AE588
Assignment 5: Computing Derivatives

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 5 (PDF)` and source code to `Assignment 5 (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.

- `prob5_1.py`: Runscript for Prob. 5.1
- `prob5_3.py`: Runscript for Prob. 5.3
- All the other Python files you import from the runscripts.

The autograder will execute your scripts by `python prob5_3.py` etc. You don't need to submit a script for Prob. 5.2. 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

5.1 (40 pts) Consider the following function from Example 6.3 in the textbook:

$$f(x) = \frac{e^x}{\sqrt{\sin^3 x + \cos^3 x}} \ . \tag{1}$$

Our goal is to compute $\mathrm{d}f/\mathrm{d}x$ at $x = 1.5$ using a finite-difference method, the complex-step method, and your own implementation of AD.

(a) Reproduce the comparison between the finite difference and the complex step from Example 6.4 in the textbook. Reversing the x-axis as we did in the textbook is not necessary.

Discuss your findings:

- Do you get any complex-step derivatives with zero error compared to the analytic reference?
- What is the smallest step you can use before you get underflow?
- What does that mean, and how should you show those points on the plot?
- Estimate the value of h required to eliminate truncation error in derivative using the equations in the textbook.
- Is this estimate consistent with your plot?

(b) Implement a forward-mode-AD tool by creating a new data type and using operator overloading to differentiate this function. Briefly explain your AD implementation, and report the AD derivative value at $x = 1.5$. Verify your AD against the complex-step method.

Hint: You need to define your own data type and provide it to overloaded functions for `exp`, `sin`, `cos`, `sqrt`, addition, division, and exponentiation. You can assume x is a scaler (float). Then, your new data type can be a list (or a numpy array) of length 2.

5.2 (20 pts) Consider Kepler's equation, and suppose that the function of interest is the difference between the eccentric and mean anomalies, i.e.,

$$\begin{aligned} E - e\sin(E) &= M \\ f(E, M) &= E - M \end{aligned} \tag{2}$$

Derive the analytical expression of the total derivative $\mathrm{d}f/\mathrm{d}M$ by applying the unified derivatives equation (UDE) to this problem. Verify your UDE results against a finite-difference method or the complex-step method.

Hints: Sec. 6.9.2 of the textbook. The size of the UDE linear system for this problem will be 3×3. For the finite-difference or complex-step verification, you'll need to use the Newton solver you implemented for Assignment 2.

5.3 (40 pts) Consider the ten-bar truss problem described in Appendix D.2.2 and Problem 5.15 in the textbook. We want to compute (1) the derivatives of the objective (mass) with respect to the design variables (ten cross-sectional areas), and (2) the derivatives of the constraints (stresses in all ten bars) with respect to the design variables (this yields a 10×10 constraint Jacobian matrix). The truss parameters are listed in Table 1.

A template file `truss.py` is provided. In this file, you will need to complete the derivative implementation in the function called `tenbartruss`. The template file is just a starter; please feel free to modify it if necessary.

Compute the derivatives at a design point of your choice using:

(a) A finite-difference formula of your choice.

(b) The complex-step method

(c) The implicit analytic direct method

(d) The implicit analytic adjoint method

(e) (Extra credit) The direct and adjoint methods applied to a KS-aggregated stress constraint.

(f) (Extra credit) Algorithmic differentiation using existing tool/library such as Jax.

Verify that all methods produce nearly identical derivative values. Then, report the errors of each method assuming that the implicit analytic method gives the exact derivatives. Compare the computational cost of each method. Discuss your findings and the relative merits of each approach.

Table 1: Parameters for the 10-bar truss problem.

Parameter	Value	Units
Modulus of elasticity (E)	70×10^9	$\mathrm{N/m^2}$
Material density (ρ)	2720	$\mathrm{Kg/m^3}$
Applied load (P)	5×10^5	N
Length of square sides (L)	10	m
Cross sectional area (A)	5×10^{-5}	$\mathrm{m^2}$
Yield stress (all members except 9)	170×10^6	$\mathrm{N/m^2}$
Yield stress for member 9	520×10^6	$\mathrm{N/m^2}$