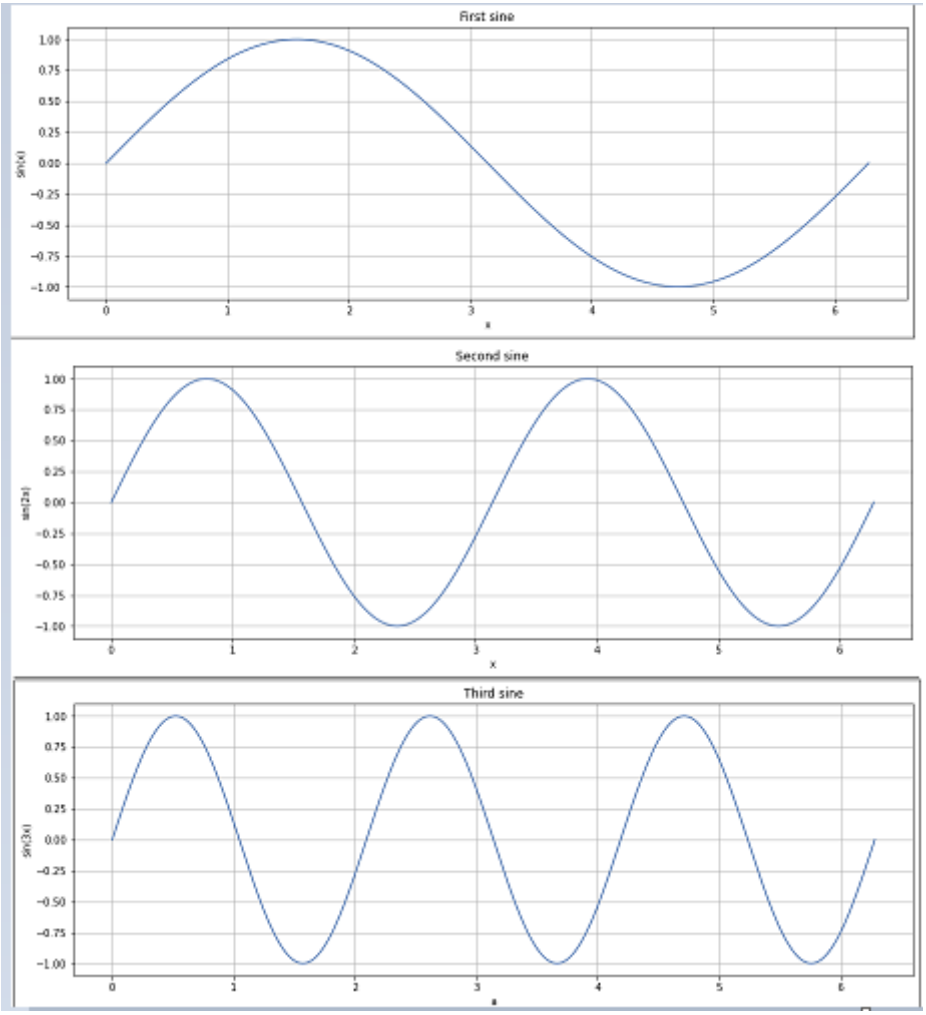
DSIP Lab 1

warmup :

A little **warmup** on looking the behaviour of multiplying the argument of $\sin$ and $\cos$ functions and getting familiar with Python tools, especially np dot product which came quite handy in calculating some otherwise "difficult" integrals. At some point the following exercise asked to resize in amplitude and "vertical shift" a triangular wave.



Approximating using fseries

fseries function which was tricky to get right due to the "double" amplitude resizing factior that comes out when using dot product. Since we assumed functions defined beetween $[0, 2\pi]$ the first normalizzation comes around considering the number of samples and the second (and third!) when I wanted to calculate the a_n and b_n terms whith the dot product (projecting to an orthonormal function). Summing up the overall right amplitude normalization to the hole series was:

$$\frac{(b-a)}{n} * \frac{1}{\sqrt{\pi}} * \frac{1}{\sqrt{\pi}} = \frac{(2\pi)}{n\pi} = \frac{2}{n}$$

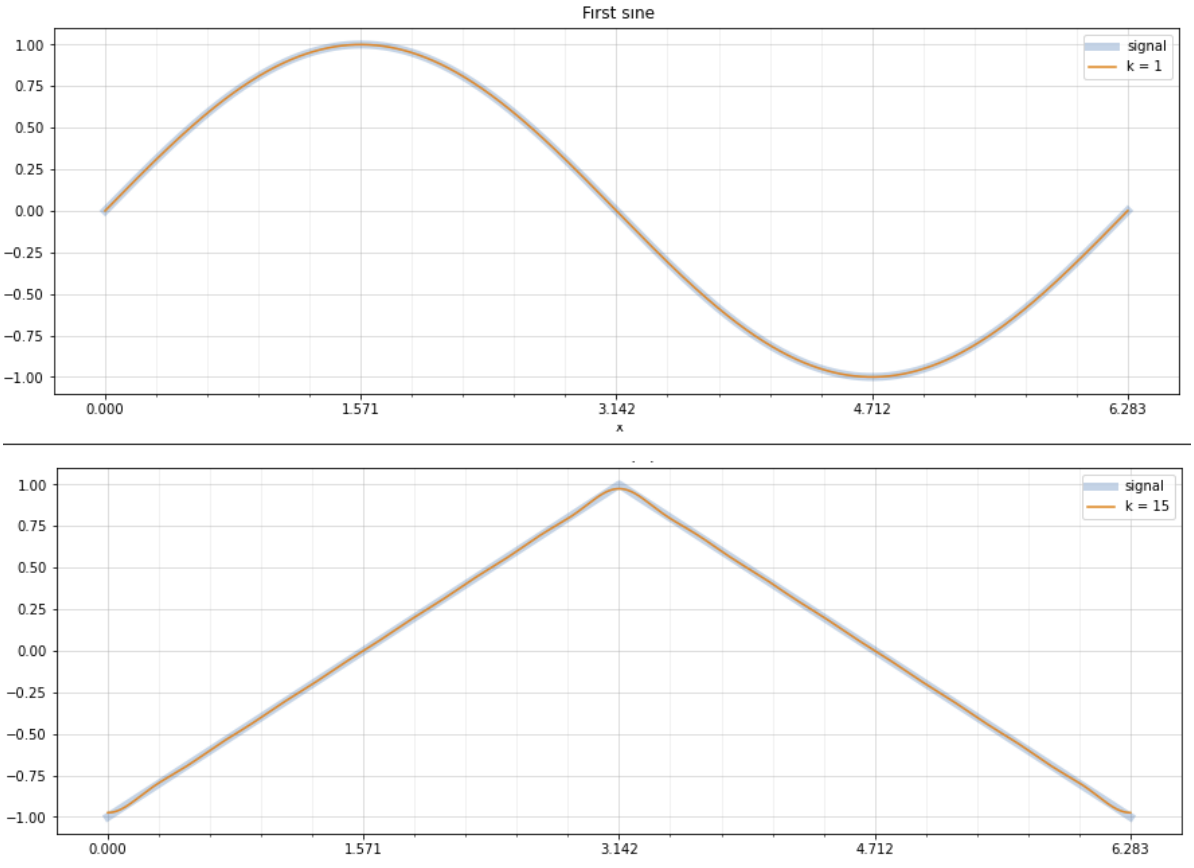
Parsival Remembering how orthogonality works, I like to think Parsival's equality as a generalization of Pythagoras theorem over a more generalized structure: Leaving all the details behind the concept here is that the sum of all (infinite and mutually orthogonal) projections *dotted* with themselves (and then normalized in amplitude) is equal to the signal's squared norm. When we take the difference within the two quantities, we have a **delta** that can be viewed as a measure of the approximation. Having this tools we can get any approximation we want with a discrete number of harmonics

It seems that the first k harmonics of the Fourier series can indeed contain as much information we need out of a signal, the other harmonics are intuitively "less important" giving a little contribute on the final approximation.

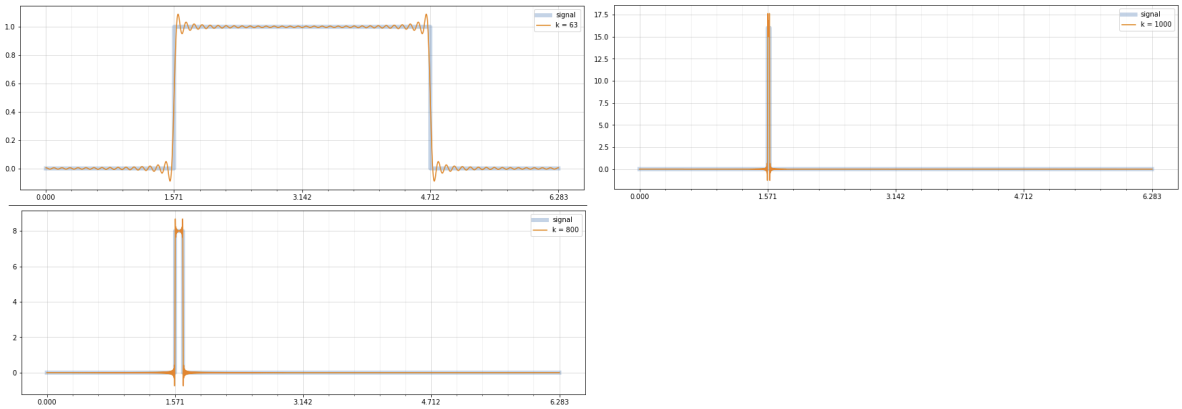
Plotting the signals

Case 1 that was a signal $\sin(x), x \in [0, 2\pi]$ of course it needs only the first harmonic of the series as $\sin(x)$ itself is a part of the basis in which we have to project the signal. the delta is in the order of 10^{-6}

Case 2 was a triangle wave which is continuous and does not have discontinuity points, but is not as "smooth" as the previous signal: As a matter of fact to obtain a $\Delta \leq 10^{-5}$ I needed **15** Fourier series terms

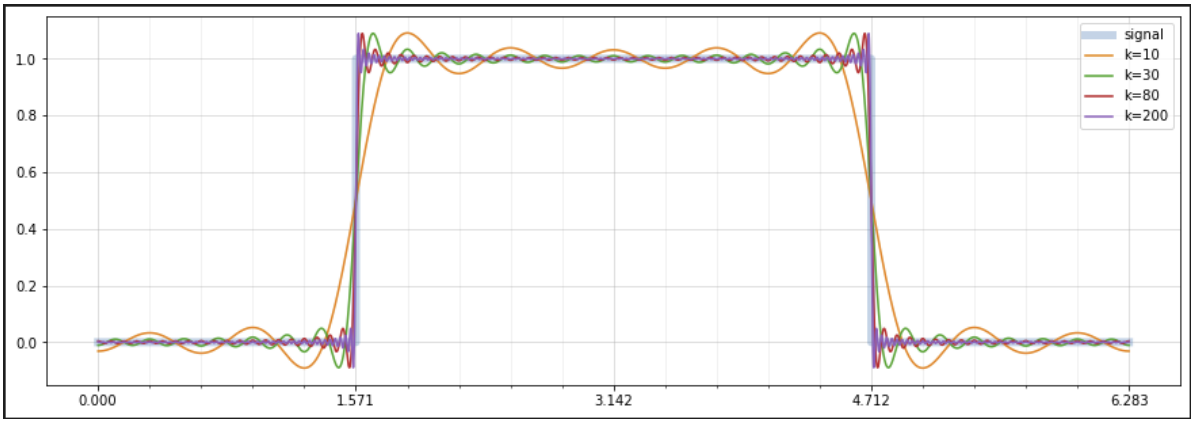


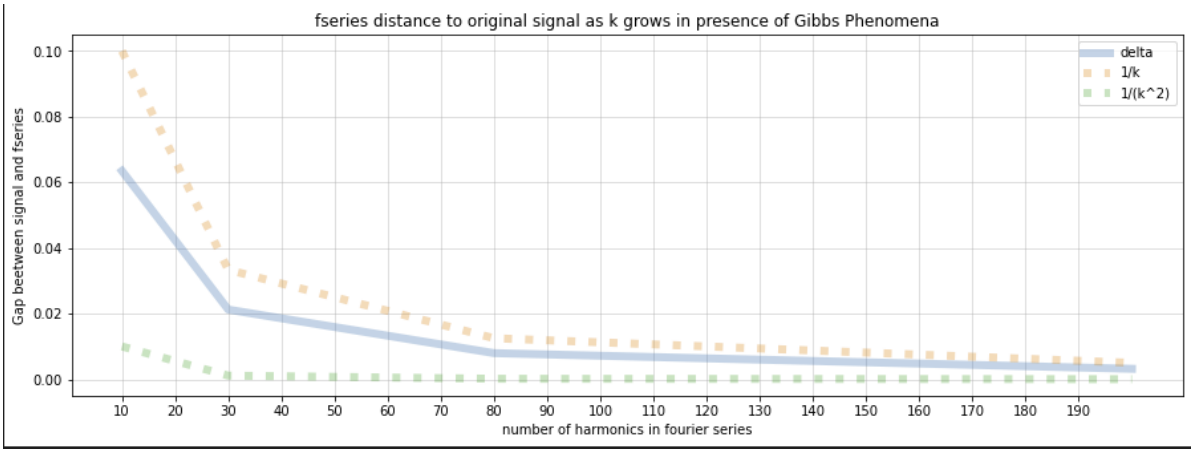
Case 3 Things became a little with a discontinuous square signal. After summing up 63 Fourier series terms I had $\Delta \leq 10^{-3}$. This time I can notice weird oscillations around the 4 edges of the square, it looks like things get worst when the area of the square is concentrated around a single value, meaning same area but smaller base as shown in the following plots:



In the worst case, (the case 3c.) even with a thousand Fourier terms and 3 minutes of CPU workload I just got close to $\Delta = 0.15$.

Gibbs Discontinuous signals that have "jumps" from one value to another may look easy to understand, but not for Fourier series: As the approximation goes near those discontinuity points, a bad oscillating behaviour starts to happened slowing the speed in which the *delta* reduces as the number k of harmonics grows.

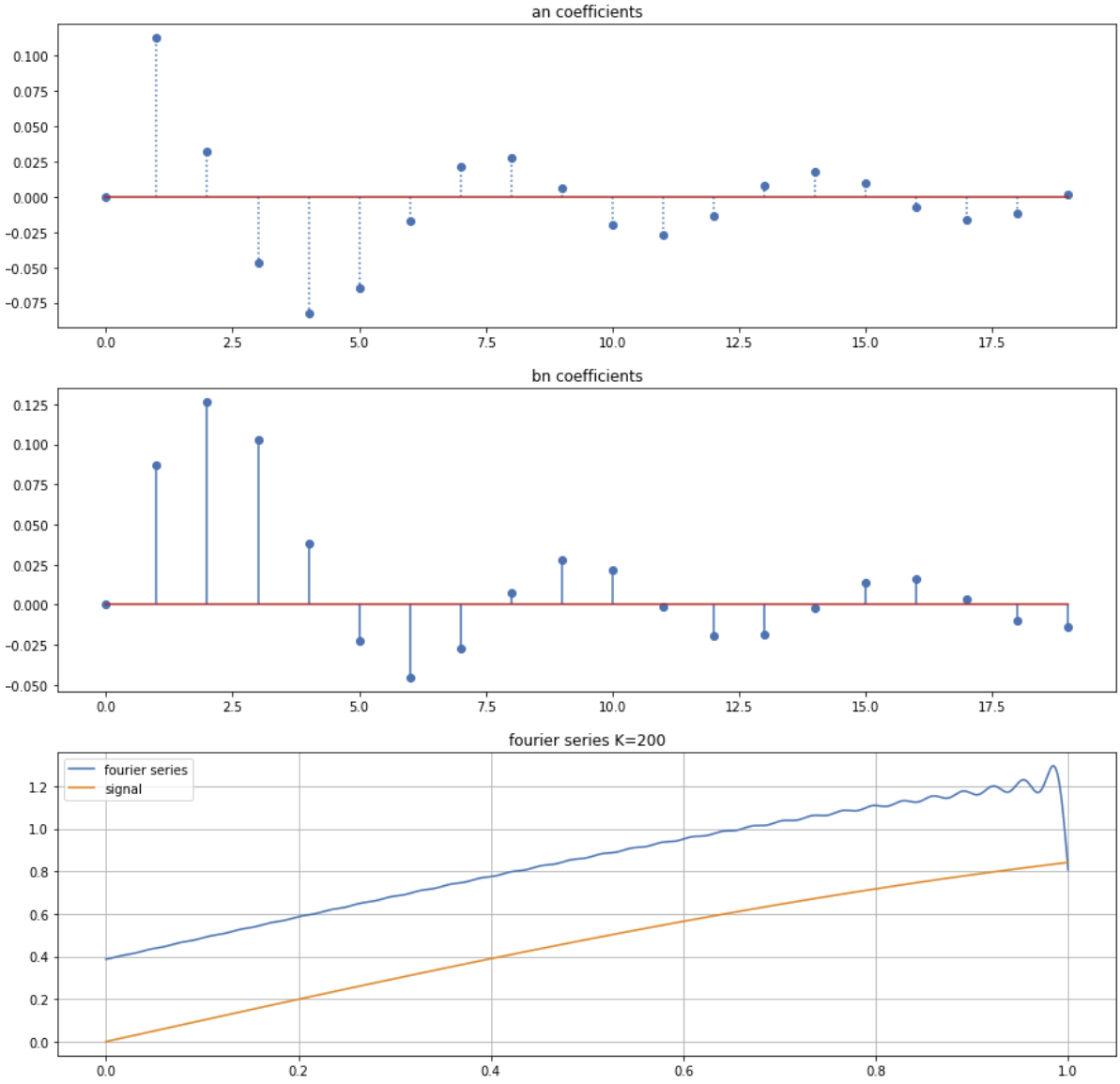




Additional questions

Q 1

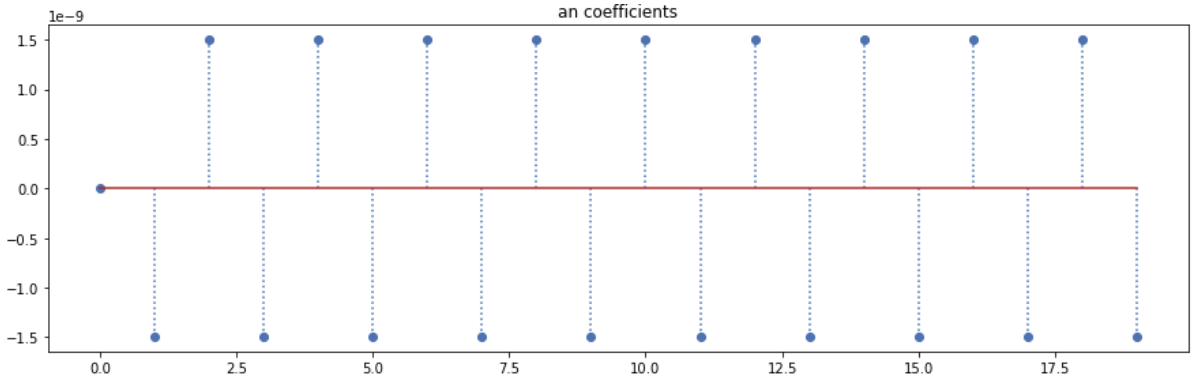
I tried to approximate using fourier series $\sin(x)$ between $x \in [0, 1]$ with $k = 200$ harmonics. The general shape of the approximation looks good but somehow vertically shifted. In order to find out why I tried to have a look at the coefficients an/bn with the following results:

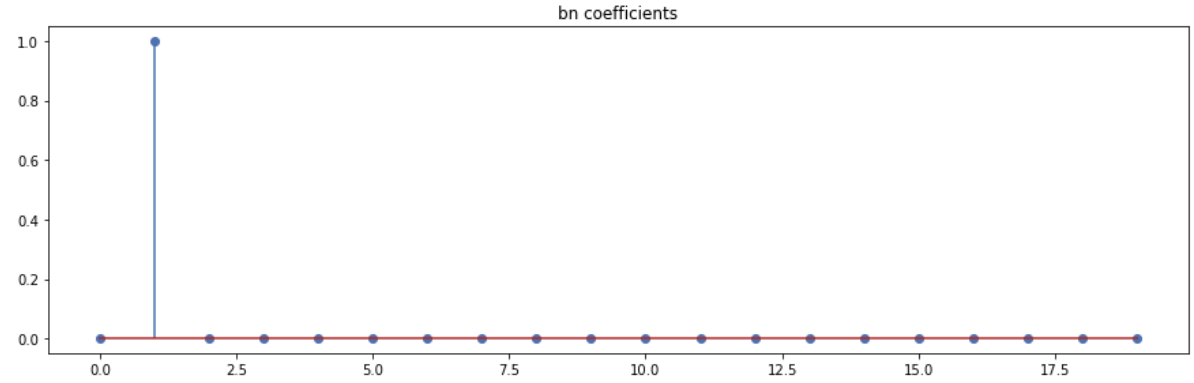


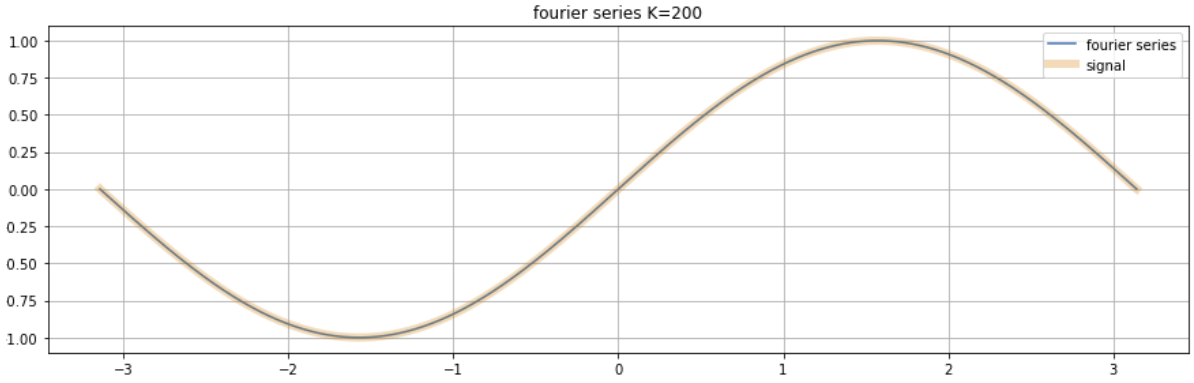
Even if it is a simple sinusoid function surplisingly a bad sampling in 0, 1 can lead to a difficult approximations

Q 2

This signal is a sinusoid $\sin(x) + \frac{\cos(300x)}{1000}$ summed with a very small "noise" in amplitude but very fast in frequency which look like does not affect the signal reconstruction which captures the general trend in the b_1 coefficient of the Fourier series.


an coefficients
1e−9
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


bn coefficients
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


fourier series K=200
fourier series
signal
1.00
0.75
0.50
0.25
0.00
0.25
0.50
0.75
1.00
−3
−2
−1
0
1
2
3