Exercise 1. Impedance mismatch in coaxial cables

Matching terminating impedance for coaxial cables (or any transmission line) is vitally important for short pulses. The effect of not terminating the cable with its characteristic impedance of 50 Ω causes reflections that result in artefacts appearing at the output of the cable.

We can create short (~ns) pulses using a digital delay generator. Here, we assemble a pulse from edge transitions – the delay time to the start of the pulse and the delay time to the end of the pulse are specified, and the edge transitions are added at the output terminal. All delay times are relative to the trigger event, here generated internally at a fixed rate.

Please watch the accompanying video which explores varying the termination of the cable with various pulse settings, and answer the questions on the observed behaviour.

Equipment:

- DG535 digital pulse generator
- DSO (200 MHz min)
- Coax cable: 3 m
- Terminators 2x 50 ohm
- Tee piece

- Ensure a 50 Ω terminator is across outputs A ⎺⎿ B of the digital delay generator (required for using the pulse outputs).

Set up the pulse generator with the following settings:

- Press TRIG; set to "Int" (internal) and rate to 100 Hz
- Press DELAY; set "A" to A=T+0
- Press DELAY; set "B" to B=A+ 10 ns
- Press OUTPUT; highlight "AB" (use arrow keys)
- Press OUTPUT; set "AB & -AB LOADS=50Ω"
- Press OUTPUT; set to "TTL"

- Connect the 3 metre coax cable to A ⎾⎿ B output on signal generator
- Connect the other end of the coax to the DSO using a tee piece and a terminator so that the cable is terminated with 50 Ω.

Task 1: Set up the DSO and observe the 10 ns, 4 V pulse on the DSO.
Q1. How does the waveform at the DSO change when the terminator is removed?
Terminators help to minimise signal distortions and reflections , therefore the waveform changes from a single pulse form to several pulses in a row due to these reflections going back and forward in the cable.

Task 2: Assuming that reflections from the mismatched termination of the cable are visible, the velocity of propagation is 0.66c, calculate the length of the cable. Note: use the "ΔX" time value displayed in the cursor window, which is the time difference between the two cursors.
Q2. Does your calculated length of the cable agree with the measured length of 3.03 m?
No, the calculation we made disagrees with the reported length of the cable.
In the video: 15 reflections occur in 547 ns.
we take an average for the "ΔX" value:
547 ns / 15 = 36.47 ns
The distance travelled for a round trip is d = v*t = 0.66c * 36.47e-9 s = 7.22 m
Therefore 7.22 / 2 = 3.61 m which differs from the reported length 3.03 m by 0.58 m.

Q3. What are the probable sources of the discrepancy?
I am not sure but from the optional section below. Maybe its due to an apparent "extra length" perceived by the signal as it travels through the cable?

Optional: You can work out the actual propagation velocity and the "extra length" by combing the measurements for the single and double length cable (cable joiner is 0.02 m long).

Task 3: With the terminator removed, investigate increasing the length of the pulse by changing the "B" delay until the pulse becomes several hundred nanoseconds long.
Q4. What do you note about the shape of the pulse?
Distortion occurs. The waveform appears as a noisy decaying wave.

Q5. What do you note about the amplitude of the pulse?
it has blown up! the amplitude is 25.70V

Task 4: Re-attach the terminator to confirm that the expected form of the pulse is observed when the cable is correctly terminated. With the pulse length again reduced to 10 ns, change the internal output impedance of the delay generator by setting the output of the "AB" outputs to the "High Z" setting.

Q6. Investigate the effects of connecting and disconnecting the terminator.
Why do you no longer see reflections when the terminator is removed? (Hint: notice the difference in pulse amplitude when the terminator is added).
The reflections and distortions visible in the DSO change.

Q7. What do you think the "High Z" setting actually introduces at the output of the delay generator?
It changes the impedance of the cable automatically to match cancel out the noise from the reflections?

Exercise 2. Synthesising a square wave using sine waves

The central concept in Fourier analysis is that any function can be composed from a series of basis sinusoids.
A square wave consists of summation of a base sinusoidal wave (of frequency f) and an infinite number of odd harmonics (that is waves with frequencies $3f, 5f, 7f$ ...).
Components must also be added in the correct proportions. The amplitudes of the required components are inversely proportional to the number of the harmonic. Consequently, a square wave (without a phase shift) and with an amplitude of 1 is given by:

$$S(t)=\frac{4}{\pi}\sum_{n=1,3,5...}^{\infty}\frac{1}{n}\sin(2\pi n f t)\ ,$$

or explicitly:

$$S(t)=\frac{4}{\pi}\left[\sin(2\pi f t)+\frac{1}{3}\sin(6\pi f t)+\frac{1}{5}\sin(10\pi f t)+...\right].$$

Investigate summing a Fourier series to create a square wave (Matlab suggested), using a base frequency of 1 cycle per 2π (i.e. $f=1$) and its odd harmonics.
It is suggested to use 1000 time points ranging from -2π to $+2\pi$. (use `t= linspace(-2*pi,2*pi, 1000)` in Matlab to do this).

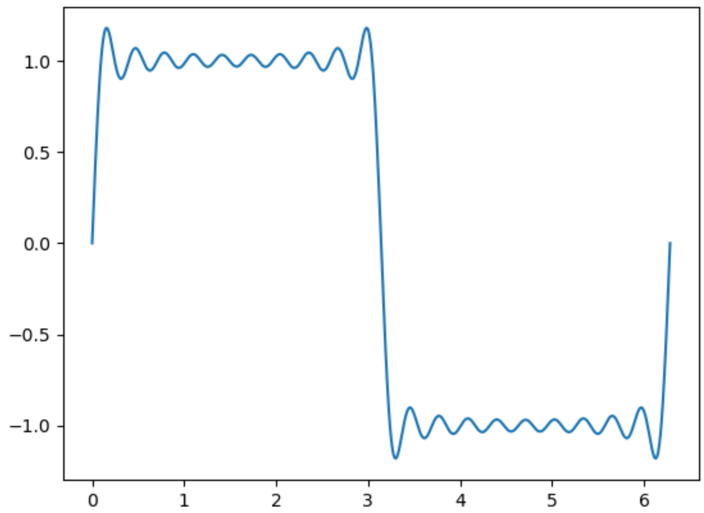
There are several ways to perform this calculation.

The most elegant uses matrix algebra to form a matrix of values for all frequency components at all time points, which are then summed (use `sum`) to give the final wave. (Remember that in Matlab, .* is element-by-element multiplication and * is matrix multiplication).
An alternative is to use a loop to add each wave to each other (see the Matlab help on `for` to see syntax and examples).

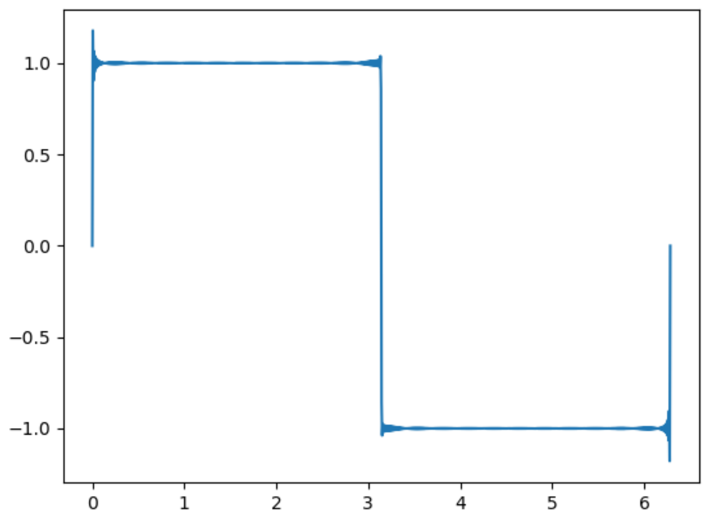
Tasks:

- Plot a square wave assembled from the first 10 appropriate sinusoidal components.
- Plot a square wave assembled from the first 250 appropriate sinusoidal components.

10 sinusoidal components



250 sinusoidal components



10/23/20 04:37:46 /home/ana/Documents/uni/PHS3000/weekly works/wk10/wk10.py

```
# PHS3000
# week 10 - FOURIER TRANSFORMS
# Ana Fabela Hinojosa, 23/10/2020
import numpy as np
import matplotlib.pyplot as plt

nterms = 250 # modify this value to get desired number of sinusoidal components

t = np.linspace(0, 2 * np.pi, 1024)
omega = np.arange(1, 2*nterms, 2).reshape(nterms,1)

# print(f"{t}")
# print(f"{omega}")

sin = np.sin(omega * t)
# print(f"{sin}")

S = (4 / (np.pi * omega) * sin).sum(axis=0)
# print(f"{S}")

plt.plot(t, S)
plt.show()
```