
  -v, --Verbose         Verbose printout (if activated)
```

Python Scripts

- Source code location: SU2/SU2_PY/
- Installed location: SU2/bin/
- To run a local version: ./python_script.py
- To run version installed in the bin/ directory: python_script.py



```python
#!/usr/bin/env python

## \file Compute_polar.py
#  \brief Python script for performing polar sweep.
#  \author E Arad (based on T. Lukaczyk and  F. Palacios script)
#  \version 6.2.0 "Falcon"
#
```

Starts python environment

...

```python
#
# Several combinations of angles are possible:
#------------------------------------------------
# 1. Polar-sweep in alpha per given phi                        ...... polarVar   = aoa
# 2. Polar-sweep in alpha per given beta (side slip angle)     ...... polarVar   = aoa
# 3. Polar-sweep in phi per given alpha                        ...... polarVar   = phi
# 4. Mach ramp  (single values for alpha, phi or both permitted)  ... polarVar   = MachRampNumbers
#
# Note: Seting a list of both phi and beta is impossible
#       For mach ramp you can specify alpha, phi (or both), but not a list of either of them

# make print(*args) function available in PY2.6+, does'nt work on PY < 2.6
from __future__ import print_function

# imports
import os, sys
from optparse import OptionParser
sys.path.append(os.environ['SU2_RUN'])
import SU2
import SU2.util.polarSweepLib as psl
import copy
import numpy as np

def main():
    # Command Line Options
    parser = OptionParser()
    parser.add_option("-c", "--ctrl", dest="ctrlFile",
                      help="reads polar control parameters from FILE (default:polarCtrl.in) ",
                      metavar="FILE", default="polarCtrl.in")
    parser.add_option("-n", "--partitions", dest="partitions", default=2,
                      help="number of PARTITIONS", metavar="PARTITIONS")
    parser.add_option("-i", "--iterations", dest="iterations", default=-1,
                      help="number of ITERATIONS", metavar="ITERATIONS")
```

Import python packages and functions defined in other files

'import SU2' loads numerous functions defined in SU2_PY/SU2/

Option definitions

...

```python
# load config, start state
config = SU2.io.Config(inputbaseFile)
state = SU2.io.State()
# Set SU2 defaults units, if definitions are not included in the cfg file
if 'SYSTEM_MEASUREMENTS' not in config:
    config.SYSTEM_MEASUREMENTS = 'SI'
if config.PHYSICAL_PROBLEM == 'NAVIER_STOKES':
    if 'REYNOLDS_LENGTH' not in config:
        config.REYNOLDS_LENGTH = 1.0

# prepare config
config.NUMBER_PART = options.partitions
if options.iterations > 0:
    config.EXT_ITER = options.iterations
config.NZONES = 1

# find solution files if they exist
state.find_files(config)

# start results data
results = SU2.util.bunch()
```

Loads config options

State object

Set options different from config file

Required: links the mesh and solution files

Starts an object to designed to store solutions

National Institute of Aerospace

...

```
# local config and state
konfig = copy.deepcopy(config)
# enable restart in polar sweep
konfig.DISCARD_INFILES = 'YES'
ztate = copy.deepcopy(state)
```

Copy the config and state objects

...

```
drag = SU2.eval.func('DRAG', konfig, ztate)
lift = SU2.eval.func('LIFT', konfig, ztate)
```

Runs SU2_CFD

The state object stores whether the solution has already been run:
- Only the first SU2.eval.func… will start a new simulation, subsequent calls will pull from stored data.
- A deepcopy is necessary to avoid pulling results from previous solutions.

...

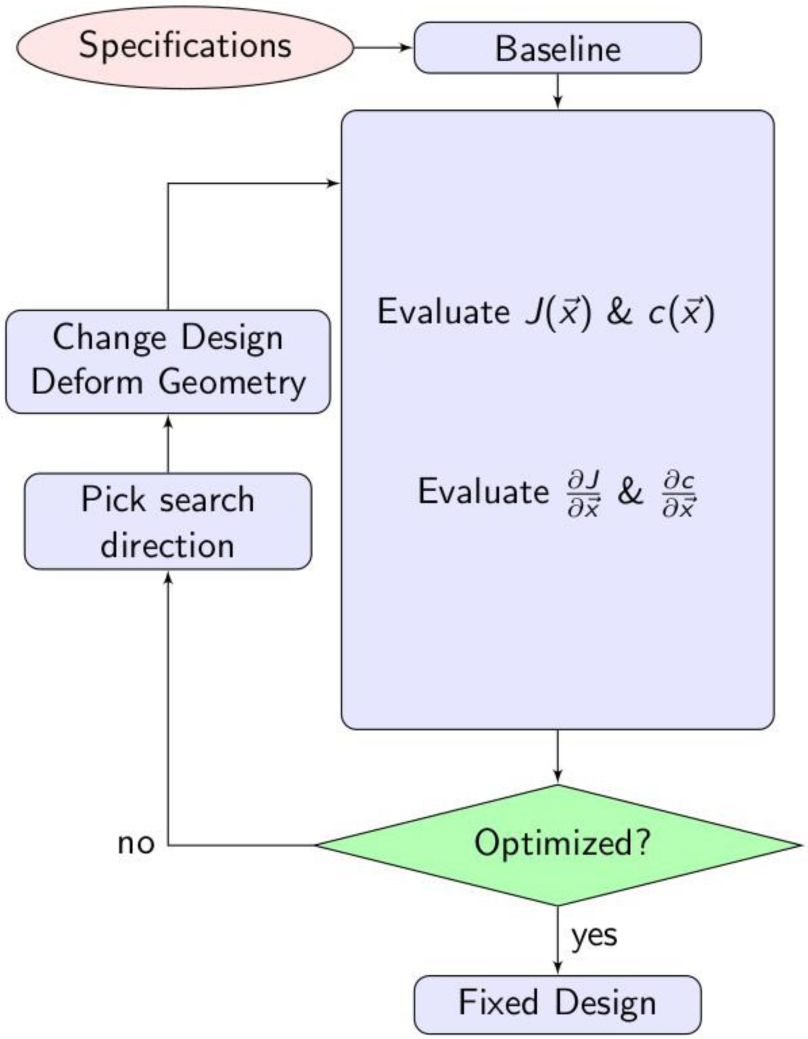
```
if __name__ == "__main__":
    main()
```

Runs function defined in ‘main’ when script is executed at the command line

The Other Python Scripts

- **parallel_computation.py**
  - Most used: runs a parallel SU2_CFD simulation using a specified input file and number of processors.
  - MPI behavior defined in **SU2/run/interface.py**
- **finite_differences.py, continuous_adjoint.py, discrete_adjoint.py**
  - Evaluate gradients using the associated method.
  - Uses the design variables and deformation settings defined in the SU2 config file.
- **set_ffd_design_var.py**
  - Generates FFD box design variable definitions
- **shape_optimization.py**
  - Executes a shape optimization problem defined in a specified SU2 config file, using gradient information with a method specified by script inputs.
  - More on this covered in a later tutorial in this workshop.
  - Next: what is optimization?

Introduction to Optimization



Non-Linear Program:

$$\begin{array}{lll} \text{minimize} & J(\vec{x}) & \\ \text{with respect to} & \vec{x} \in \mathbb{R}^n & \\ \text{subject to} & \hat{c}_j(\vec{x}) = 0, & j = 1, ..., \hat{m} \\ & c_k(\vec{x}) \geq 0, & k = 1, ..., m \end{array}$$

$\vec{x}$: **design variables**, bump functions, FFD control points
J : **objective function**, an evaluation of SU2_CFD
c : **constraints**, an evalution of SU2_CFD or SU2_GEO

Optimization Algorithm: SciPy SLSQP
Gradient Techniques: continuous adjoint, finite difference, discrete adjoint.

Results from compute_polar.py script

- Polar_M0.8.dat
  - Output of AoA, Mach, and aerodynamic coefficients
- DIRECT_… folders

Questions?

Up next: hackathon