Practice Exercise 02

You may use MATLAB/python and any package within.

P.101 Q.4.13

A lot of technology, especially most types of digital audio devices for processing sound, is based on representing a signal of time as a sum of sine functions. Say the signal is some function *f (t)* on the interval $[-\pi, \pi]$ (a more general interval $[a, b]$ can easily be treated, but leads to slightly more complicated formulas). Instead of working with *f (t)* directly, we approximate *f* by the sum :

$$S_N(t) = \sum_{n=1}^{N} b_n \sin(nt),$$

where the coefficients *bn* must be adjusted such that *SN(t)* is a good approximation to *f (t)*. We shall in this exercise adjust *bn* by a trial-and-error process.

a) Make a function sinesum(t, b) that returns *SN(t)*, given the coefficients *bn* in an array b and time coordinates in an array t. Note that if t is an array, the return value is also an array.

b) Write a function test_sinesum() that calls sinesum(t, b) in a) and determines if the function computes a test case correctly. As test case, let t be an array with values $-\pi/2$ and $\pi/4$, choose $N = 2$, and $b1 = 4$ and $b2 = -3$. Compute *SN(t)* by hand to get reference values.

c) Make a function plot_compare(f, N, M) that plots the original function *f (t)* together with the sum of sines *SN(t)*, so that the quality of the approximation *SN(t)* can be examined visually. The argument f is a Python function implementing *f (t)*, N is the number of terms in the sum *SN(t)*, and M is the number of uniformly distributed *t* coordinates used to plot *f* and *SN*.

d) Write a function error(b, f, M) that returns a mathematical measure of the error in *SN(t)* as an approximation to *f (t)*:

$$E = \sqrt{\sum_i \left(f(t_i) - S_N(t_i)\right)^2},$$

where the t_i values are M uniformly distributed coordinates on $[-\pi, \pi]$. The array b holds the coefficients in S_N and f is a Python function implementing the mathematical function $f(t)$.

e) Make a function trial(f, N) for interactively giving b_n values and getting a plot on the screen where the resulting $S_N(t)$ is plotted together with $f(t)$. The error in the approximation should also be computed as indicated in d). The argument f is a Python function for $f(t)$ and N is the number of terms N in the sum $S_N(t)$. The trial function can run a loop where the user is asked for the b_n values in each pass of the loop and the corresponding plot is shown. You must find a way to terminate the loop when the experiments are over. Use M=500 in the calls to plot compare and error.

f) Choose $f(t)$ to be a straight-line $f(t) = 1\ \pi\ t$ on $[-\pi, \pi]$. Call trial(f, 3) and try to find through experimentation some values b_1, b_2, and b_3 such that the sum of sines $S_N(t)$ is a good approximation to the straight line.

g) Now we shall try to automate the procedure in f). Write a function that has three nested loops over values of b_1, b_2, and b_3. Let each loop cover the interval $[-1, 1]$ in steps of 0.1. For each combination of b_1, b_2, and b_3, the error in the approximation S_N should be computed. Use this to find, and print, the smallest error and the corresponding values of b_1, b_2, and b_3. Let the program also plot f and the approximation S_N corresponding to the smallest error.

SOLUTION:

https://hplgit.github.io/prog4comp/doc/pub/._p4c-bootstrap-Python014.html#2nd:exer:fitSines