# 2 Nov 23 - Activity: Automated Signal Deconstruction & Reconstruction

We found that it was possible to use the Fourier transform to decompose the signal into its constituent frequencies. Later we will use this idea to build up the [Fast Fourier Transform](https://en.wikipedia.org/wiki/Fast_Fourier_transform) (FFT) algorithm. For now, we will write everything out in full to mirror our analytical work.

We will start with the same signal as before, but then you will work with other signals to achieve a particular signal decomposition.

Recall that the sign was given by the square wave function:

We define the approximate form of the signal using the complex form of the Fourier series:

where and is the Fourier coefficient given by:

Remember that we use the real to compute the complex . And once we have found the we can reconstruct the signal using the series expansion.

where is the number of terms we use in the series expansion.

## Analytical Calculation

**✅ Do this**

1. Take the signal and determine which integrals you have to do.
2. Construct the integrals you need to do (over what cycle are you integrating?).
3. Perform the integrals (consider using sympy) and find the coefficients ’s.
4. Write the function in terms of the ’s. (What about the imaginary part?)

## Automating the Calculation

This is a lot of work to do by hand, and we have to perform new integrals every time we want to find this decomposition. We can automate this process using a little code. We have to break the process up into a few steps.

1. Define the signal .
   * Here you need to make sure you have a function that can be evaluated for any .
   * You should also be able to change that function easily for different signals.
2. Compute the Fourier coefficient for a given .
   * Here you need to perform an integral of over the period .
   * It would be good to use a builtin integrator like numpy.trapz or scipy.integrate.quad to do this.
   * The documentation for numpy.trapz can be found [here](https://docs.scipy.org/doc/numpy/reference/generated/numpy.trapz.html) and the documentation for scipy.integrate.quad can be found [here](https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.quad.html).
3. Compute the approximation up to a given .
   * Here you need a complex array to take on the approximate values for .
   * You will need to loop over the values and add the contribution from each to the array.
   * Return the real part of the array to get the approximation.

**✅ Do this**

1. Review the code scaffold below and make sure you understand what is happening.
2. Complete the code below to compute the Fourier coefficients for a given .
3. Complete the code below to compute the approximation up to a given .
4. Test your code with the provided signal and plot the approximation for different values of .
5. Write function that plots your real and approximate signal on the same plot.
6. Write a function that plots the magnitude of Fourier coefficients as a function of .

import numpy as np  
import matplotlib.pyplot as plt  
from scipy.signal import square  
  
# Compute the Fourier coefficient  
def compute\_cn(v, n, T, f0):  
   
 ## CODE HERE ##  
   
 return cn  
  
# Fourier series expansion using complex form  
def complex\_fourier\_series\_expansion(v, t, T, N):  
 f0 = 1 / T  
 series = np.zeros\_like(t, dtype=complex)  
  
 ## CODE HERE ##  
   
 return series.real  
  
# Define the square wave signal  
T = 2 \* np.pi # Period  
def v(t):  
 return square(t)

## Physics Relationship

You might be wondering why we do this kind of deconstruction and what it is used for. It turns out that for many systems this kind of analysis tells you how energy, momentum, and other quantities might be distributed in the system. For example, the analogy to a complex quantum mechanical system might be the various occupied energy states of the system. The known frequencies could correspond to the energy levels of the system. The Fourier coefficients tell you how much of the system is in each energy state. There are many such analogies in physics; and many such transformations in multiple dimensions to extract this information. Fourier is just one of the most common.

## Decomposition Matching

We want to develop your intuition for this kind of analysis now that you have started to develop the tools to do it. So we will ask you to use your code to perform a few tasks. But these will start from the expected coefficients and ask you to find the signal. This is the reverse of what we have done so far. With your code you should be able to play and adjust the signal to match the coefficients qualitatively.

**✅ Do this**

Use your code to find the signal that matches the following coefficients: 1. No odd coefficients. 2. No even coefficients. 3. Only the coefficient. 4. Only the coefficient. 5. High values for low and low values for high (not too high). 6. Low values for low and high values for high (not too high). 7. Roughly equal values for all .

In doing this, take note of the signals you are making. What are you doing to construct them to produce this kind of output. What kind of device or physical system might produce those signals?

### your code here