GOPH 420/699 – Inv. and Param. Est. for Geophysicists (W2025)

Lab Assignment #4

1 Introduction

In this lab assignment, you will develop Python code to perform linear least squares regression. You will then use your code to analyze an application drawn from environmental monitoring using microseismic data.

2 Prerequisites

The software requirements for this lab are the same as for Labs #0-3:

1. Python 3.8+
2. A package for virtual environments such as `virtualenv` or the built-in `venv`
3. The version control system `git`
4. A GitHub account

You should also have reviewed the course notes on the following topics:

1. Topic #4 – Least Squares Regression

3 Project Setup and Deliverables

You should set up a ***new repository*** for this project using a similar procedure as in Lab #0, and it is recommended to use a ***new virtual environment*** for this project to ensure that you can test it in isolation from other code projects. You will submit both a ***report*** (in **.pdf** format) to the Dropbox on D2L and your ***source code*** (by sharing information on how to clone your `git` repository in the introduction of your report). See the *Lab Report Format* document on D2L for the details of the submission format and review the *Lab Report Rubric* for information on how the report and code will be evaluated. The lab report and code will be due on **Friday, April 11, 2025**.

4 Implementation Details and Problem Description

4.1 Linear Regression Package

Implement a Python package `lab_04` that has a module called `regression` (i.e. in a file `src/lab_04/regression.py`) containing a function that solves for the coefficients in a multi-linear model of the form

$$y_j = f_j\left(\mathbf{z}_j, \mathbf{a}\right) + e_j = a_0 z_{j,0} + a_1 z_{j,1} + \cdots + a_{m-1} z_{j,m-1} + e_j \tag{4.1}$$

where y_j is the dependent variable at data point j, f_j is the model function output at data point j, e_j is the error (or residual) at data point j, $\mathbf{a} = \{a_0, a_1, ..., a_{m-1}\}^T$ is a column vector of the m model parameters, and $\mathbf{z}_j = \{z_{j,0}, z_{j,1}, ..., z_{j,n-1}\}$ is a row vector of the m independent variables at data point j. If there are n data points then equation (4.1) can be written in vector-matrix format as

$$\mathbf{y} = \mathbf{f}\left(\mathbf{z}, \mathbf{a}\right) + \mathbf{e} = \mathbf{Z}\mathbf{a} + \mathbf{e} \tag{4.2}$$

where **y** is the n×1 column vector of dependent variable data, **f** is the n×1 column vector of model output, **e** is the n×1 column vector of residuals, **Z** is the n×m matrix of independent variable data, and **a** is the m×1 column vector of model parameters. The function should have an interface as described by the docstring shown below.

```
def multi_regress(y, Z):
    """Perform multiple linear regression.

    Parameters
    ----------
    y : array_like, shape = (n,) or (n,1)
        The vector of dependent variable data
    Z : array_like, shape = (n,m)
        The matrix of independent variable data

    Returns
    -------
    numpy.ndarray, shape = (m,) or (m,1)
        The vector of model coefficients
    numpy.ndarray, shape = (n,) or (n,1)
        The vector of residuals
    float
        The coefficient of determination, r^2
    """
```

A simple script that uses the function above might look like the following (e.g. in `examples/driver.py` in your repository).

```
from lab_04.regression import multi_regress

def main():
    y = ...  # enter dependent variable data (or load from a file)
    Z = ...  # enter independent variable data (or load from a file)
    a, e, rsq = multi_regress(y, Z)

if __name__ == "__main__":
    main()
```

You will need to extend and modify the driver script above as you respond to the example problem and questions in the next section.

4.2 Example Problem Description

In seismology, the Gutenberg-Richter Law is a model for the relationship between earthquake magnitude and frequency. It is a power law of the form

$$N = 10^{(a-bM)} \tag{4.3}$$

where *N* is the number of earthquake events having magnitude ≥*M*. The parameter *a* (sometimes called the *productivity*) gives how many earthquakes of magnitude *M*≥1.0 generally occur in a region during a given period. The parameter *b* (called simply the *b-value*) controls the rate of decrease of earthquake occurrence as *M* increases. Natural fault systems in seismically active regions generally have *b* ≈ 1.0, but *b* can range from 0.5 to 2.5 depending on the source of the earthquakes. Often, regions experiencing seismicity induced by industrial activities (e.g. hydraulic fracturing, wastewater injection) have *b* ≈ 2.0, though the meaning of this is still an open area of research. It has also been observed that during a swarm of earthquakes of smaller magnitude, *b* can reach as high as 2.5.

a) Linearize the model in equation (4.3) so that is has a form suitable for use with your function for multi-linear regression in Section 4.1.
b) You have been provided with time series data for earthquake events over the course of a week of operations at an industrial site at which fluid is being injected through a well into a shale formation. Plot the data in the time domain for each day, and the *M* vs. *N* data for periods of your choosing. The intervals over which you choose to plot *M* vs. *N* need not be daily, or even regular intervals. It is up to your discretion based on what you observe in the data.
c) For each of the intervals over which you plotted *M* vs. *N*, use your function from Section 4.1 to determine the best fit *a* and *b* parameters. Plot the best-fit curve along with the data.

4.3 Discussion Questions

1) How and why did you select the intervals to plot *M* vs. *N*?
2) Based on the information provided in Section 4.2, and any additional support from the literature, did you notice any significant trends in how the *b* parameter changed over time? Can you make any recommendations on the nature of the earthquakes being generated at this site?
3) Describe any significant or non-trivial components in your algorithm's implementation.