Universidade de São Paulo
Instituto de Física de São Carlos
Mathematical-Computational
Modeling

Harmonic Signals

Éverton Luís Mendes da Silva (10728171)

1 Introduction

In this project, signals corresponding to the sum of cosines according to the harmonine series will be analyzed, with a reference frequency set. For each signal, three types of component amplitude were used. Finally, a damped harmonic oscillator signal obtained from the Physics Laboratory 2 was studied.

2 Synthesized Signs

For a better analysis of these types of signals, two frequencies were used as a reference.

$$\begin{cases} f_1 = 10Hz \\ f_2 = 25Hz \end{cases}$$

For each one, the harmonic series up to grade 4 was used.

$$\begin{cases} n = 1, 2, 3, 4 \end{cases}$$

The signals were obtained using the formula below:

$$signal(t) = m(t) \sum_n cos(2\pi f_n t) \tag{1}$$

Where m (t) can be one of the three functions below:

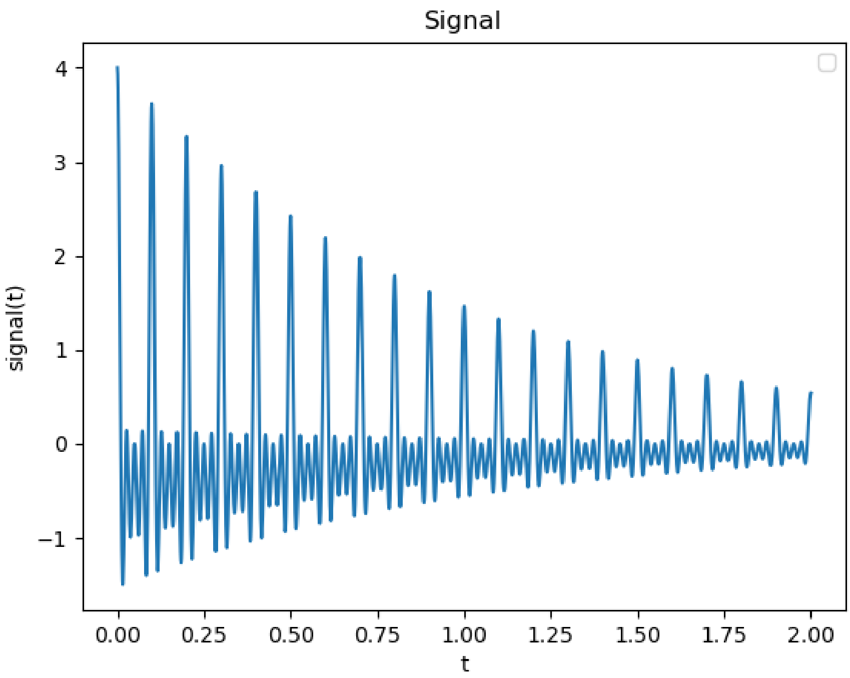
$$\text{m(t)} \begin{cases} f_1(t) = e^{-t} \\ f_2(t) = t^2 - t^3 \\ f_3(t) = t^2 - t^4 + tcos(t) \end{cases}$$

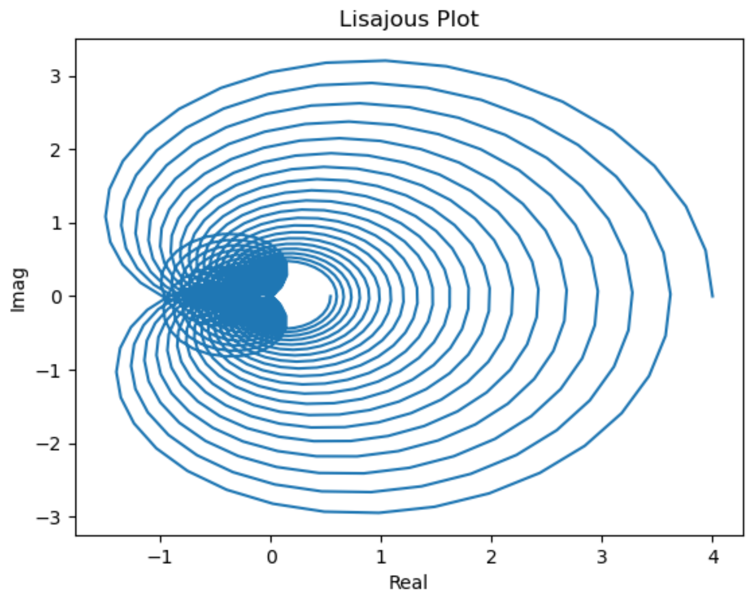
3 Visualization of Signals

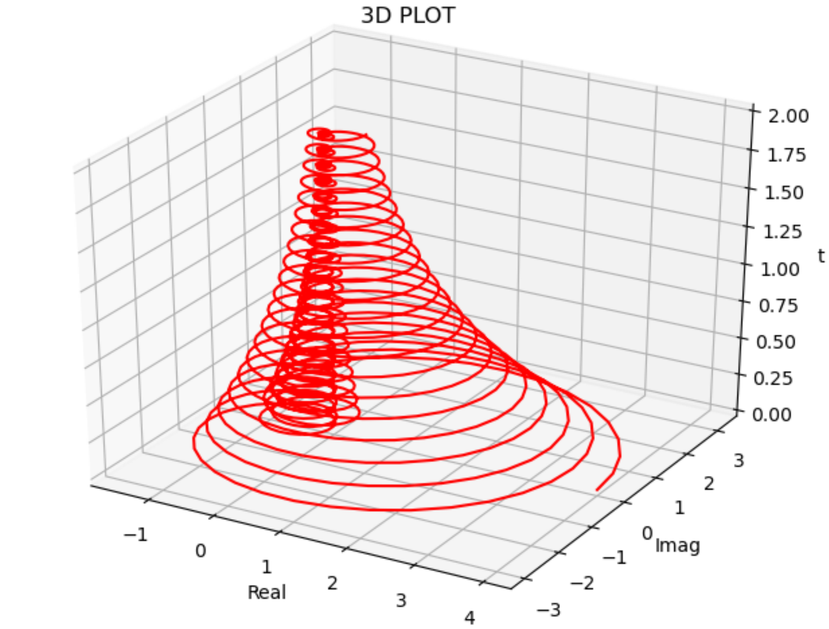
```
#function that returns a signal which is the sum of several cosines
def SignalCosine(m, freq, time=2, H=4):

    signal=[]
    signal1=[]
    t=np.linspace(0,time, 2000)

    Hlist=[]
    for i in range(H):
        Hlist.append(i+1)

    for j in t:
        sum_cosines=0.0
        sum_sines=0.0

        for i in Hlist:

            argument=2*(np.pi)*freq*i*j

            element=np.cos(argument)
            element1=np.sin(argument)

            sum_sines+=float(element1)
            sum_cosines+=float(element)

        sum_cosines= sum_cosines*(m(j))
        sum_sines= sum_sines*(m(j))

        signal.append(sum_cosines)
        signal1.append(sum_sines)

    return signal, signal1, t
```

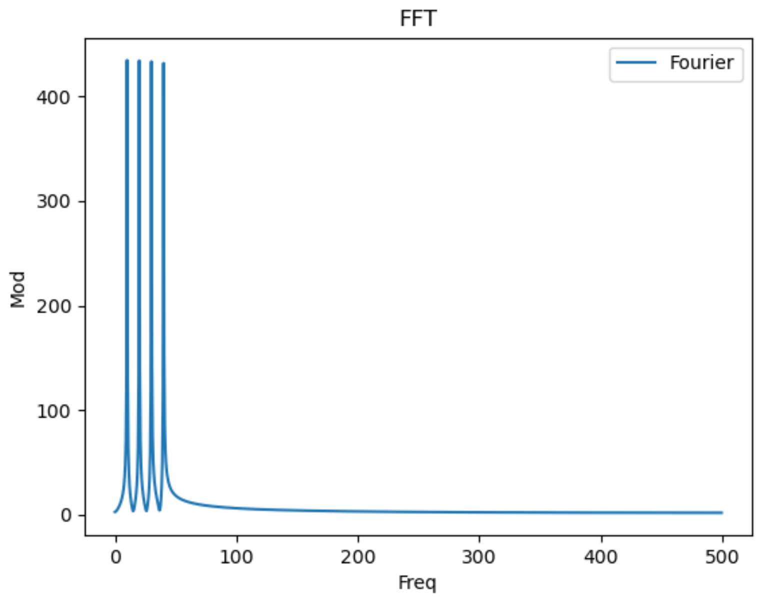
3.1 First Signal (10Hz)

3.1.1 f_1

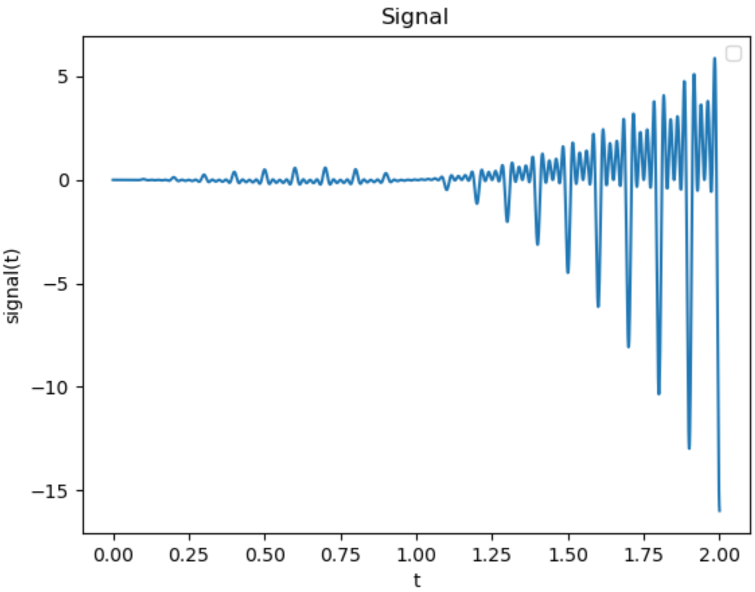

Signal
signal(t)
t
4
3
2
1
0
−1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

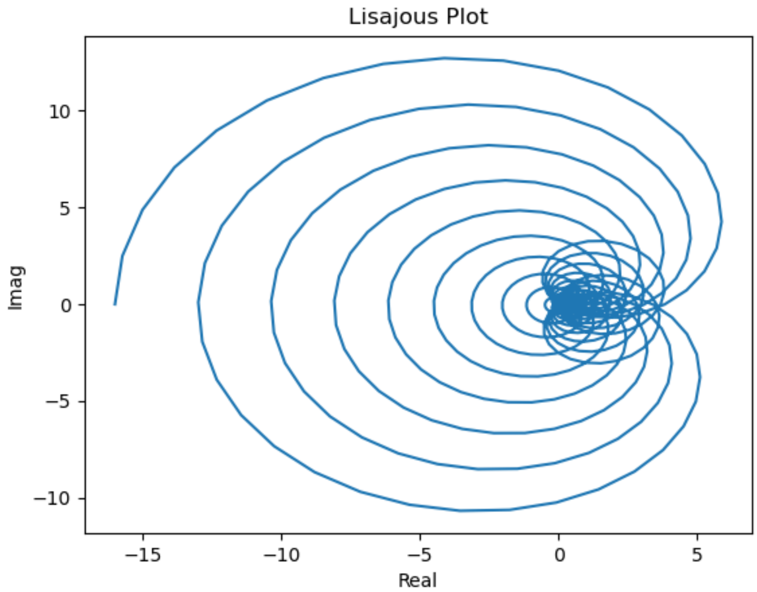

Lisajous Plot
Imag
Real
3
2
1
0
−1
−2
−3
−1
0
1
2
3
4

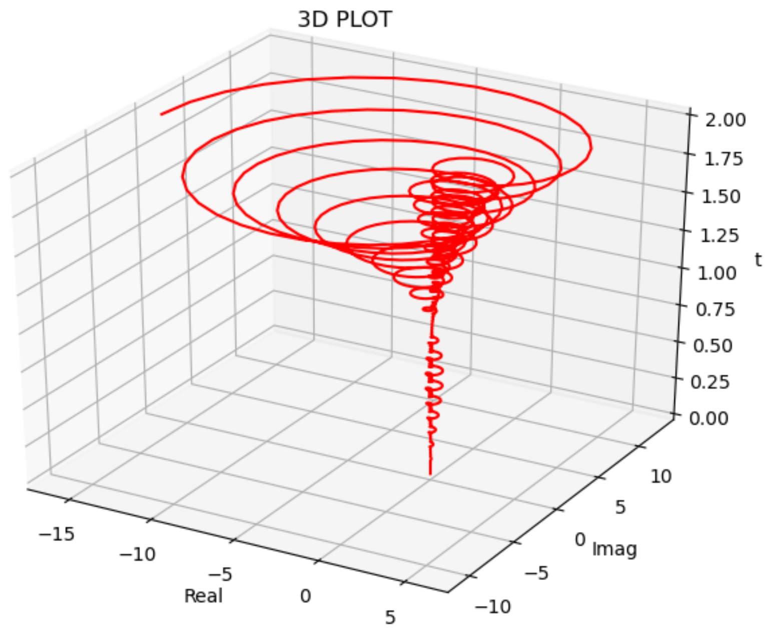

3D PLOT
Real
Imag
t

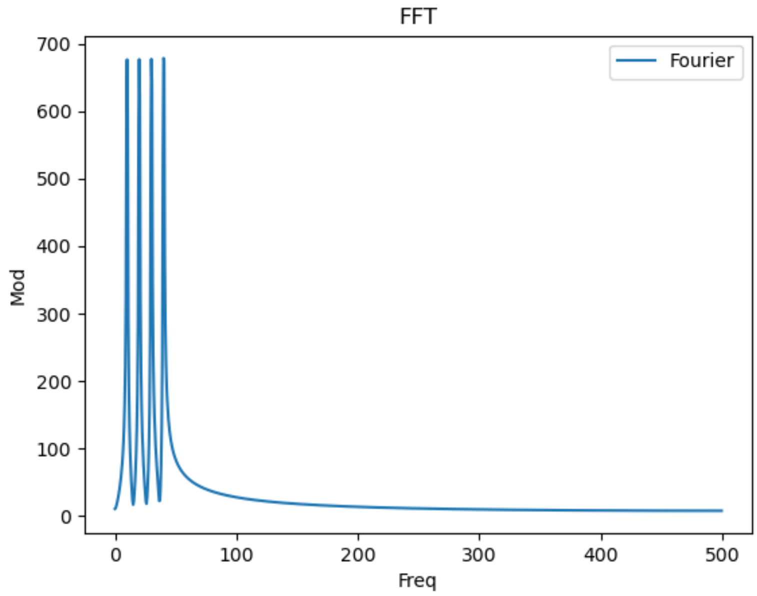

FFT
Fourier
Mod
Freq

3.1.2 $\mathbf{f}_2$

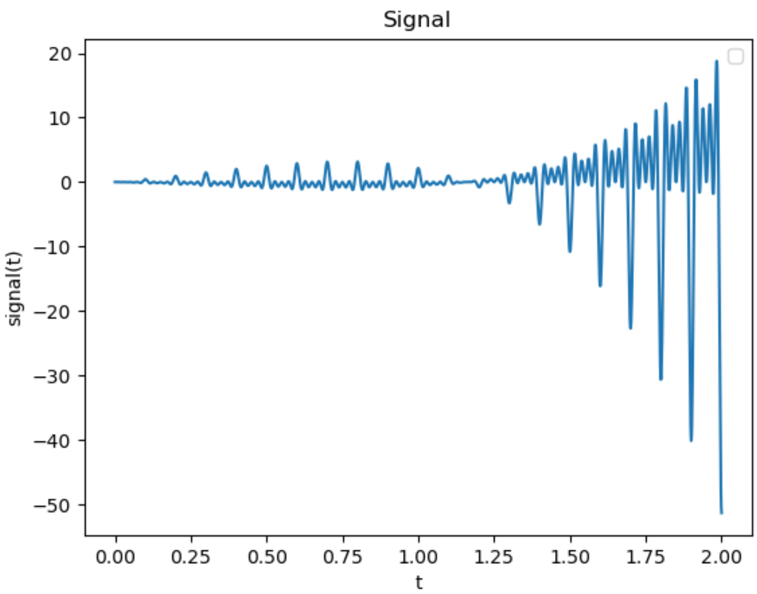


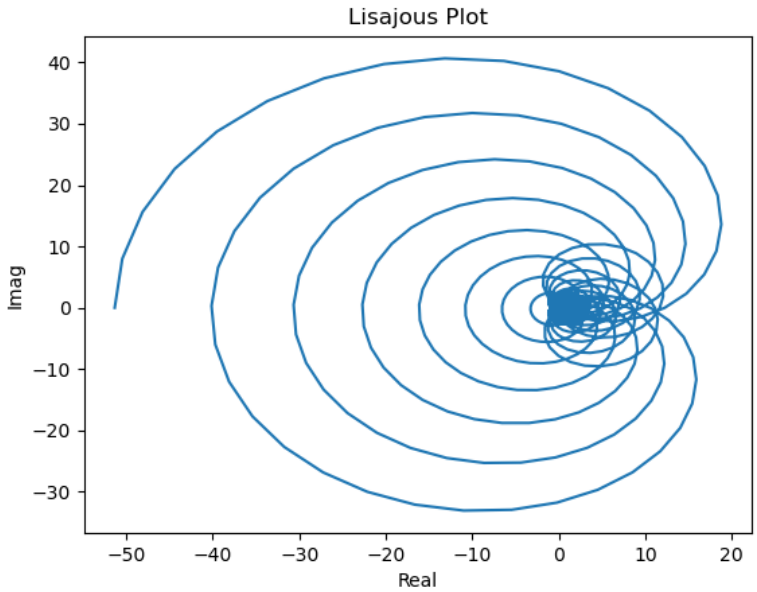

Lisajous Plot
Imag
Real
10
5
0
−5
−10
−15
−10
−5
0
5

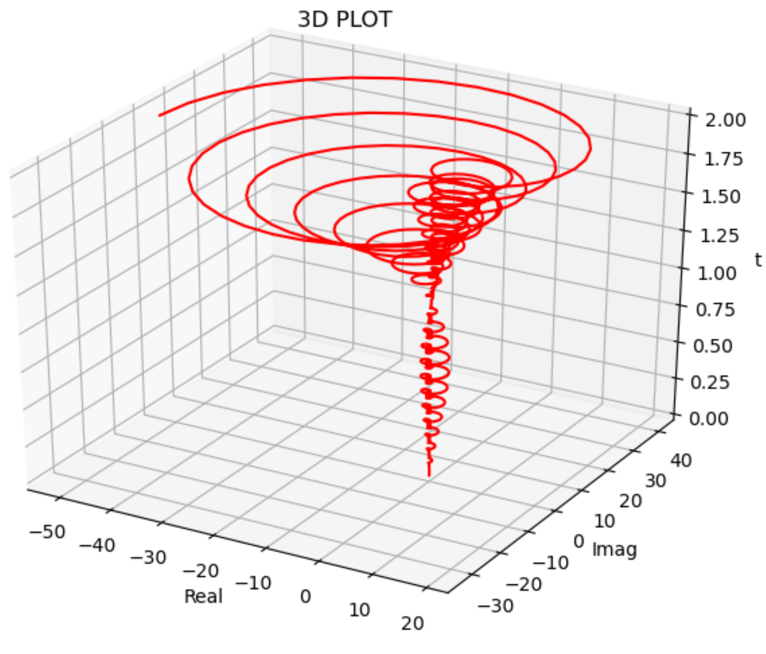

3D PLOT
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
t
10
5
0
Imag
−5
−10
−15
−10
−5
Real
0
5

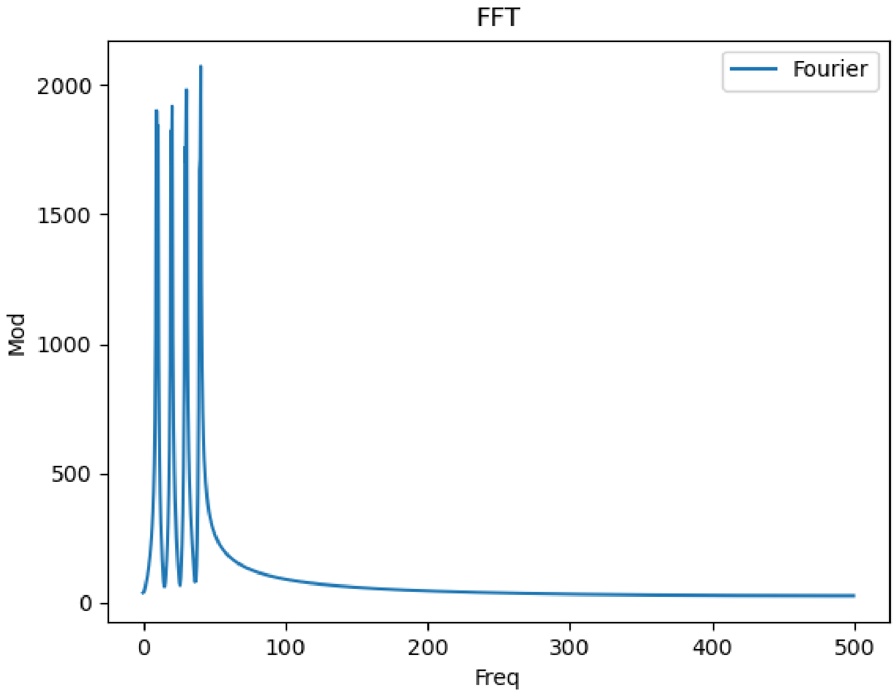

FFT
Fourier
Mod
700
600
500
400
300
200
100
0
0
100
200
300
400
500
Freq

3.1.3 f_3


Signal
signal(t)
20
10
0
−10
−20
−30
−40
−50
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
t

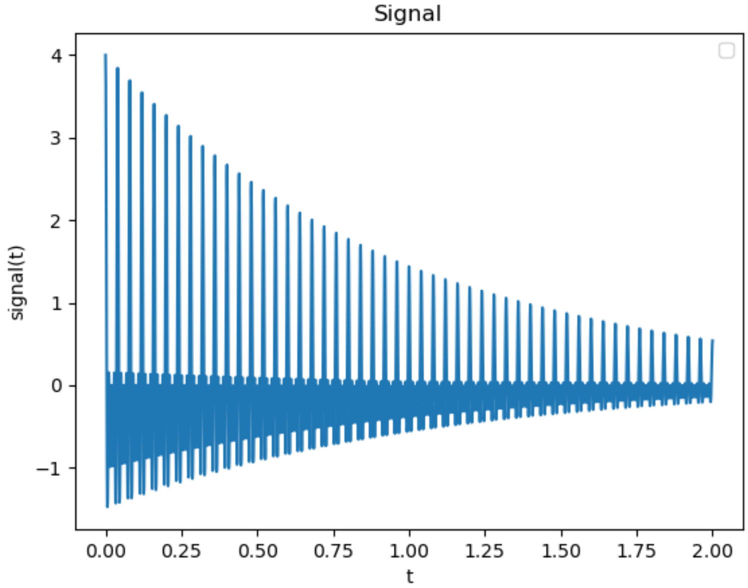

Lisajous Plot
Imag
Real
40
30
20
10
0
−10
−20
−30
−50
−40
−30
−20
−10
0
10
20

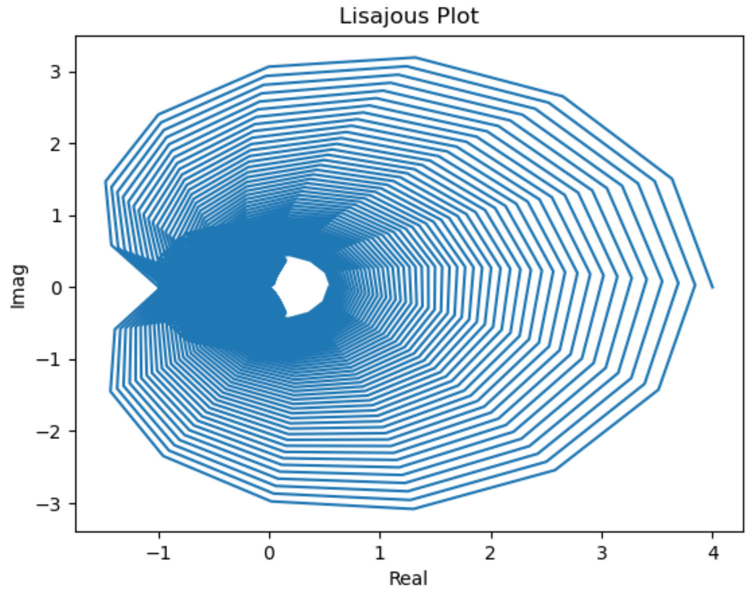

3D PLOT
Real
Imag
t
−50
−40
−30
−20
−10
0
10
20
−30
−20
−10
0
10
20
30
40
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

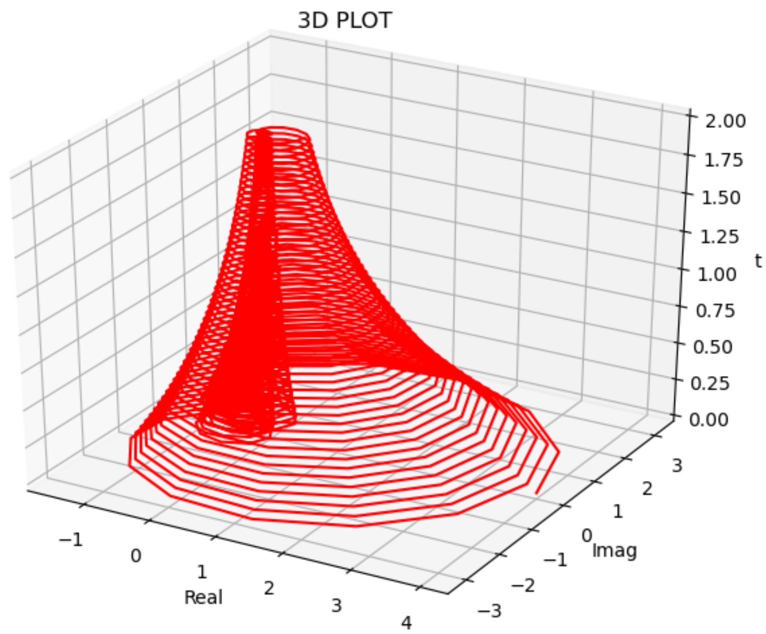


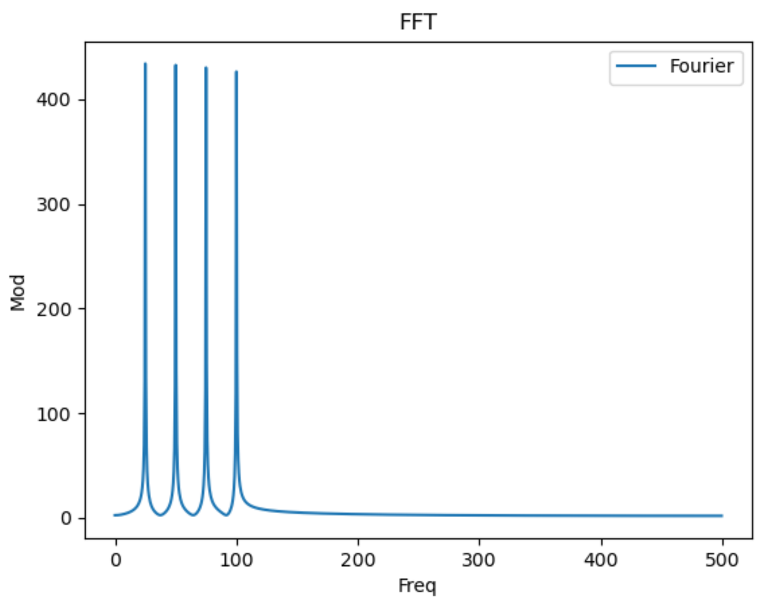
3.2 Second Signal 25Hz

3.2.1 f_1

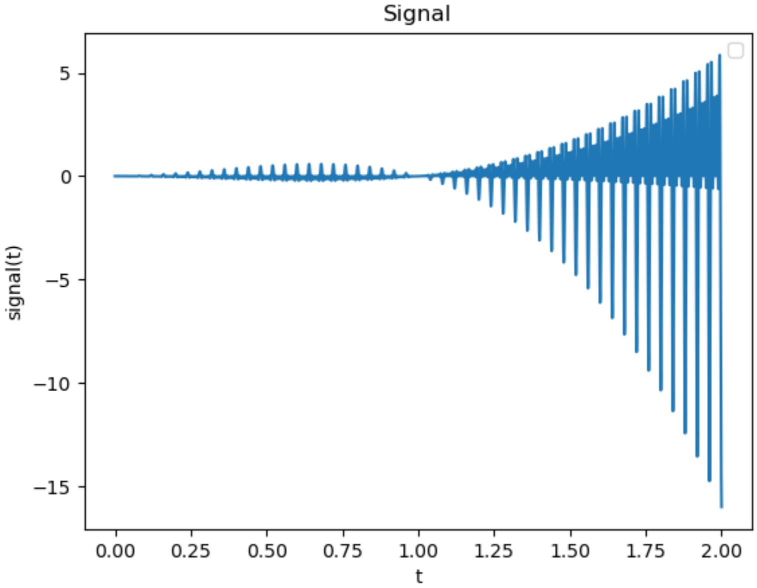

Signal
signal(t)
t
4
3
2
1
0
−1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

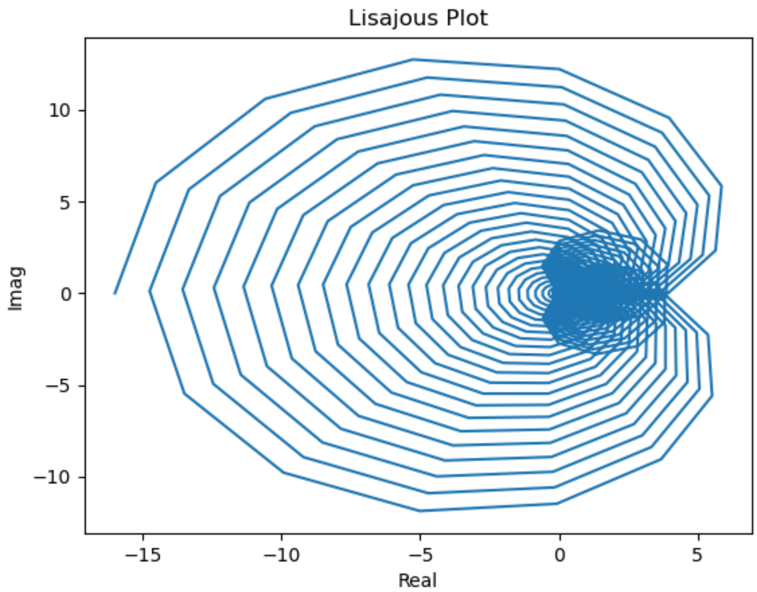

Lisajous Plot
Imag
Real
3
2
1
0
−1
−2
−3
−1
0
1
2
3
4

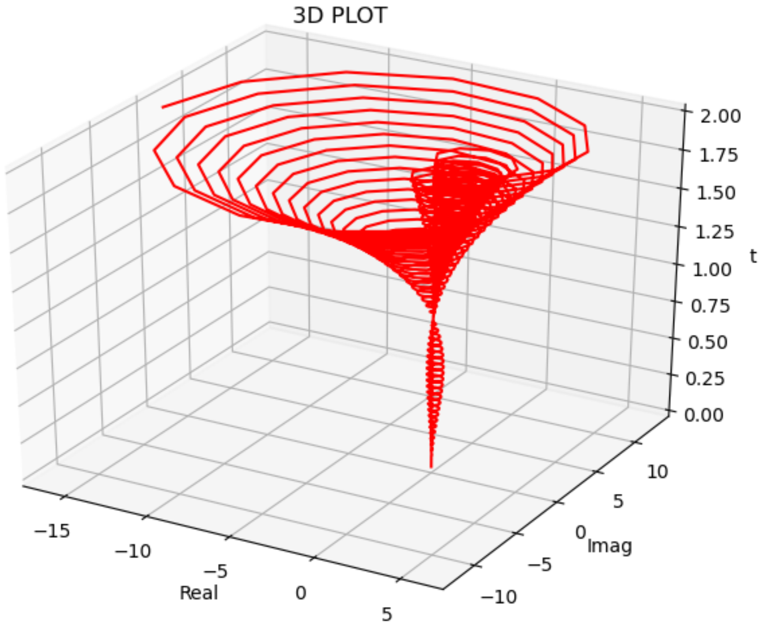

3D PLOT
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
t
−1
0
1
2
3
4
Real
3
2
1
0
−1
−2
−3
Imag

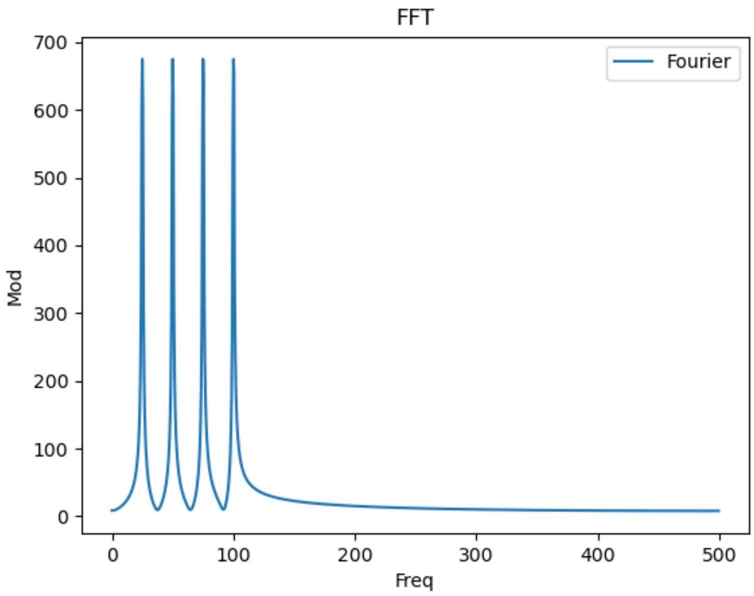

FFT
Fourier
400
300
200
100
0
Mod
0
100
200
300
400
500
Freq

3.2.2 f_2

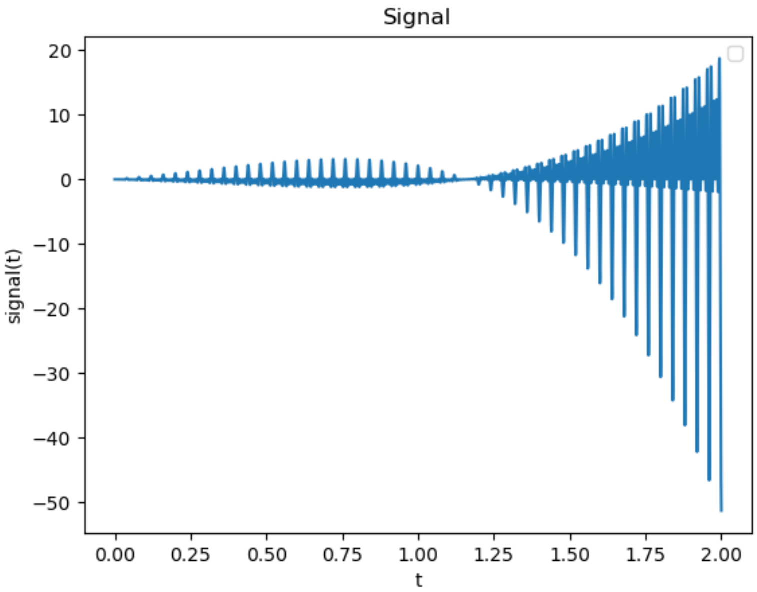

Signal
signal(t)
t
5
0
−5
−10
−15
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

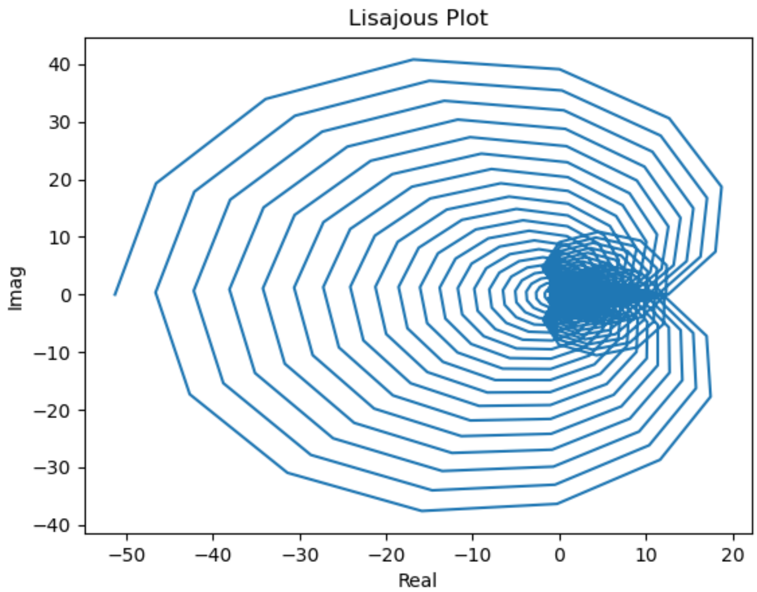

Lisajous Plot
Imag
Real
10
5
0
−5
−10
−15
−10
−5
0
5

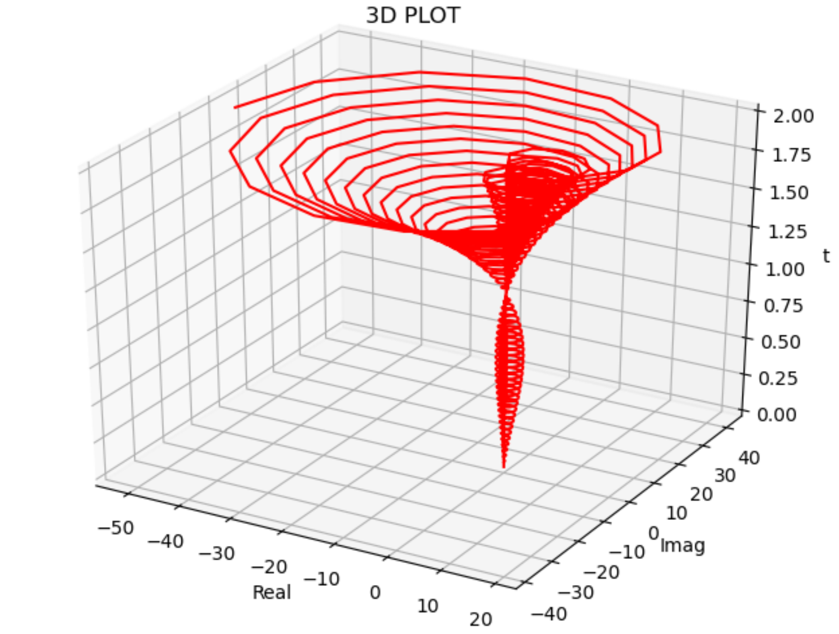

3D PLOT
t
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−15
−10
−5
0
5
Real
10
5
0
−5
−10
Imag

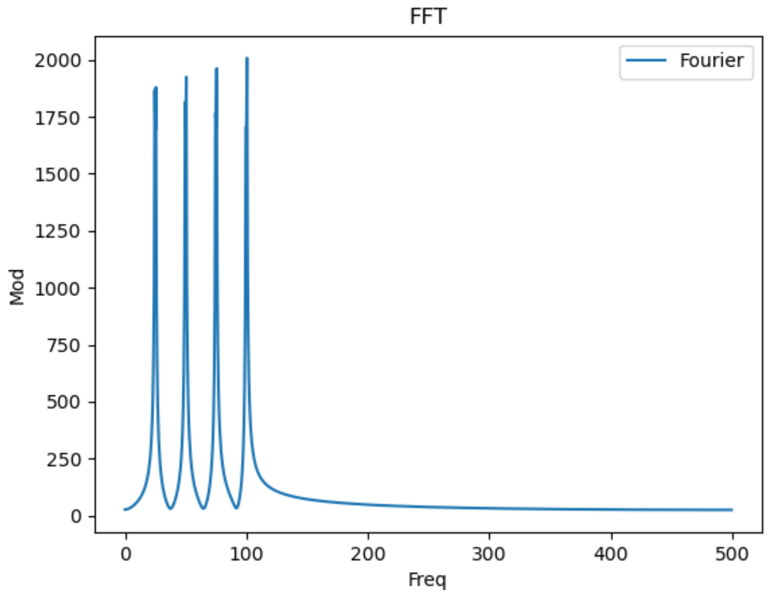

FFT
Fourier
Mod
700
600
500
400
300
200
100
0
0
100
200
300
400
500
Freq

3.2.3 f_3


Signal
signal(t)
t


Lisajous Plot
Imag
Real


3D PLOT
t
Real
Imag


FFT
Fourier
Mod
Freq

4 Signal Filters

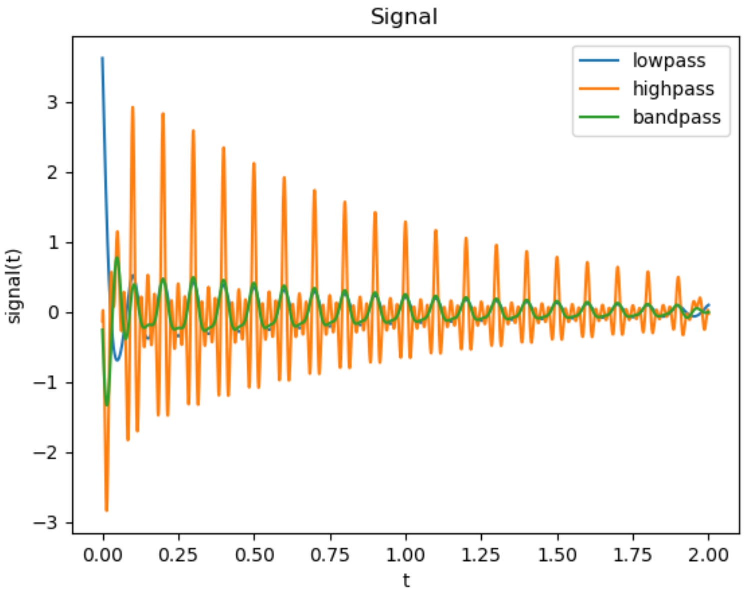
Now, having seen the signals synthesized by the sum of cosines, we can see the action of four types of filters ('lowpass', 'highpass', 'bandpass', 'Gaussian'). In this way, the code in python made to perform this task is shown below.

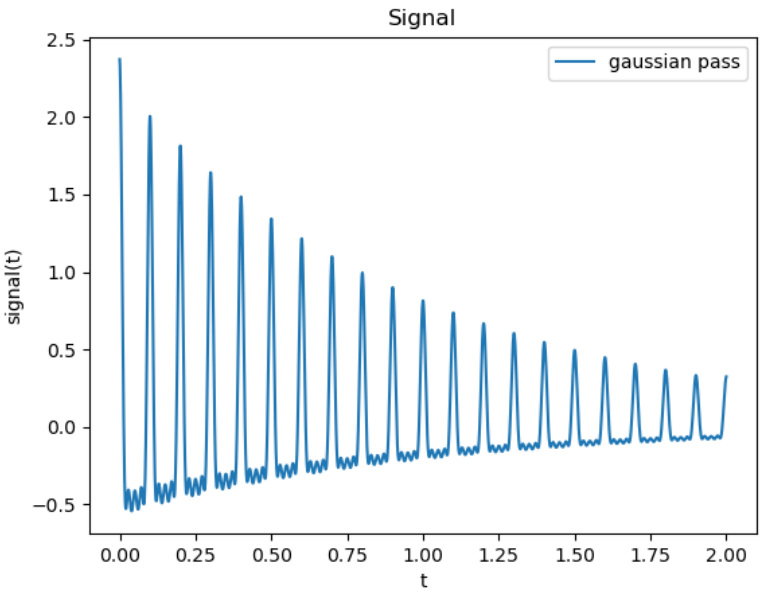
```python
def filter_lowpass(data, cutoff=20, fs=2000, order=2):
    '''
    data , array to be filtered
    cutoff, the frequency cut of the low pass
    fs, max size of the time
    order, order of the filter
    '''
    nyq=0.5*fs
    normal_cutoff=cutoff/nyq
    b, a= butter(order, normal_cutoff, btype='low')
    y=filtfilt(b, a, data)

    return y


def filter_highpass(data, cutoff=20, fs=2000, order=5):

    nyq=0.5*fs
    normal_cutoff=cutoff/nyq
    b, a = butter(order, normal_cutoff, btype='high', analog=False)
    y = filtfilt(b, a, data)

    return y
```

```python
def filter_bandpass(data, lowcut=20, highcut=35, fs=2000, order=2):

    nyq=0.5*fs
    low= lowcut/nyq
    high= highcut/nyq

    b, a = butter(order, [low, high], 'bandpass', analog=False)
    y=filtfilt(b, a, data)


    return y



def filter_gaussian(data, sigma=7):

    y=gaussian_filter(data, sigma)


    return y
```

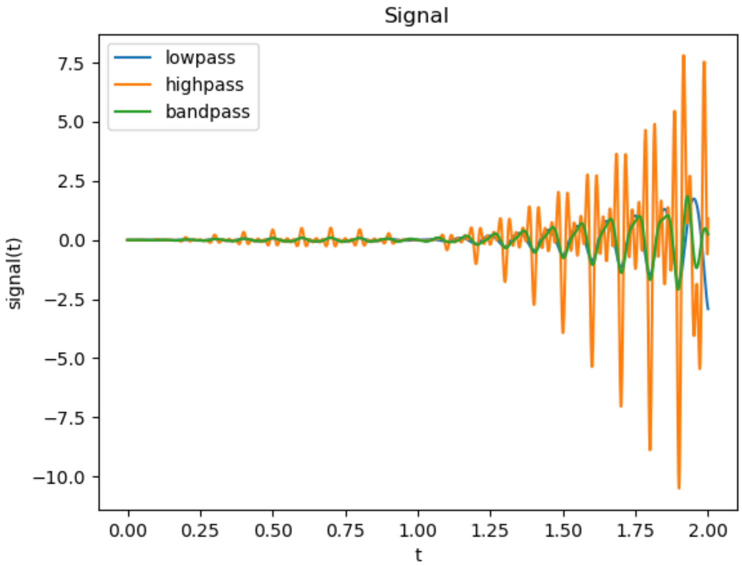
The following are the respective signal filters from the previous section.

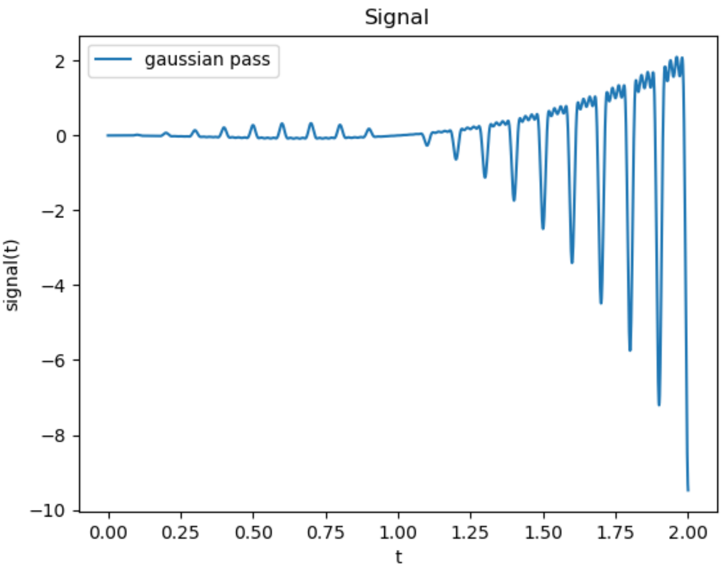
4.1 f_1


Signal
lowpass
highpass
bandpass
signal(t)
t
3
2
1
0
−1
−2
−3
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

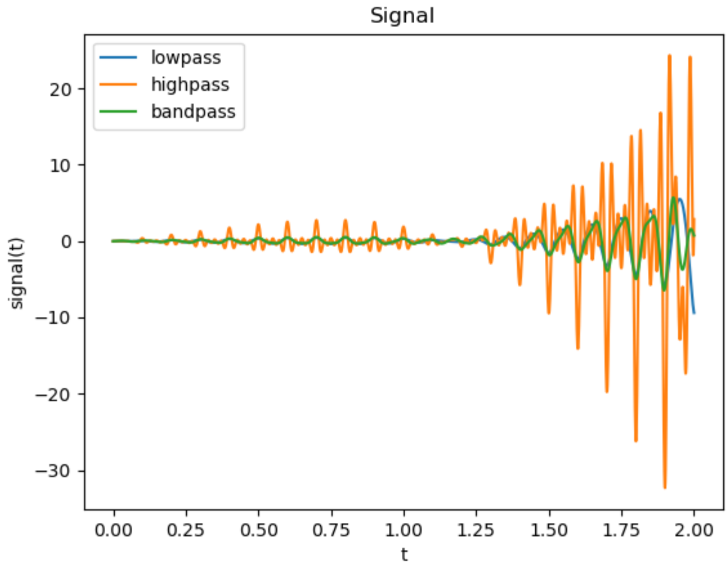

Signal
gaussian pass
signal(t)
t
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

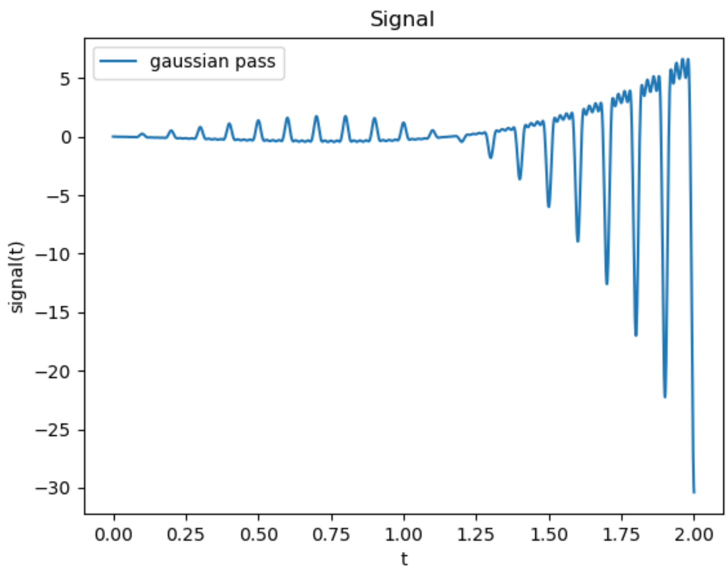
4.2 f_2


Signal
lowpass
highpass
bandpass
signal(t)
t
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

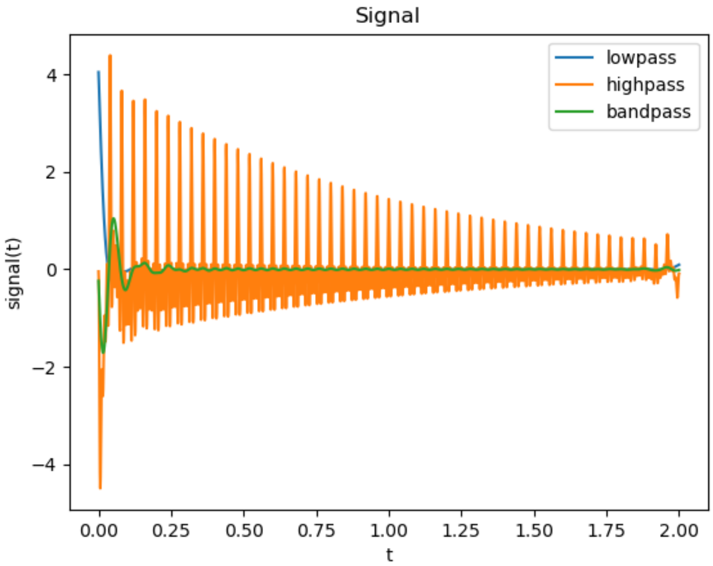

Signal
gaussian pass
signal(t)
t
2
0
−2
−4
−6
−8
−10
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

4.3 f_3

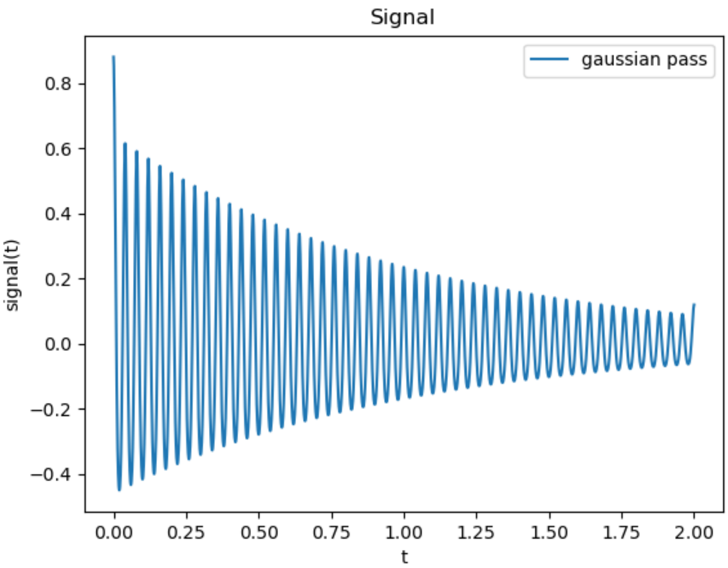

Signal
lowpass
highpass
bandpass
20
10
0
−10
−20
−30
signal(t)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
t


Signal
gaussian pass
5
0
−5
−10
−15
−20
−25
−30
signal(t)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
t

4.4 f_1

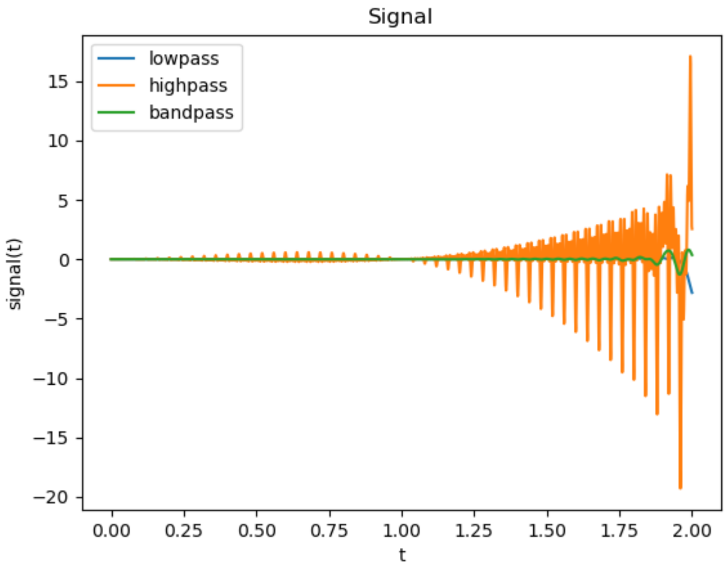


Signal
lowpass
highpass
bandpass
signal(t)
t

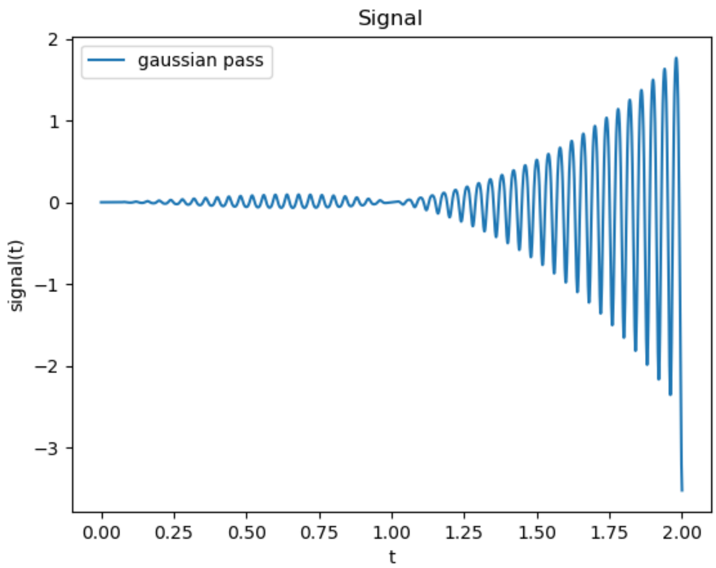


Signal
gaussian pass
signal(t)
t

4.5 f_2

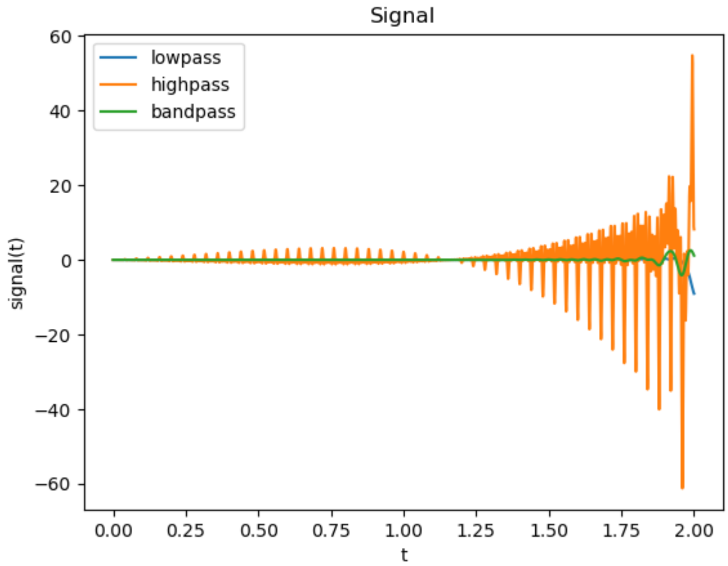


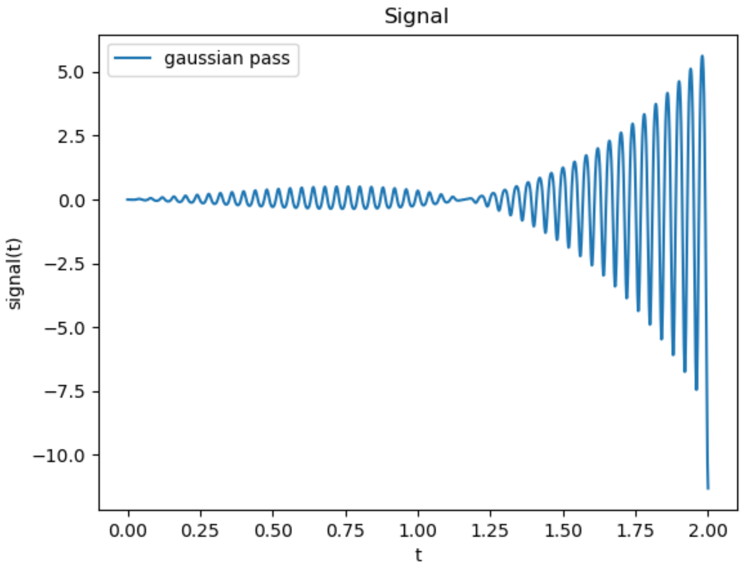
Signal
lowpass
highpass
bandpass
signal(t)
15
10
5
0
−5
−10
−15
−20
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
t



Signal
gaussian pass
signal(t)
2
1
0
−1
−2
−3
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
t

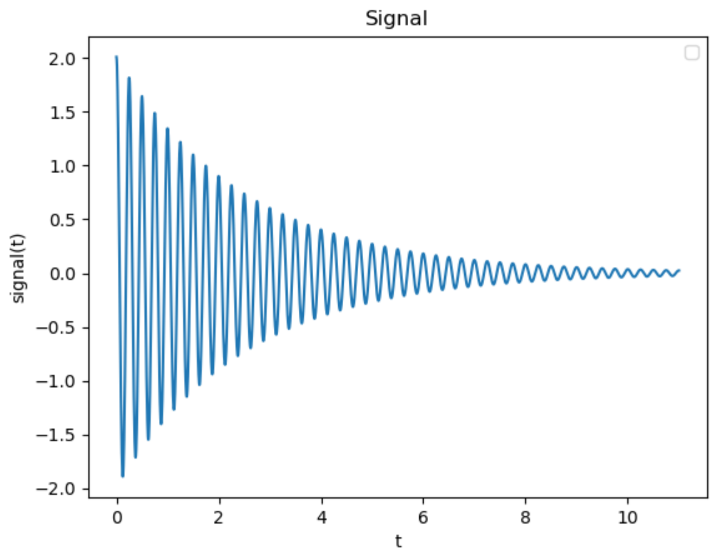
4.6 f_3

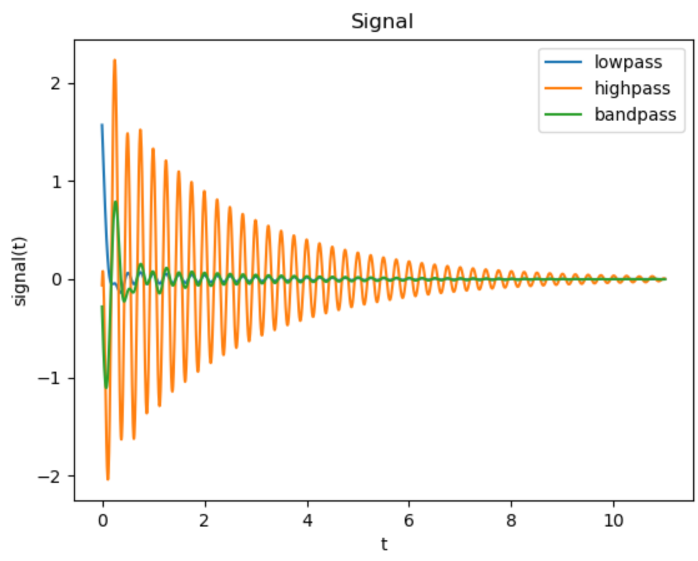

Signal
lowpass
highpass
bandpass
signal(t)
t

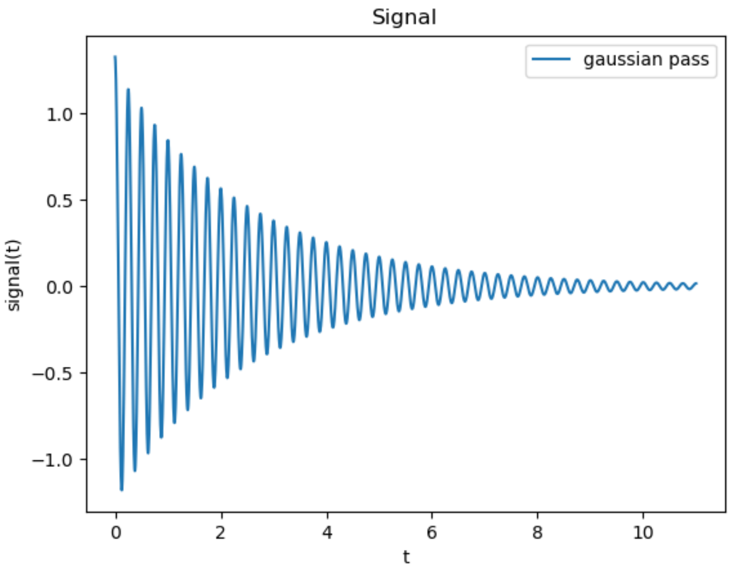

Signal
gaussian pass
signal(t)
t

5 Real Signal

Finally, we have the data of a real signal from a damped oscillator from the physics lab 2.







6 References

[1] Codes and images used in this project. https://github.com/evertonmendes