CX 4010 / CSE 6010
Assignment 1
Power Law with Least Squares Fitting

Due Date: 11:59pm on Thursday, August 27
Submit a single zipfile as described herein to Canvas!

Quick note: This assignment may appear intimidating initially because of the length of the exposition and all the mathematics. Don't get too worried; at the end, everything will be summarized in a fairly straightforward way, and the programming involved in the end will not be too complicated.

Power laws occur frequently in nature. Essentially, one quantity varies according to some power of another variable: for example, we can write an output z that varies according to some power of u as $z = cu^a$, where a is the power that describes the nature of the relationship and c is a coefficient. We are all familiar with *linear* relationships, in which the power $a = 1$. For example, when walking at a constant rate, the walking twice as far will require walking for a time that is twice as long. Any other exponent describes a *nonlinear* relationship. We are also familiar with some other exponents. In physics, for example, the kinetic energy of an object depends on the square of its velocity, which is described by $a = 2$. Other exponents describe other types of relationships. For example, $a = 0.5$ would indicate a square-root relationship. Non-integer relationships also occur frequently. By using the full range of real numbers, it is possible to describe how different quantities are related in many applications, such as animal mass and metabolism, cache size and number of cache misses, speed of an object and air resistance, and animal size and lifespan.

A mathematical convenience associated with a power law is revealed when we take logs. (Logs with any base will work, but we assume here that the natural log is used.)

$$\log z = \log(cu^a) = \log c + \log u^a = \log c + a \log u$$

Thus, if two quantities are related by a power law, we find that there is a ***linear relation*** between the ***logs*** of those quantities. Furthermore, the slope of the line is exactly the exponent of the power law! So, we can find the two parameters of a power law by finding the best straight-line fit of the data. The exponent in the power law is the slope of the line and the coefficient in the power law is found by taking e to the line's y-intercept (that is, the y-intercept is the log of the coefficient in the power law).

In this assignment, you will be performing a linear fit, so we will next talk about how to do this generally. To find the best straight-line fit of two corresponding data sets given by pairs (x_j, y_j) for $0 \le j \le n-1$ for some n, we can proceed as follows. We wish to solve an overdetermined set of n equations $k_0 + k_1 x_j = y_j$, where $k = [k_0, k_1]^T$ is the vector of coefficients we seek to find. To see how we can write this in matrix form, let's first write out a few of these.

$$k_0 + k_1 x_0 = y_0$$

$$k_0 + k_1 x_1 = y_1$$

…

$$k_0 + k_1 x_{n-1} = y_{n-1}$$

We can write these n equations in matrix form as

$$Ak = \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ \vdots & \vdots \\ 1 & x_{n-1} \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} = y$$

To solve for $k = [k_0, k_1]^T$, we will use an approach called least squares. We will not fully derive the method, but the idea is that because A is not square (here, it is $n \times 2$), we cannot invert it, so we seek a way to obtain a square matrix that we can invert. What we will do is multiply both sides by A^T. Thus we wish to solve

$$A^T Ak = A^T y.$$

Because A is an $n \times 2$ matrix, $A^T A$ is just a 2×2 matrix. $A^T y$ is a 2×1 vector. Provided we know the entries of A and y, this is an easy system to solve, as we simply use the inverse. For convenience, let's define $M = A^T A$ and $b = A^T y$. Now we can work out the entries of M and b, which turns out to be an easy task given the simple form of A.

First, let's work out the entries of $M = A^T A$. Note that one column of A is all 1's and the other is just the entries of x. Essentially we are calculating the following:

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_{n-1} \end{bmatrix} \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ \vdots & \vdots \\ 1 & x_{n-1} \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n-1} 1 & \sum_{i=0}^{n-1} x_i \\ \sum_{i=0}^{n-1} x_i & \sum_{i=0}^{n-1} x_i^2 \end{bmatrix}.$$

Similarly, it is straightforward to calculate the entries of $b = A^T y$.

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_{n-1} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n-1} y_i \\ \sum_{i=0}^{n-1} x_i y_i \end{bmatrix}$$

Finally, now knowing the entries of M and b, we solve the system $Mk = b$ for k.

$$k = M^{-1} b$$

We can write the inverse of a 2×2 matrix $M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$ analytically:

$$M^{-1} = \frac{1}{m_{11} m_{22} - m_{12} m_{21}} \begin{bmatrix} m_{22} & -m_{12} \\ -m_{21} & m_{11} \end{bmatrix}.$$

So, in the end, we use the inverse to solve for

$$k = M^{-1} b.$$

Your assignment

Write a C program that uses least squares to fit the two parameters of a power law for an interesting data set: the maximum speed of a flying arthropod as a power of its mass. The included paper by Hirt et al.[1] includes a wealth of data on animal speeds and masses in the supplement (appearing after the paper). The data you are to fit are the 19 data points related to arthropod flying; these are the first 19 entries in the table beginning on p. 4 of the supplement. You do not need to worry about any other details beyond relating body mass (kg) and max. speed (km/h) for these 19 rows.

Your program should find and display on the screen (with proper labeling) the values of the exponent a and coefficient c in the power law you find to describe maximum flying speed of arthropods as a function of body mass. You can use a statement like the following to display float or double precision variables `exponent` and `coefficient`. Depending on your editor, you may need to erase and re-type the quotes if you choose to copy and paste.

```
printf"The power law exponent is %.3f and the coefficient is %.3f\n",
exponent, coefficient);
```

Although we will cover arrays in C in more detail, it is expected that you have seen arrays in some programming language as part of your previous programming experience. For simple uses of arrays in C, such as what you likely will use in this assignment, you should need to know only three main points.

1. The simplest way to declare an array is to declare it as with any variable but add the number of elements in the array in square brackets after the variable name.
   `double Mass[15];`
   The size can be a literal, as here, or may be specified using `#define`.
2. To access an entry in an array, simply use square brackets, as in the example below.
   `Mass[2] = 2*Mass[2];`
   Of course the index can be a variable, such as a loop index.
3. Arrays in C are indexed beginning with 0, not with 1. So an array with `N` elements is indexed from `0` to `N-1`.

Your code must be well structured and documented to receive full credit. Be sure to include comments that explain your code statements and structure.

You should submit to Canvas a single zipfile that is named according to your Georgia Tech login—the part that precedes @gatech.edu in your GT email address. For example, I would name my zipfile echerry30.zip.

The zipfile should include the following files:

(1) your code, named powerlaw.c.

(2) a README text file (not formatted in a word processor, for example) that includes the compiler and operating system you used for compiling and running your code along with instructions on how to compile and run your program.

(3) a series of three slides composed in PowerPoint or similar software, saved either in PowerPoint or as a PDF and named slides.pptx or slides.pdf, structured as follows:

- Slide 1: your name and a brief explanation of how you developed/structured your program. This should not be a recitation of material included in this assignment document but should focus on the main structural and functional elements of your program (e.g., the purpose of any loops you used, the purpose of any if statements you used to change the flow of the program, the purpose of any functions you created, etc.). In other words, under the assumption that the mathematics behind what you are doing is already known, what were the main things you did to translate those requirements into code? You are limited to one slide.
- Slide 2: a brief statement explaining why you believe your program works correctly. For example, you could use a plot or table of the original data values and your power-law fit to the same body mass values (or range of body mass values). You are again limited to one slide.
- Slide 3: a brief exposition of what you felt were the most challenging one or two aspects of this assignment—also limited to one slide.

General steps you need to take:

- Code the data (mass and maximum speed pairs) into the program. **Do not use a file** to read in the data for this assignment.
- For purposes of fitting, convert (mass and maximum speed) by taking their logs and thus defining $x = \log(mass)$ and $y = \log(maxSpeed)$. (You do not have to name the variables in your program like this.) Note that to calculate logs (and later exp) you likely will need to include math.h at the top of your program:
  `#include <math.h>`
- Calculate the matrix entries needed for solving $Mk = b$: $m_{11} = \sum_{i=0}^{n-1} 1$, $m_{12} = m_{21} = \sum_{i=0}^{n-1} x_i$, $m_{22} = \sum_{i=0}^{n-1} x_i^2$, $b_1 = \sum_{i=0}^{n-1} y_i$, and $b_2 = \sum_{i=0}^{n-1} x_i y_i$. Note that you are not expected or required to use a matrix type of representation; for a small matrix like this, it may be convenient simply to calculate and store these entries.
- Use the matrix entries to calculate the entries of $M^{-1} = \frac{1}{m_{11}m_{22} - m_{12}m_{21}} \begin{bmatrix} m_{22} & -m_{12} \\ -m_{21} & m_{11} \end{bmatrix}$. Again, there is no need to use a matrix representation.
- Solve $k = M^{-1}b$ using matrix multiplication (again, it is not necessary for you to set up general matrix multiplication to do this; you can feel free just to solve for each of the two components of k using the matrix entries).
- Convert the values of the vector k into the coefficient and exponent of the original power law form and display them on the screen with proper labeling. You can use `exp(a)` to calculate e^a. **Do not use a file** to write out the results for this assignment.

Some hints:

- Start early, and start small! Since this is your first assignment, try implementing a "hello world" code (or something similar) in C first to make sure you can compile and run code with expected results.
- You may want to try this procedure first using data you generate from your own power law. For example, you may choose something like $z = 0.1u^2$ and generate a series of (u, z) pairs as data, rather than reading in the data from the paper. This way you can easily confirm that your program returns the correct coefficient 0.1 and exponent 2 of the power law.
- Another way you can verify your results is to plot the output. For example, using your favorite plotting program, you can plot the pairs (x_i, y_i) from your data points and superimpose the power law you found ($z = cu^a$) evaluated for a series of u values. Or you could plot the pairs $(\log x_i\,, \log y_i)$ together with the log version of your power law.

Reference:

1. Hirt MR, Jetz W, Rall BC, Brose U. A general scaling law reveals why the largest animals are not the fastest. *Nat Ecol Evol*. 2017;1(8):1116-1122. doi:10.1038/s41559-017-0241-4