MATH-UA 253/MA-UY 3204 - Fall 2022 - Homework #2

Problem 1. Use Python to solve the nonlinear least squares problem:

$$\text{minimize} \qquad \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(x_i, c_1, c_2, c_3)\right)^2 \tag{1}$$

where:

$$f(x, c_1, c_2, c_3) = c_1 + c_2 e^{c_3 x}. \tag{2}$$

To do this, randomly choose (c_1, c_2, c_3) using `np.random`, pick N, sample N x_i's, and compute y_i by evaluating $f(x_i, c_1, c_2, c_3)$ for each x_i. To actually solve the nonlinear least squares problem, you should write a function that implements Gauss-Newton and use it to find (c_1, c_2, c_3). Next, add a small amount of noise to the observed y_i's. You can use `np.random.randn` times a small constant and solve the problem again. For each problem, make a plot of the predicted function, and evaluate the MMSE (the minimum mean squared error—just the cost function for this problem).

Go through the process described above for several different choices of parameters, different choices of N, and different noise intensities (vary the small parameter multiplying the output from `np.random.randn`). Write a short (~1 paragraph) explanation of your findings.

A word about Problem 1. Let $c^{(n)}$ be the nth iterate of your Gauss-Newton algorithm for computing the coefficients in the previous problem. You will generate a sequence of iterates:

$$c^{(0)}, c^{(1)}, c^{(2)}, \ldots \tag{3}$$

This sequence *may not converge*. The choice of $c^{(0)}$ plays an important role. For this problem, what is a good choice of $c^{(0)}$? That is, how can we make an educated guess about the coefficients without solving the problem exactly? The length of each step that you take is also important. We will learn more about this in class later, but for the time being, a good heuristic you can try using is as follows. Assume that your iteration is of the form:

$$c^{(n+1)} = c^{(n)} + \Delta c^{(n)}, \qquad n \geq 0, \tag{4}$$

and let $F^{(n)} = F(c^{(n)})$ be the cost function evaluated at n. If $F^{(n+1)} > F^{(n)}$ ($\Delta c^{(n)}$ isn't a "*descent direction*"—i.e., moving from $c^{(n)}$ to $c^{(n+1)}$ fails to decrease the cost function), then you can try modifying the step length. Replace your iteration with:

$$c^{(n+1)} = c^{(n)} + \alpha^{(n)}\Delta c^{(n)}, \qquad 0 \leq \alpha^{(n)} \leq 1, \qquad n \geq 0. \tag{5}$$

A simple way to calculate the factor $\alpha^{(n)}$ is to iterate:

$$\alpha^{(n)} := \beta\alpha^{(n)}, \qquad 0 < \beta < 1, \qquad \alpha^{(0)} = 1 \tag{6}$$

until $F^{(n+1)} < F^{(n)}$. (For example, pick $\beta = 0.9$, then scale the step length by β repeatedly until we've found a descent direction.)

Problem 2 (gradient flows). do an updated version of the problem in HW1 but using an ODE solver to show the idea of gradient flows