AE588
Assignment 4: Constrained Optimization

Submission Instructions

Please submit both your report (in PDF) and source code to Gradescope. You must make two separate submissions: submit your report to `Assignment 4 (PDF)` and source code to `Assignment 4 (code)`. Your grade will be based on your report, not solely on the autograder's score. Please make sure to include all key steps, results, tables, and figures in your report. The report must be typed. We do not accept handwriting, screenshots of the code outputs, or "see code".

Code submission
Please submit the following Python files without zipping or putting them in a folder.

- `prob4_2.py`: Runscript for Prob. 4.2
- `prob4_3.py`: Runscript for Prob. 4.3
- `prob4_4.py`: Runscript for Prob. 4.4
- All the other Python files you import from the runscripts.

The autograder will execute your scripts by `python prob4_4.py` etc. You can resubmit your scripts as many times as you wish before the deadline.

Code outputs
For each subproblem, please print key results to `stdout` (by simply using the `print` function) in a human-readable format. For example, print x and f values at the converged solution, number of iterations, etc. There is no specific format, but please be concise and turn off debug prints. If a subproblem only requires plotting, you don't need to print anything to `stdout`.

Please note that passing all tests does not imply you will get full credit. The autograder only tests if your code runs without errors; however, it does not check whether the outputs of your code are correct. The outputs will be manually checked by a (human) grader. Also, the autograder does not show figures. If the tests fail in your final submission, points will be deducted.

Autograder Environment
Python 3.10.6, Numpy 1.25.2, Scipy 1.11.2, Sympy 1.12, Jax and Jaxlib 0.4.16, and Matplotlib 3.7.2.

Other packages are not installed by default but can be added upon request. Please email the GSI (shugok@umich.edu) and provide the package name, (its version), and a brief explanation of why you need it. If your script fails with a `ModuleNotFoundError`, it means you're using a package not installed on the autograder.

4.1 (10 pts) In Chapter 2, we mentioned that Euclid showed that among rectangles of a given perimeter, the square has the largest area. Formulate the problem and solve it analytically. What are the units and the physical interpretation of the Lagrange multiplier in this problem? *Exploration*: Show that if you minimize the perimeter with an area constrained to the optimum value you found above, you get the same solution.

4.2 (20 pts) Consider a cantilevered beam with an H-shape cross-section composed of a web and flanges subject to a transverse load as shown in Fig. 1. The objective is to minimize the structural weight by varying the web thickness t_w and the flange thickness t_b, subject to stress constraints. The other cross-sectional parameters are fixed; the web height h is 250 mm and the flange width b is 125 mm.

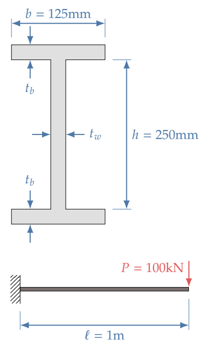

Figure 1: Cantilever beam with H section.

The axial stress in the flange and the shear stress in the web should not exceed the corresponding yield values ($\sigma_{\text{yield}} = 200$ MPa and $\tau_{\text{yield}} = 116$ MPa, respectively). The optimization problem an be stated as,

$$
\begin{aligned}
\text{minimize} \quad & 2bt_b + ht_w && \text{cross-section area} \approx \text{mass} \\
\text{by varying} \quad & t_b, t_w && \text{flange and web thicknesses} \\
\text{subject to} \quad & \frac{P\ell h}{2I} - \sigma_{\text{yield}} \leq 0 && \text{axial stress} \\
& \frac{1.5P}{ht_w} - \tau_{\text{yield}} \leq 0 && \text{shear stress}
\end{aligned}
$$

The second moment of area is

$$
I = \frac{h^3}{12}t_w + \frac{b}{6}t_b^3 + \frac{h^2 b}{2}t_b.
$$

Find the optimal values of t_b and t_w by solving the KKT conditions analytically. Plot the objective contours and constraints to verify your result graphically. Hints:

- You should not use numerical optimization, but you can use a symbolic tool for differentiation and a numerical solver for root finding.
- Manipulate the constraint equations to make them easier to differentiate.

4.3 (20 pts) *Penalty method implementation.* Program two penalty methods (an exterior penalty method and an interior penalty method) from Sec. 5.4. Use the unconstrained optimizer you implemented in the previous assignment.

(a) Solve the inequality-constrained problem in Example 5.4 of the textbook using both of the two methods. Discuss your results: How far can you push the penalty parameter until the optimizer fails? How close can you get to the exact optimum? Is the optimal point strictly feasible? Try different starting points and verify that the algorithms always converge to the same optimum.

(b) Solve the cantilever beam optimization from Problem 4.2 and discuss your results. Plot the optimization search paths on top of the objective contours and constraint boundaries. Did you get exactly the same solution as the analytical solution you derived in 4.2?

4.4 (50 pts) *Constrained optimizer implementation.* Program one of the quasi-Newton SQP or interior point algorithm. Repurpose the quasi-Newton algorithm that you implemented in the previous assignment. For SQP, you only need to implement equality-constrained SQP (you can ignore inequality constraints for this assignment).

(a) Solve the same problem as 4.3 (a). Plot the optimization search path on top of the objective contours and constraint boundaries. For SQP, you can replace the inequality constraint with an equality constraint.

(b) Solve constrained multidimensional Rosenbrock function and discuss your findings. You can make up reasonable constraints (for example, $x_1 + x_2 + x_3 = 0$ for a 4D case) to convert the Rosenbrock problem into a constrained problem.
Solve several problem instances by varying the problem dimensions, number of constraints, starting points, etc. Also, compare the performance (computational cost, accuracy, and robustness) of your optimization implementation to an off-the-shelf software such as Scipy's minimize.