DSIP Lab 3

Andres Coronado

Rectangle signal using $u(t, T0)$

- An elegant one liner to define a rectangle using the unitary step
- plot of some signals using the following

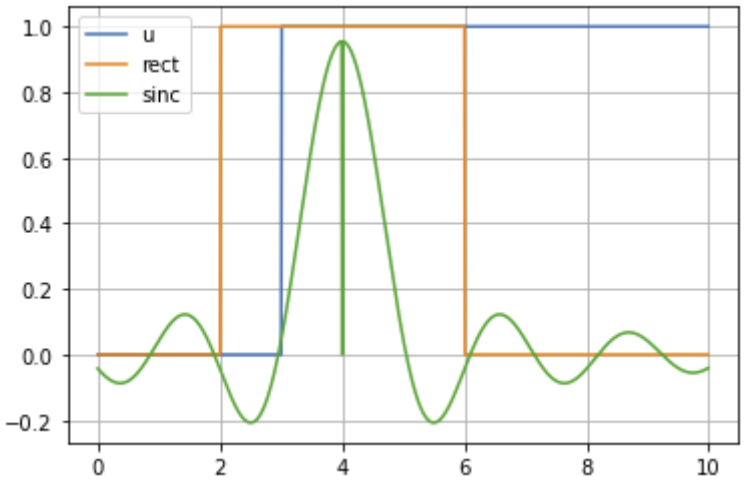
```
def u(t, T0):
    """ u(t-T0) """
    return np.array([1 if x > T0 else 0 for x in t])

def rect(t, T0, T):
    """ rect((t-T0)/T) """
    return u(t, T0-T) - u(t-T, T0)

def sinc(t, T0, W,epsilon=10e-10):
    """ sinc(b(t-T0)) - its FT has bandwidth 2W in omega space"""
    return np.sin(W*(t-T0)) / (np.pi * (epsilon+(t-T0)))

t = np.arange(0,10,10e-3)

plt.plot(t, u(t, 3), label="u")
plt.plot(t, rect(t, 4, 2),label="rect")
plt.plot(t, sinc(t, 4, 3), label="sinc")
plt.legend()
plt.grid()
```



Convolution

With $W = 20$ "looks like" the sync wave takes 2 periods of the summed cosine functions so the overlapping area over time is not very regular and has a notable scaling difference compered to the "slow" function, given the fact that the amplitude of the sync is very little. The reconstructed convolution using the frequency domain looks much better but quite different both in shape and in amplitude, due to the fact that some waves overlaps in the time space, but it does not in the frequency space.

With $W = 50$ things change: there is no overlapping between the waves and leaving out the `time_max` and `-time_max` the convolution looks very similar to the "slow" function.

As expected setting the W parameter on the sinc pulse signal with higher angular velocity 90 makes the convolution exactly like the slow signal because it "captures" its frequency with the convolution with the sinc.

If I try to convolve with higher $W >= 135$ the convolution starts to look like the sum of the cosines as it starts to capture all the inner frequencies of $y(t)$