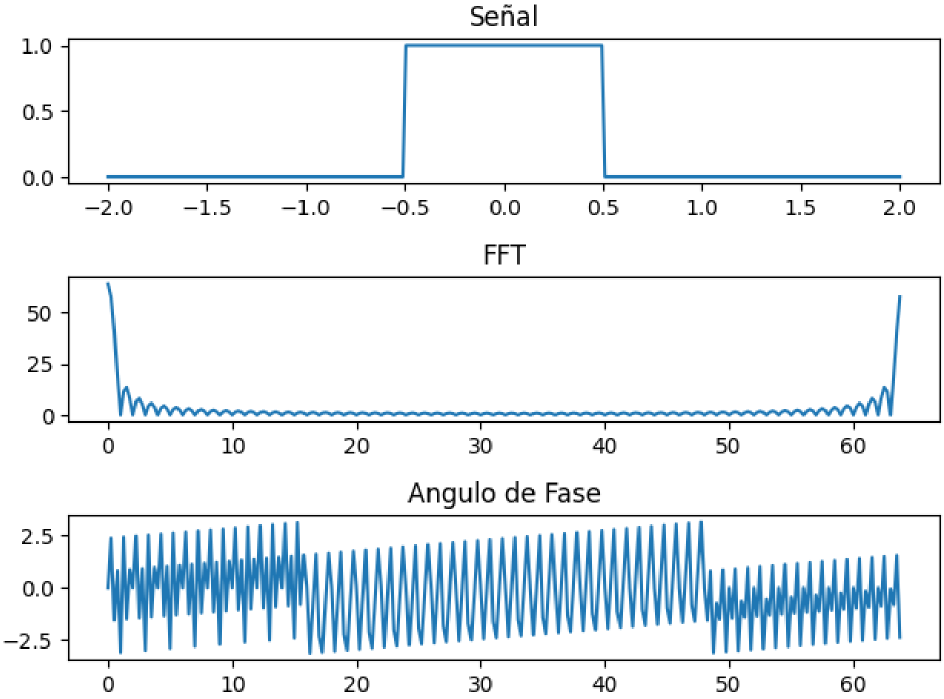
CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD TAMAULIPAS

Assignment #3

Student: *Luis Alberto Ballado Aradias*
Course: *Introducción al Análisis de Fourier (Sep - Dec 2022)*
Professor: *Dr. Wilfrido Gómez-Flores*
November 22, 2022

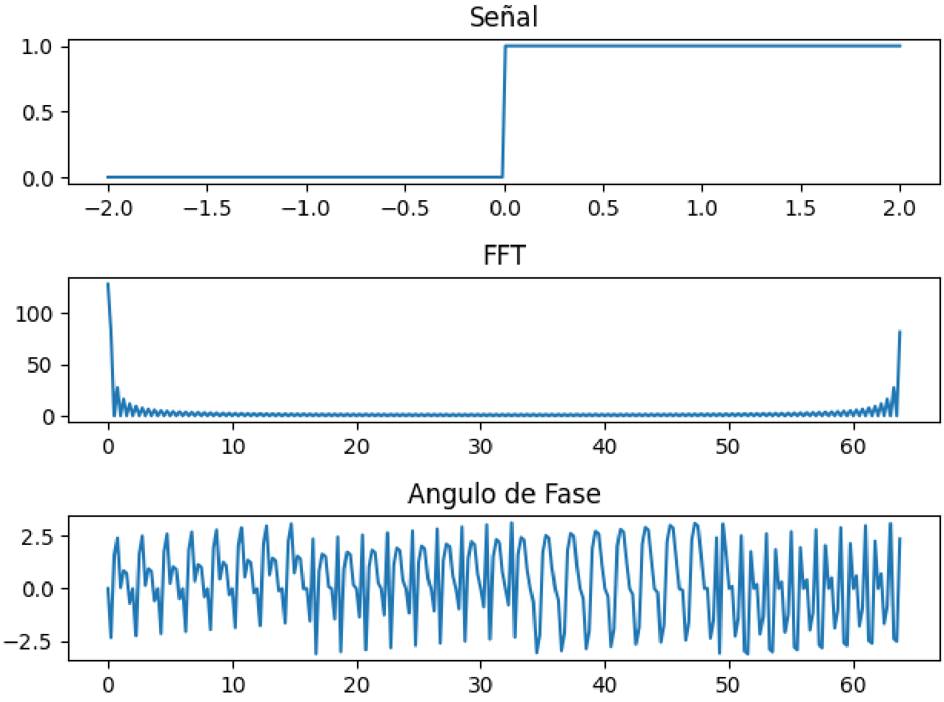
PULSO RECTANGULAR

$$f(t) = \begin{cases} 1, & |t| < \frac{\tau}{2} \\ 0, & |t| > \frac{\tau}{2} \end{cases}$$



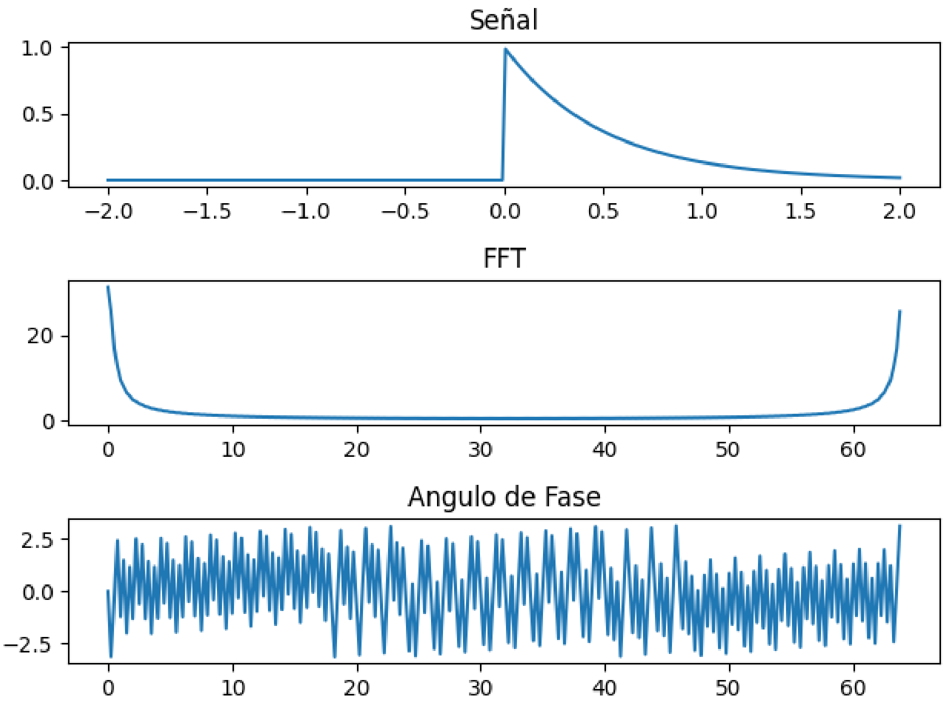
ESCALON UNITARIO

$$f(t) = \begin{cases} 1, & t > 0 \\ 0, & t < 0 \end{cases}$$



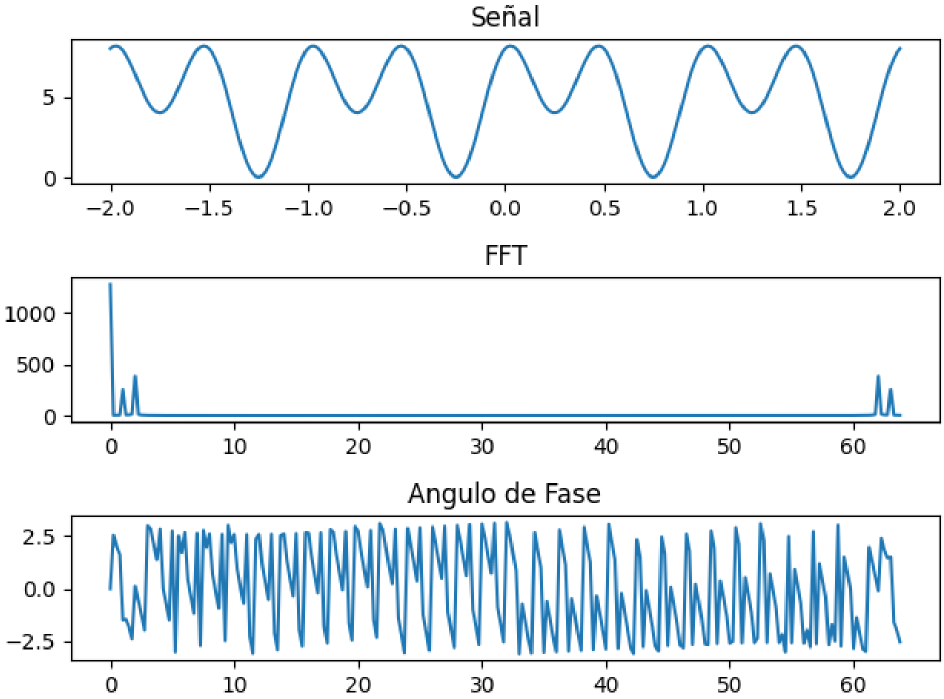
EXPONENCIAL

$$f(t) = \begin{cases} e^{-at}, & t > 0 \\ 0, & t < 0 \end{cases}$$



FUNCION A

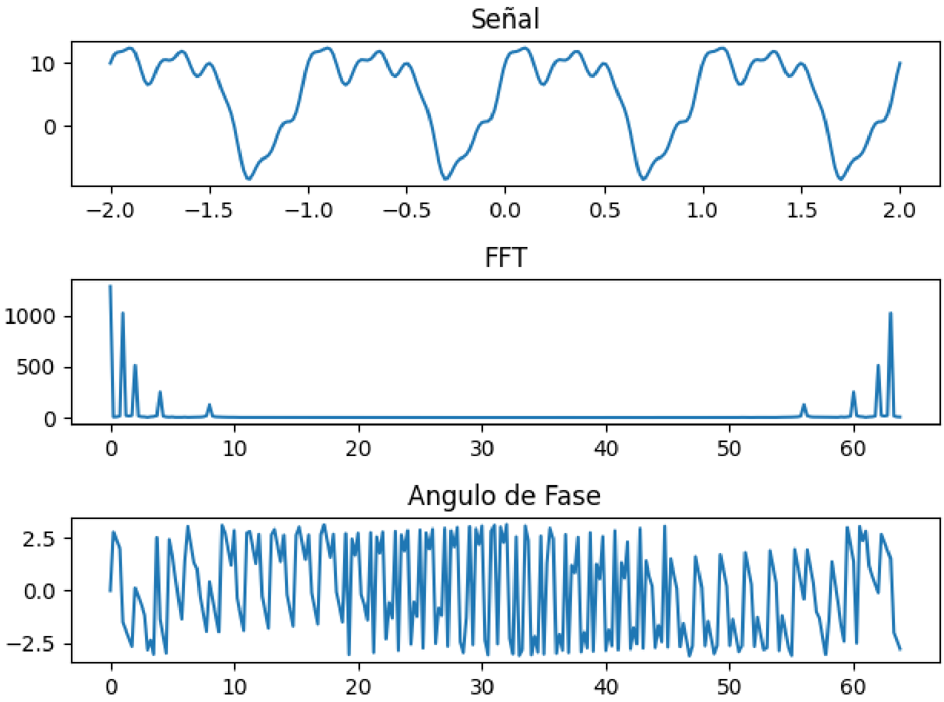
$$f(t) = 5 + 2 * cos(2 * \pi * t - \pi/2) + 3 * cos(4 * \pi * t)$$



FUNCION B

$$f(t) = 5 + 8 * cos(2 * \pi * t - \frac{\pi}{2}) + 4 * cos(4 * \pi * t) + 2 * cos(8 * \pi * t - \frac{\pi}{2}) + cos(16 * \pi * t) +$$

$$+2 * cos(32 * \pi * t - \frac{\pi}{2})$$



Código python

```python
def pulso_rectangular(t):
    '''PULSO RECTANGULAR'''
    return 1 * (abs(t) < 0.5)

def escalon_unitatio(t):
    '''ESCALON UNITARIO'''
    return 1 * (t >= 0)

def exponencial(t):
    '''FUNCION EXPONENCIAL'''
    alpha = randrange(3)
    return np.exp(-alpha * t) * (t > 0)

def funcion_a(t):
    '''FUNCION A'''
    return 5+2*np.cos((2*np.pi*t)-(np.pi/2)) + 3*np.cos(4*np.pi*t)

def funcion_b(t):
    '''FUNCION B'''
    return 5+8*np.cos((2*np.pi*t)-(np.pi/2))+4*np.cos(4*np.pi*t)+2*np.cos((8*np.pi*t)-(np.pi/2))+np.cos(16*np.pi*t)+2*np.cos((32*np.pi-(np.pi/2)))
```

Código 1: Dibujo de funciones

```
def FFT2(x):
    ''' radix-2 FFT '''
    x = np.array(x, dtype=float)
    N = int(x.size)
    n = np.log2(N)
    d = 1
    for i in range(1,int(n)):
        w = np.exp(-(1j*2*np.pi)/(2*d))
        for a in range(0,d-1):
            b = 0
            while(b<N-1):
                wa=pow(w,a)
                id1 = b+a+1
                id2 = b+d+a+1
                t_0 = x[id1] + (wa)*x[id2]
                t_1 = x[id1] - (wa)*x[id2]
                x[id1] = t_0
                x[id2] = t_1
                b = b+2*d

        d = 2*d
    return x
```

Código 2: FFT - radix2

```

def bit_reversal(data):
    '''
    gold rader - zero padding
    :param data: array de datos
    :return: array signal con zeros
    '''
    n = int(data.size)
    j = 0
    i = 0
    while i < n-1:
        k = n/2
        if (i<j):
            temp = data[int(i)]
            data[int(i)] = data[int(j)]
            data[int(j)] = temp
        while(k<=j):
            j = j-k
            k = k/2
        j = j+k
        i += 1
    return data
```

Código 3: Algoritmo Gold Rader