DM561 – Linear Algebra and Applications
DM562 – Scientific Programming

Obligatory Assignment 2, Autumn 2019

Deadline: Monday, November 25 at noon

This document is associated with the files `asg2.py` and `draw.py` and data files, which are available in the git repository. The file `asg2.py` is the only one that needs to be edited and submitted.

In the Introduction to Python - Part 3, on slide 9, we compared three important models for growth functions in computer science applications:

exponential model	$y = ae^{bx}$
power function model	$y = ax^b$
logarithmic model	$y = a + b \ln x$

where a and b are to be determined to fit experimental data as closely as possible. In this exercise you will work with a procedure called *linearization*, by which the data are transformed to a form in which a least squares straight line fit can be used to approximate the constants. Some calculus is required to carry out the task.
Let x denote the different size of two square matrices that we multiply with each other and y the computation time registered by executing, for example, the script benchmark from Assignment 1.
We will assume to have collected the following data $D = \{(x_i, y_i)\}$ (also available in the associated python script):

x	2	3	4	5	6	7	8	9
y	1.75	1.91	2.03	2.13	2.22	2.30	2.37	2.43

We will fit a linear model and the three models above using least squares and decide which is the best model.

Linear function Implement the function `least_squares(A,b)` that takes an appropriate matrix A and vector $\mathbf{b}$ and returns the least square solution $\mathbf{z}$ of the system $A\mathbf{z} = \mathbf{b}$. In the implementation, you are not allowed to use the function function `linalg.lstsq(A,b)` from Numpy and Scipy. Using the function `least_squares(A,b)` determine the linear function $y = ax + b$ that best fits the given data (x_i, y_i). You can visualize the situation executing the file `draw.py` that uses the code of the function `linear_model(x,y)` that you implement to draw a plot with the points and the fitted linear regression.

Exponential function Making the substitution

$$Y = \ln y$$

in the equation $y = ae^x$ produces the equation $Y = bx + \ln a$, whose graph in the xY-plane is a line of slope b and Y-intercept $\ln a$. Verify this fact!
Hence, a curve of the form $y = ae^x$ can be fitted to the given n data points (x_i, y_i) by letting $Y_i = \ln y_i$, then fitting a straight line to the transformed data points (x_i, Y_i) by least squares to find b and $\ln a$, and then computing a from $\ln a$. Implement this method to fit an exponential model in the function `exponential_fit` that uses your `least_squares(A,b)`. (You may need to use the functions `np.log` and `np.exp` for computing the natural logarithm). You can visualize the situation executing the file `draw.py` that uses the function that you implement to draw a plot with the points and the fitted exponential curve.

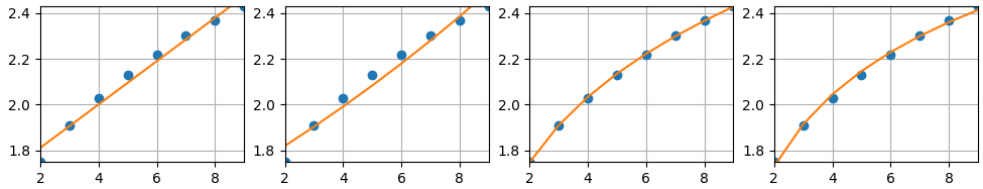

Figure 1: The result of this assignment.

Power function Making the substitutions

$$X = \ln x \qquad Y = \ln y$$

in the equation $y = ax^b$ produces the equation $Y = bX + \ln a$, whose graph in the XY-plane is a line of slope b and Y-intercept $\ln a$. Verify this fact!
Hence, a curve of the form $y = ae^x$ can be fitted to the given n data points (x_i, y_i) by letting $X_i = \ln x_i$ and $Y_i = \ln y_i$, then fitting a straight line to the transformed data points (X_i, Y_i) by least squares to find b and $\ln a$, and finally computing a from $\ln a$. Implement this method to fit an power model in the function `power_fit`. You can visualize the situation executing the file `draw.py` that uses the function that you implement to draw a plot with the points and the fitted curve.

Logarithmic function Making the substitution

$$X = \ln x$$

in the equation $y = a + b \ln x^b$ produces the equation $Y = a + bX$, whose graph in the Xy-plane is a line of slope b and y-intercept a. Verify this fact!
Hence, a curve of the form $y = a + b \ln x$ can be fitted to the given n data points (x_i, y_i) by letting $X_i = \ln x_i$, then fitting a straight line to the transformed data points (X_i, y_i) by least squares to find b and a, and finally computing a from $\ln a$. Implement this method to fit a logarithmic model in the function `logarithmic_fit`. You can visualize the situation executing the file `draw.py` that uses the function that you implement to draw a plot with the points and the fitted curve.

Training error Implement the function `training_error(f,x,y)` that returns the **root** sum of squared errors for the model `f`. Using this function compare the training error of the four models and determine which model has the best training error.