MDO assignment; v. 1.1

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

February 21, 2017

For all problems in this assignment, we will use the `problems.py` file. By supplying command-line arguments to the file, you can choose which problem to run. For example, `python problems.py prob1` runs the code for problem 1. Available options are `prob1, prob2, prob3ab, prob3c`.

1. **Structural optimization.**

   (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 for each case?
   Commands:

      i. run the optimization: `python prob1.py`
      ii. view the results: `python plot_all.py s`
      iii. view the optimization history: `python OptView.py s`

   (b) Run the optimization with tip loads applied for a range of different mesh sizes by changing the `num_y` value. 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.** 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 run with `python problems.py prob2a`.

   (b) Now that you have assembled the analysis groups correctly in part (a), we will try different nonlinear solvers. Look at this OpenMDAO documentation page to see what solvers are available and how to use them. Note that we are focusing on nonlinear solvers here and will examine linear solvers in part (c).

   Try NLGS, Newton, and hybrid NLGS/Newton for `num_y = 9` and `num_y = 13` then compare run times with these different solvers. Then try to run the problem with the Newton solver in the `root` group instead of the `coupled` group. Why do we normally put the nonlinear solver on the 'coupled' group instead of the 'root' group?

   (c) Again visit the OpenMDAO documentation page to see what linear solvers are available. Try LNGS, Krylov, Krylov-PC-GS, and direct linear solvers while using the Newton nonlinear solver. Which ones can successfully converge the linear problem? Which one gives the fastest convergence for the Newton solver? Why should we not we use the `DirectSolver` with high-fidelity problems?

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

   (a) Compute the analytic derivatives of the multidisciplinary system by running `python problems.py prob3ab`. Take note of the run times and derivatives values output by the run script.

(b) Compute the same derivatives using finite differences and compare the timings for different `num_y` values. Use the same code, but add a few lines to force the system to use finite-differencing to compute the derivatives. Examine this OpenMDAO documentation page to see how to use finite-differencing.

(c) We will now perform aerostructural optimization. Add the following design variables in the appropriate locations:

- 'twist', lower = -10, upper = 10
- 'alpha', lower = -10, upper = 10
- 'thickness', lower = 0.003, upper = 0.025, scaler = 1000

and the follow objective and two constraints respectively:

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

Now run the code using `python problems.py prob3c`. 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 as`.

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 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. Refer to the previous OpenMDAO documentation pages for more information about possible options.

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