AE588
Assignment 6: Gradient-free 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 6 (PDF)` and source code to `Assignment 6 (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.

- `gradfree.py`: Your gradient-free algorithm implementation
- `prob6_2.py`: Runscript for Prob. 6.2
- `prob6_3.py`: Runscript for Prob. 6.3
- All the other Python files you import from the runscripts.

The autograder will execute your Problem 6.2 runscript by `python prob6_2.py`. It will not run `prob6_3.py` to avoid the timeout, but you still need to submit the file. 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.

Problems

6.1 (20 pts) Implement a gradient-free algorithm of your choice. Briefly explain the algorithm.

6.2 (40 pts) Perform the following studies using the gradient-free algorithm you implemented. The gradient-based optimization can be your own, an existing code, or both.

(a) Optimize the bean function and plot the search history like Examples 7.1 or 7.7 in the textbook.

(b) Add random noise to the function with a magnitude of 10^{-4} using a Gaussian distribution and see if that makes a difference in the convergence of the gradient-free algorithm. Compare the results to those of a gradient-based algorithm. For the gradient-based optimization, make sure that the gradient is consistent with the new noisy function.
(Extra credit) Experiment with different levels of noise, and discuss your results.

(c) Study the effect of adding checkerboard steps (see Example 7.9 in the textbook) with a suitable magnitude to the bean function. How does this affect the performance of the gradient-free and the gradient-based algorithms compared to the smooth case?
(Extra credit) Study the effect of reducing the magnitude of the steps.

(d) Consider the function,

$$f(x_1, x_2, x_3) = |x_1| + 2|x_2| + x_3^2.$$

Minimize this function with your gradient-free algorithm and a gradient-based algorithm. Discuss your results.

6.3 (40 pts) Study the effect of increased problem dimensionality using the n-dimensional Rosenbrock function:

$$f(x) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - {x_i}^2)^2 + (1 - x_i)^2\right)$$

Solve the problem using four different approaches:

(a) Your implementation of a gradient-free algorithm

(b) An off-the-shelf gradient-free optimizer, such as Nelder-Mead or differential evolution of Scipy.

(c) Gradient-based with finite-difference (your own implementation or an off-the-shelf optimizer)

(d) Gradient-based with analytical gradients (your own implementation or an off-the-shelf optimizer)

In each case, repeat the minimization for $n = 2, 4, 8, 16, \ldots$ up to the highest number you can *reasonably* manage. Plot the number of function calls required vs. the number of design variables (n) for all four methods on one figure (see Fig. 7.1 in the textbook). Discuss any differences in the optimal solutions found by the various algorithms and dimensions.