MDO assignment; v. 1.0

John T. Hwang, Justin S. Gray, John P. Jasa, and Joaquim R. R. A. Martins

February 20, 2017

1. **Structural optimization.** For this problem, we will use `prob1.py` as our starting point.

   (a) This script performs structural analysis and optimization of a tubular beam clamped in the middle. Run the optimization, first with uniform loading and then again with tip loads applied. What optimized thickness distributions do you see?
   Commands:

      i. run the optimization: `python prob1.py`

      ii. view the results: `python plot_all.py s`

   (b) Run the optimization with tip loads applied for a range of different mesh sizes (num_y). Plot the computation time vs num_y.

   (c) The script produces an html file, `prob1.html`, that can be useful for studying the problem structure. You can open this file in any web-browser. What is the physical interpretation of this problem? That is, what are we minimizing and subject to what constraint?

2. **Multidisciplinary analysis.** We now want to couple aerodynamics and structures together.

   (a) Open `aerostruct.html` to use a guide. Assemble the aerostructural analysis group following the layout presented there. For this problem start with `prob2a.py`.

   (b) Try NLGS and Newton, and Hybrid NLGS/Newton for mesh sizes (9) and (13) and compare run times. Then try to run the problem with the Newton solver in the `root` group instead of the `coupled` group. For this problem work with `prob2b.py`. Why do we put the nonlinear solver on the 'coupled' group, instead of the 'root' group?

   (c) (Bonus) Try LNGS, Krylov, Krylov-PC-GS, and direct linear solvers with the Newton nonlinear solver. Which ones can successfully converge the linear problem? Which one gives the fastest convergence for the Newton solver? Why shouldn't we use the `DirectSolver` with high-fidelity problems?

3. **Multidisciplinary optimization.** Now that you've worked with the aerostructural analysis, you're ready to try aerostructural optimization.

   (a) Compute the derivatives of the multidisciplinary system using finite differences by running `prob3ab.py`. Take note of the run times and derivatives values output by the run script.

   (b) Now compute the same derivatives using the adjoint method and compare the timings for different mesh sizes. Use `prob3ab.py` again, but comment out the settings near the bottom of the file to switch on analytic derivatives.

   (c) We will now perform aerostructural optimization. Edit `prob3c.py` by adding the following design variables:

      - 'twist', lower = -10, upper = 10, scaler = 1000
      - 'alpha', lower = -10, upper = 10, scaler = 1000

- 't', lower = 0.003, upper = 0.025, scaler = 1000

and the follow objective and constraints:

- 'fuelburn'
- 'failure', upper = 0
- 'eq_con', equals = 0

This optimization will take some time, but you can monitor the progress while it runs. Without stopping the optimization, open a second command window and type the command:

`python Optview.py aerostruct.db`.

You can change the settings to adjust what variables you're plotting and you can check the `Automatically refresh` option to have Optview update the plots as new iterations are saved.

You can also open a 3D visualization of your wing by typing the command:

`python plot_all.py as`

(d) Now that you've run the aerostructural optimization case, experiment with the different solver parameters and design variable options to find the optimum in the fewest number of function evaluations. You must keep the same design variables, objective, and constraints as in part c, but you are free to vary the solver setup, parameter scaling, and optimizer settings.

Your final fuelburn measure needs to be below 987511 for the optimization to be considered successful.