MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

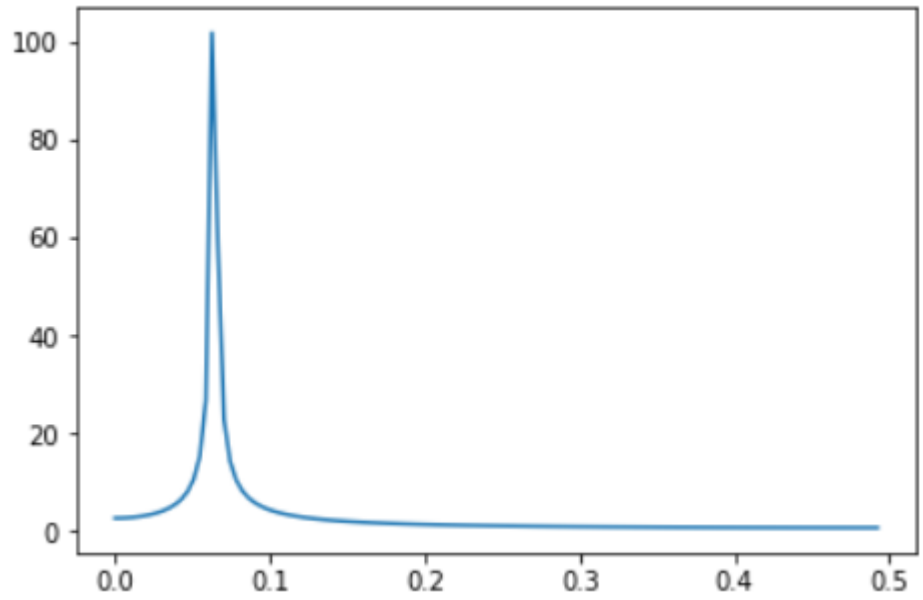
CHỦ ĐỀ 1: BIẾN ĐỔI FOURIER RỜI RẠC DFT

Mô tả: Viết hàm biến đổi DFT không sử dụng hàm FFT có sẵn trong thư viện Numpy

```
import numpy as np

import pandas as pd

%matplotlib inline

import matplotlib.pyplot as plt

import matplotlib as mpl

from scipy import fftpack

from scipy import signal
```

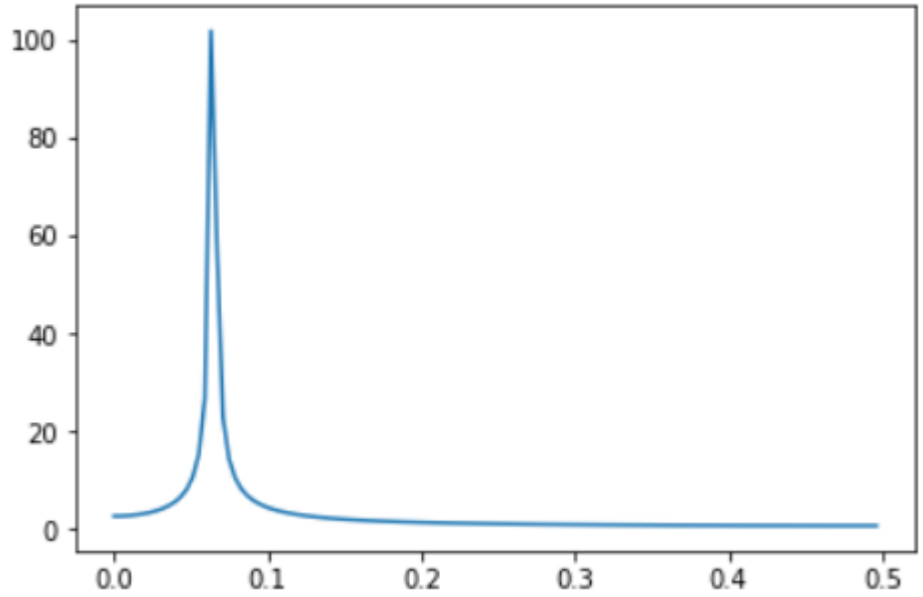
```
f_s = 1

N=256

T=N/f_s

t = np.linspace(0, T, N)

b = np.array([-6.7619496, 13.456336, -6.7619496])
```

```
x=np.asarray(x_pass)

Xk = np.fft.fft(x)

tanso = np.fft.fftfreq(N)

plt.plot(tanso[0:127], np.abs(Xk[0:127]))

plt.show()
```



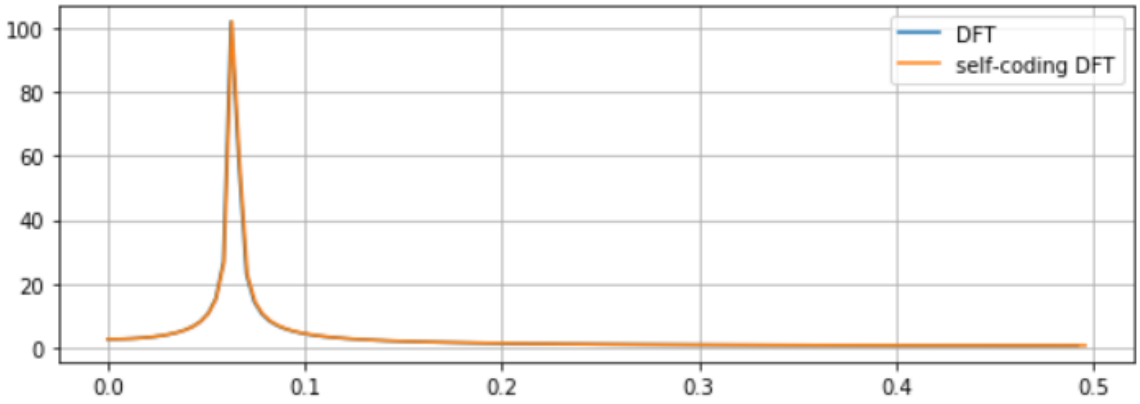
Hình 1-1 Phổ tần số sử dụng hàm FFT có sẵn trong thư viện Numpy

```
n=np.arange(N)

k=n.reshape((N, 1))

heso=np.exp(-2j * np.pi * k * n / N)

ketqua=np.dot(heso,x)

tanso2=np.linspace(0.0, 0.5, 127, endpoint=False)

plt.plot(tanso2, np.abs(ketqua[0:127]))

plt.show()
```



Hình 1-2 Phổ tần số sử dụng hàm tính toán DFT tự viết

```
fig, ax = plt.subplots(figsize=(8, 3))

plt.plot(tanso[0:127], np.abs(Xk[0:127]), label="DFT")

plt.plot(tanso2, np.abs(ketqua[0:127]), label="self-coding DFT")

ax.legend(loc=0)

fig.tight_layout()

plt.grid(True)
```



Hình 1-3 Phổ tần số sử dụng hàm FFT có sẵn trên Numpy và hàm tính toán DFT tự viết cho kết quả hoàn toàn tương đương

CHỦ ĐỀ 2: BỘ LỌC SỐ

2.1 Bộ lọc thông thấp

Giả thiết tín hiệu x(n) là tổng của 2 tín hiệu x1(n) và x2(n). x1(n) là tín hiệu cosin có tần số góc là 0,1rad/s, x2(n) cũng là tín hiệu cosin có tần số góc là 0,4rad/s. Người ta dùng bộ lọc thông thấp FIR có độ dài đáp ứng xung bằng 3 với giả thiết h(0) = h(2) = α và h(1) = β để triệt tiêu tín hiệu x2(n) và cho qua hoàn toàn tín hiệu x1(n). Hãy xác định các hệ số a, b và vẽ sơ đồ khối thực hiện bộ lọc FIR này.

Sau khi đã tính được các hệ số a, b, lập trình Python thực hiện bộ lọc FIR để lọc tín hiệu x(n) như trên (vẽ dạng tín hiệu x(n), phân tích phổ x(n) trước khi lọc và sau khi lọc.

Bài giải:

$H(e^{j\omega})=h[0)]+h[1]\ e^{-j\omega}+ h[2]\ e^{-j2\omega}=\alpha(1+ e^{-j2\omega})+ \beta e^{-j\omega}= \alpha(e^{j\omega} + e^{-j\omega})\ e^{-j\omega}+ \beta e^{-j\omega}$

$=(2\alpha\cos\omega+\beta)e^{-j\omega}$

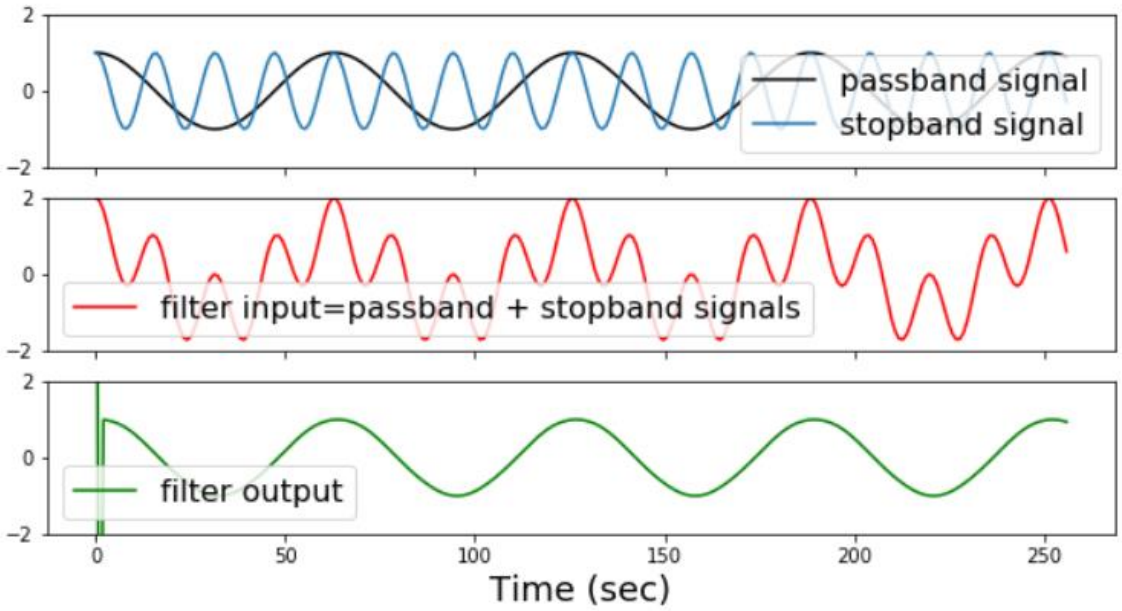
$| H(e^{j0.1})|=1;\ | H(e^{j0.4})|=0;<=> \alpha=-6.76195,\ \beta=12.45634;$

Mô tả: Biểu diễn đáp ứng tần số của bộ lọc thông thấp
import numpy as np import pandas as pd %matplotlib inline import matplotlib.pyplot as plt import matplotlib as mpl from scipy import fftpack from scipy import signal

```
f_s = 1

N=256

T=N/f_s

t = np.linspace(0, T, N)
```

```
b = np.array([6.76195, -12.45634, 6.76195])

x_pass = np.cos(0.1*t)

x_stop = np.cos(0.4*t)

x = x_pass + x_stop

y=signal.lfilter(b,1,x)

fig,axs = plt.subplots(3,1,sharey=True,sharex=True)

fig.set_size_inches((10,5))

ax=axs[0]

ax.plot(t,x_pass,label="passband signal",color="k")

ax.plot(t,x_stop,label="stopband signal")

ax.set_ylim([-2, 2])

ax.legend(loc=0,fontsize=16)

ax=axs[1]

ax.plot(t,x,label="filter input=passband + stopband signals",color="r")

ax.legend(loc=0,fontsize=16)

ax=axs[2]
```

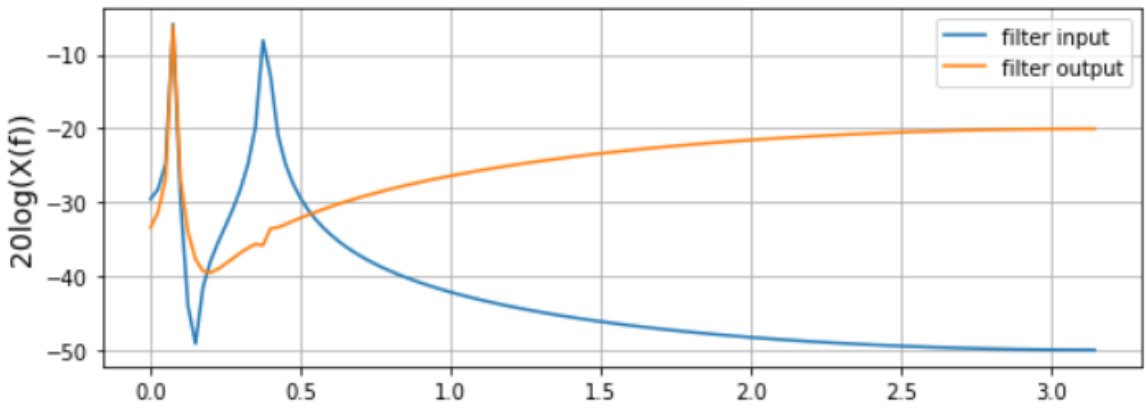
```
ax.plot(t,y,label="filter output",color="g")

ax.set_xlabel("Time (sec)",fontsize=18)

ax.legend(loc=0,fontsize=16);
```



Hình 2-1 Tín hiệu đầu vào và tín hiệu đầu ra bộ lọc thông thấp

```
f = fftpack.fftfreq(N, 1/f_s)

mask = np.where(f > 0)

X = fftpack.fft(x)

Y = fftpack.fft(y)

X_mask=20 * np.log10(abs(X[mask])/N)

Y_mask=20 * np.log10(abs(Y[mask])/N)

fig, ax = plt.subplots(figsize=(8, 3))

plt.plot(np.linspace(0, np.pi, len(X_mask)), X_mask, label="filter input")

plt.plot(np.linspace(0, np.pi, len(Y_mask)), Y_mask, label="filter output")
```

```
ax.set_ylabel("20log(X(f))", fontsize=14)

ax.legend(loc=0)

fig.tight_layout()

plt.grid(True)
```



Hình 2-2 Đáp ứng tần số của bộ lọc thông thấp

2.2 Bộ lọc thông cao

Giả thiết tín hiệu x(n) là tổng của 2 tín hiệu x1(n) và x2(n). x1(n) là tín hiệu cosin có tần số góc là 0,1rad/s, x2(n) cũng là tín hiệu cosin có tần số góc là 0,4rad/s. Người ta dùng bộ lọc thông cao FIR có độ dài đáp ứng xung bằng 3 với giả thiết h(0) = h(2) = α và h(1) = β để triệt tiêu tín hiệu x1(n) và cho qua hoàn toàn tín hiệu x2(n). Hãy xác định các hệ số a, b và vẽ sơ đồ khối thực hiện bộ lọc FIR này.

Sau khi đã tính được các hệ số a, b, lập trình Python thực hiện bộ lọc FIR để lọc tín hiệu x(n) như trên (vẽ dạng tín hiệu x(n), phân tích phổ x(n) trước khi lọc và sau khi lọc.

Bài giải:

$H(e^{j\omega})=h[0)]+h[1]\ e^{-j\omega}+ h[2]\ e^{-j2\omega}=\alpha(1+ e^{-j2\omega})+ \beta e^{-j\omega}= \alpha(e^{j\omega} + e^{-j\omega})\ e^{-j\omega}+ \beta e^{-j\omega}$

$=(2\alpha\cos\omega+\beta)e^{-j\omega}$

$|\ H(e^{j0.1})|=0;\ |\ H(e^{j0.4})|=1;<=>\ \alpha=-6,76195,\ \beta=13,45634;$

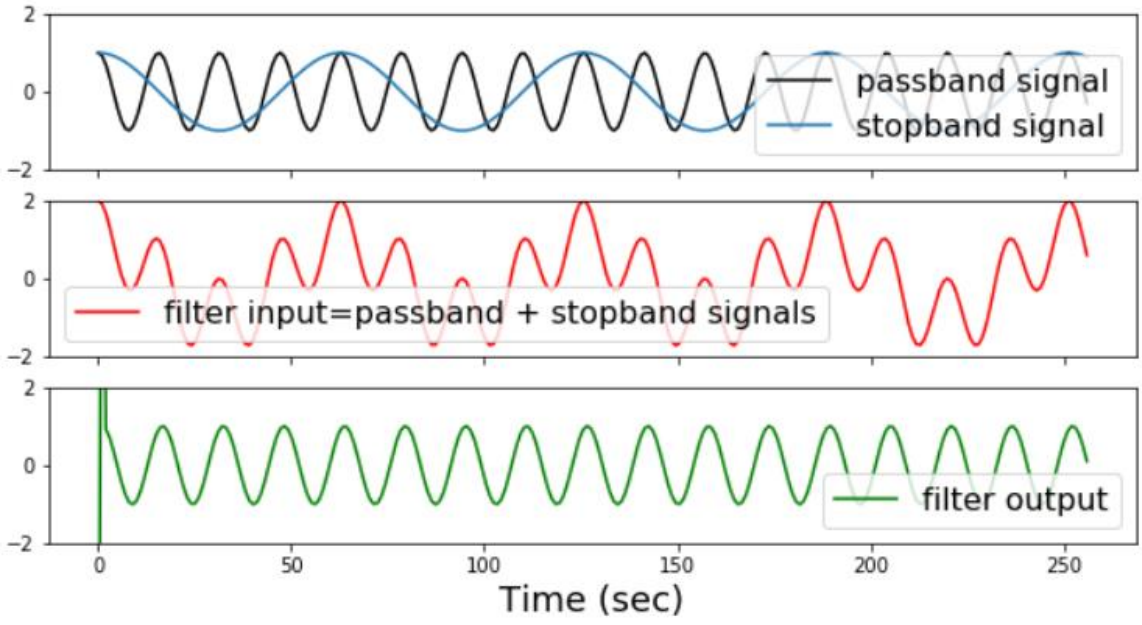
Mô tả: Biểu diễn đáp ứng tần số của bộ lọc thông cao

```
import numpy as np

import pandas as pd

%matplotlib inline

import matplotlib.pyplot as plt

import matplotlib as mpl

from scipy import fftpack

from scipy import signal
```

```
f_s = 1

N=256

T=N/f_s

t = np.linspace(0, T, N)
```

```
b = np.array([-6.76195, 13.45634, -6.76195])

x_stop = np.cos(0.1*t)

x_pass = np.cos(0.4*t)

x = x_pass + x_stop

y=signal.lfilter(b,1,x)

fig,axs = plt.subplots(3,1,sharey=True,sharex=True)

fig.set_size_inches((10,5))

ax=axs[0]

ax.plot(t,x_pass,label="passband signal",color="k")
```

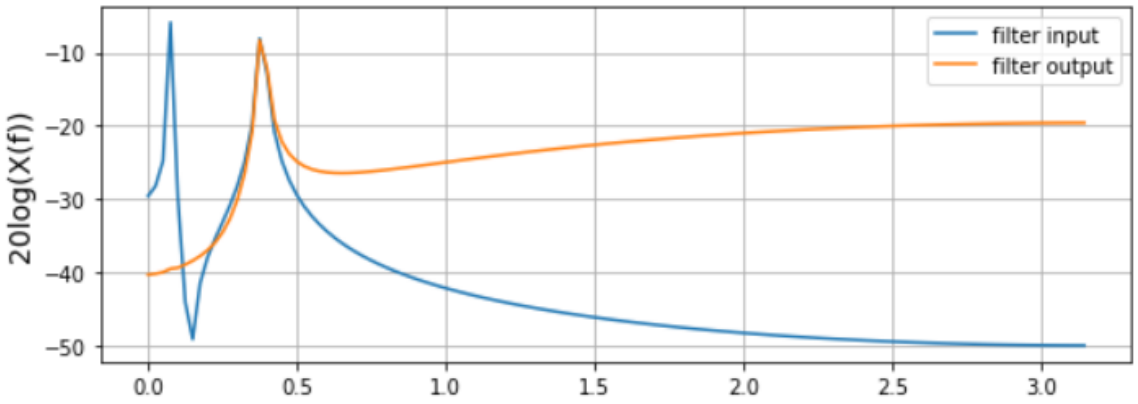
```
ax.plot(t,x_stop,label="stopband signal")

ax.set_ylim([-2, 2])

ax.legend(loc=0,fontsize=16)

ax=axs[1]

ax.plot(t,x,label="filter input=passband + stopband signals",color="r")

ax.legend(loc=0,fontsize=16)

ax=axs[2]

ax.plot(t,y,label="filter output",color="g")

ax.set_xlabel("Time (sec)",fontsize=18)

ax.legend(loc=0,fontsize=16);
```



Hình 2-3 Tín hiệu đầu vào và tín hiệu đầu ra bộ lọc thông cao

```
f = fftpack.fftfreq(N, 1/f_s)

mask = np.where(f > 0)
```

```
X = fftpack.fft(x)

Y = fftpack.fft(y)

X_mask=20 * np.log10(abs(X[mask])/N)

Y_mask=20 * np.log10(abs(Y[mask])/N)

fig, ax = plt.subplots(figsize=(8, 3))

plt.plot(np.linspace(0, np.pi, len(X_mask)), X_mask, label="filter input")

plt.plot(np.linspace(0, np.pi, len(Y_mask)), Y_mask, label="filter output")

ax.set_ylabel("20log(X(f))", fontsize=14)

ax.legend(loc=0)

fig.tight_layout()

plt.grid(True)
```

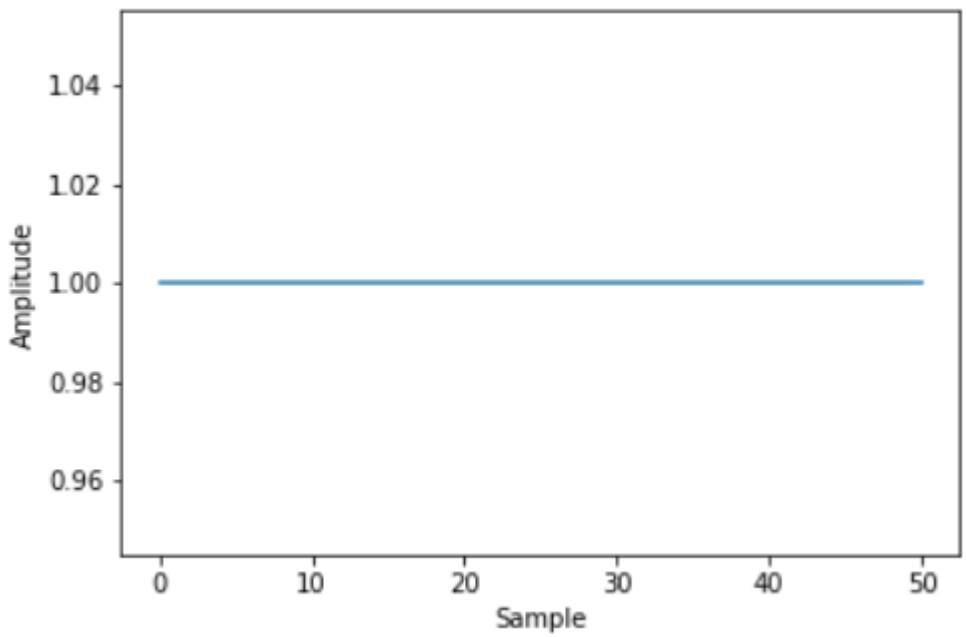


Hình 2-4 Đáp ứng tần số của bộ lọc thông cao

CHỦ ĐỀ 3: CỬA SỔ HAMMING

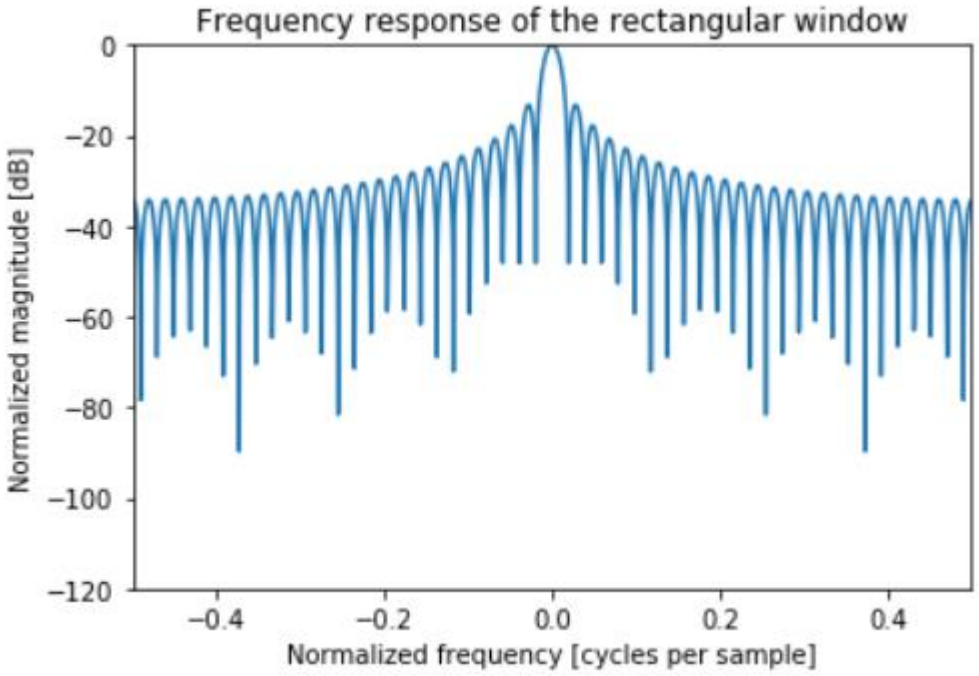
3.1 Đáp ứng tần số của cửa sổ Hamming

Mô tả: Biểu diễn đáp ứng tần số của cửa sổ hình chữ nhật và cửa sổ Hamming
import numpy as np import pandas as pd %matplotlib inline import matplotlib.pyplot as plt import matplotlib as mpl from scipy import fftpack from scipy import signal from scipy.fftpack import fft, fftshift
window_rect = signal.boxcar(51) plt.plot(window_rect) plt.ylabel("Amplitude") plt.xlabel("Sample")



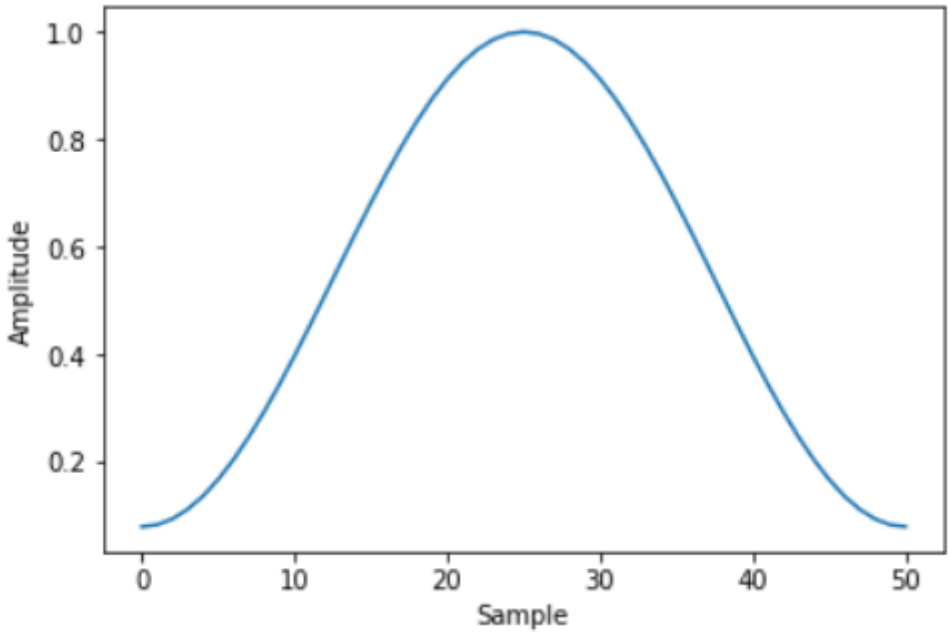
Hình 3-1 Cửa sổ hình chữ nhật

```
plt.figure()

A = fft(window_rect, 2048) / (len(window_rect)/2.0)

freq = np.linspace(-0.5, 0.5, len(A))

response = 20 * np.log10(np.abs(fftshift(A / abs(A).max())))

plt.plot(freq, response)

plt.axis([-0.5, 0.5, -120, 0])

plt.title("Frequency response of the rectangular window")

plt.ylabel("Normalized magnitude [dB]")

plt.xlabel("Normalized frequency [cycles per sample]")
```



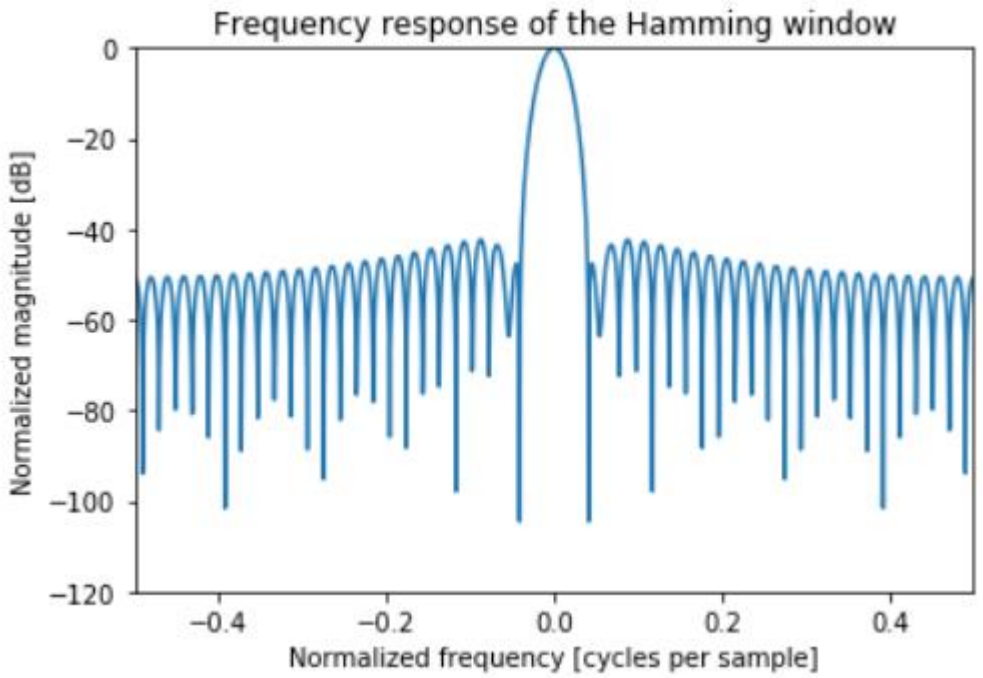
Hình 3-2 Đáp ứng tần số của cửa sổ hình chữ nhật

```
window_Hamming = signal.hamming(51)

plt.plot(window_Hamming)

plt.ylabel("Amplitude")

plt.xlabel("Sample")
```



Hình 3-3 Cửa sổ Hamming

```
plt.figure()

A = fft(window_Hamming, 2048) / (len(window_Hamming)/2.0)

freq = np.linspace(-0.5, 0.5, len(A))

response = 20 * np.log10(np.abs(fftshift(A / abs(A).max())))

plt.plot(freq, response)

plt.axis([-0.5, 0.5, -120, 0])

plt.title("Frequency response of the Hamming window")

plt.ylabel("Normalized magnitude [dB]")

plt.xlabel("Normalized frequency [cycles per sample]")
```



Hình 3-4 Đáp ứng tần số của cửa sổ Hamming

3.2 Tín hiệu đơn nhân với cửa sổ Hamming

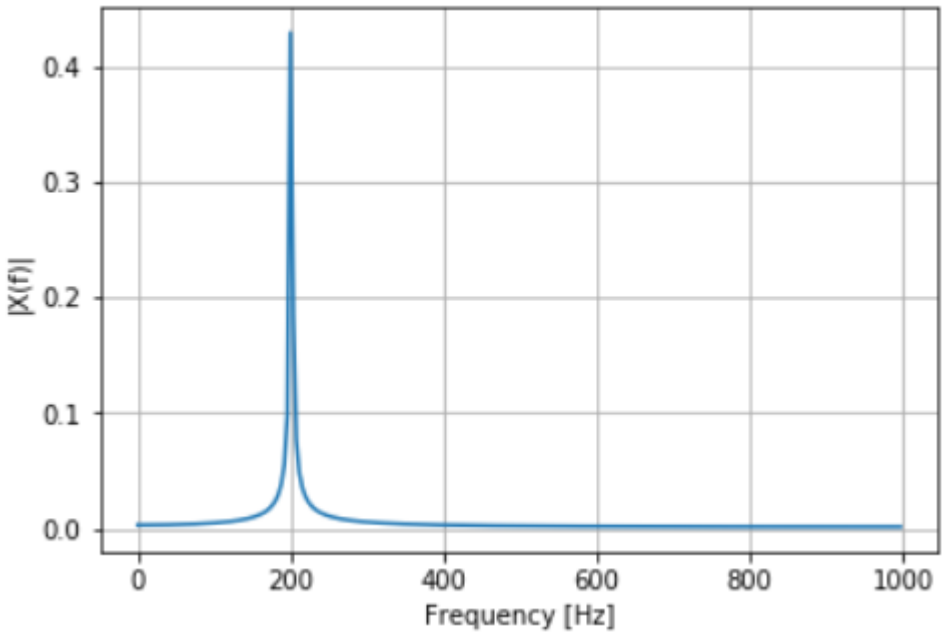
Mô tả: Biểu diễn tác động của cửa sổ Hamming lên phổ tần số của tín hiệu đơn
import numpy as np import pandas as pd %matplotlib inline import matplotlib.pyplot as plt import matplotlib as mpl from scipy import fftpack from scipy import signal
def signal_samples(t):   return np.sin(200 * 2 * np.pi * t)

```
f_s = 2000

N=512

T=N/f_s
```

```
t = np.linspace(0, T, N)

f_t = signal_samples(t)
```

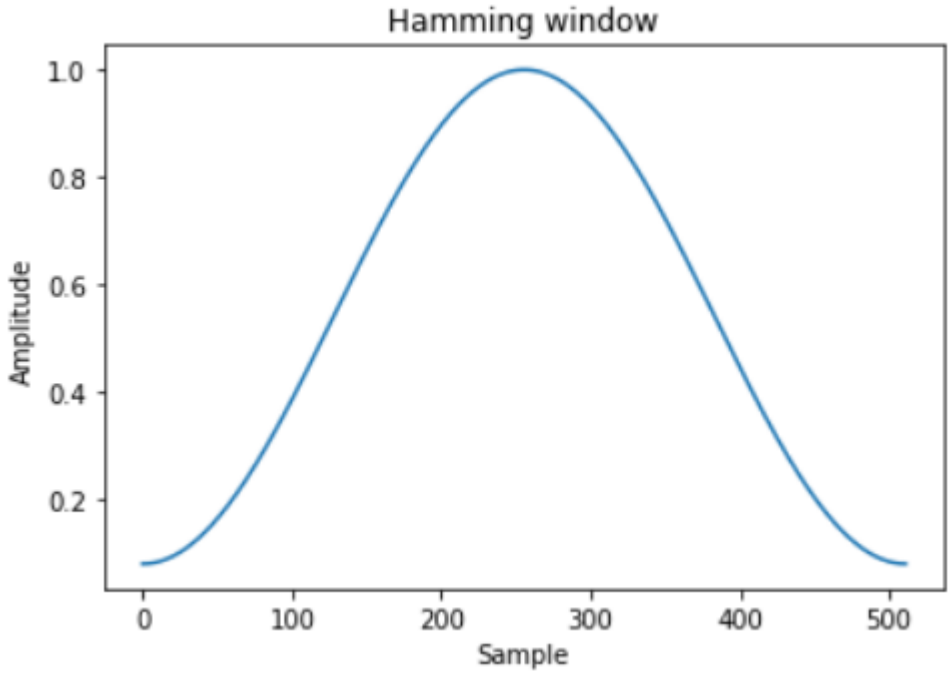
```
plt.plot(t, f_t)

plt.xlabel('Time [s]')

plt.ylabel('Amplitude')

plt.grid(True)
```

```
F = fftpack.fft(f_t)

f = fftpack.fftfreq(N, 1/f_s)

mask = np.where(f >= 0)

plt.plot(f[mask], abs(F[mask])/N, label="real")

plt.xlabel('Frequency [Hz]')

plt.ylabel('|X(f)|')

plt.grid(True)
```



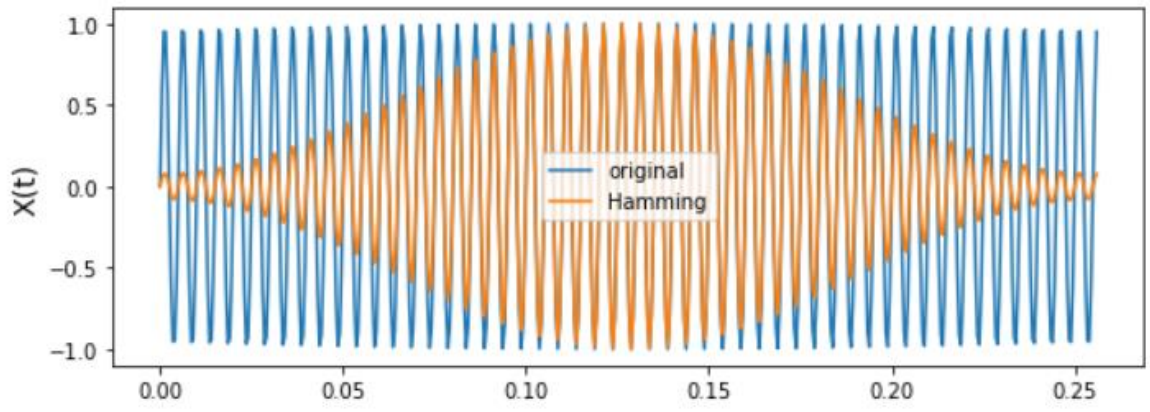
Hình 3-5 Phổ tần số của tín hiệu đơn

```
window = signal.hamming(N)

plt.plot(window)

plt.title("Hamming window")

plt.ylabel("Amplitude")

plt.xlabel("Sample")
```



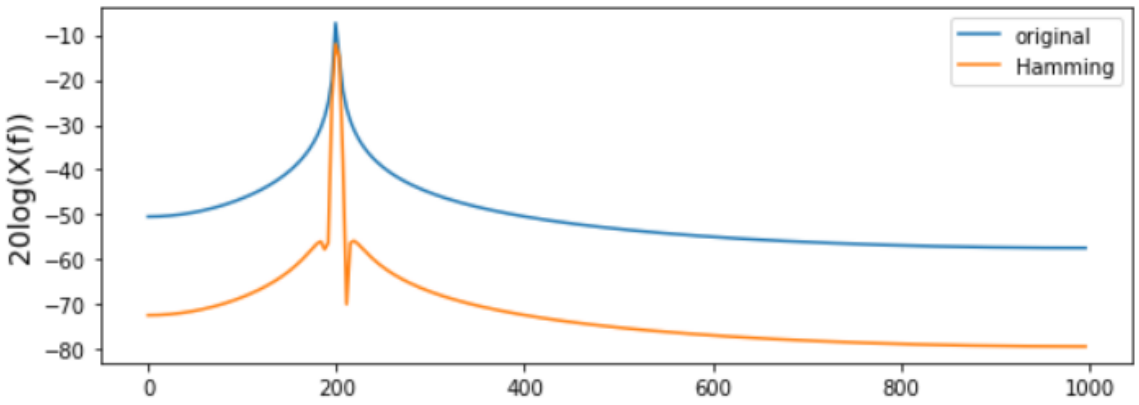
Hình 3-6 Cửa sổ Hamming có độ dài 512

```
fig, ax = plt.subplots(figsize=(8, 3))

ax.plot(t, f_t, label="original")

ax.plot(t, f_t * window, label="Hamming")

ax.set_ylabel("X(t)", fontsize=14)

ax.legend(loc=0)

fig.tight_layout()
```



Hình 3-7 Tín hiệu đơn trước và sau khi nhân với cửa sổ Hamming

```
data_fft_window = fftpack.fft(f_t * window)

data_fft = fftpack.fft(f_t)

F = fftpack.fft(f_t)

f = fftpack.fftfreq(N, 1/f_s)

mask = np.where(f >= 0)

fig, ax = plt.subplots(figsize=(8, 3))

plt.plot(f[mask], 20 * np.log10(abs(data_fft[mask])/N), label="original")

plt.plot(f[mask], 20 * np.log10(abs(data_fft_window[mask])/N),
label="Hamming")

ax.set_ylabel("20log(X(f))", fontsize=14)

ax.legend(loc=0)

fig.tight_layout()
```



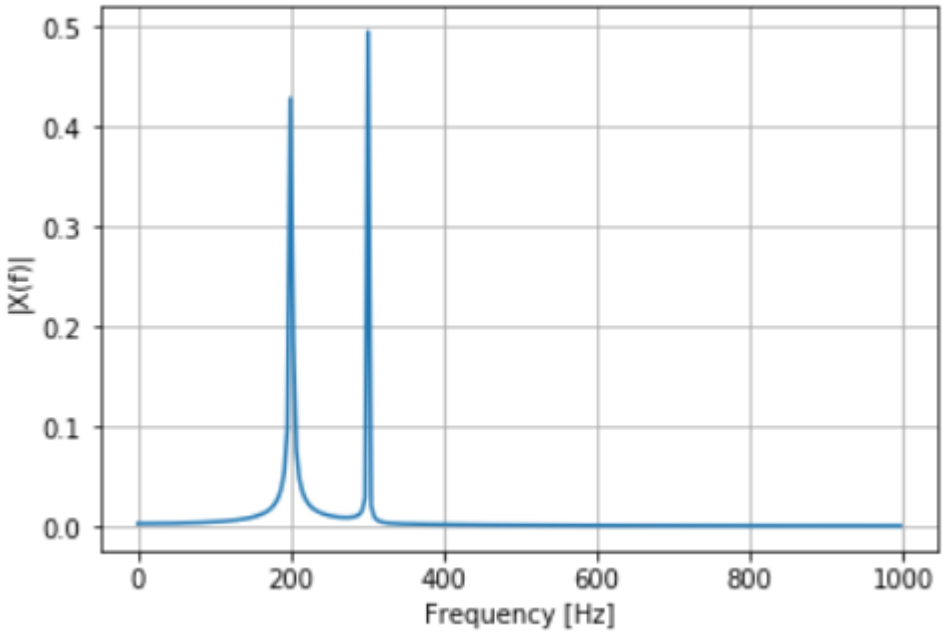
Hình 3-8: Phổ tần số của tín hiệu đơn và tín hiệu đơn qua cửa sổ Hamming

3.3 Tín hiệu tổng hợp nhân với cửa sổ Hamming

Mô tả: Biểu diễn tác động của cửa sổ Hamming lên phổ tần số của tín hiệu tổng hợp
import numpy as np import pandas as pd %matplotlib inline import matplotlib.pyplot as plt import matplotlib as mpl from scipy import fftpack from scipy import signal
def signal_samples(t):   return np.sin(200 * 2 * np.pi * t)+np.sin(300 * 2 * np.pi * t)
f_s = 2000 N=512 T=N/f_s

```
t = np.linspace(0, T, N)

f_t = signal_samples(t)
```

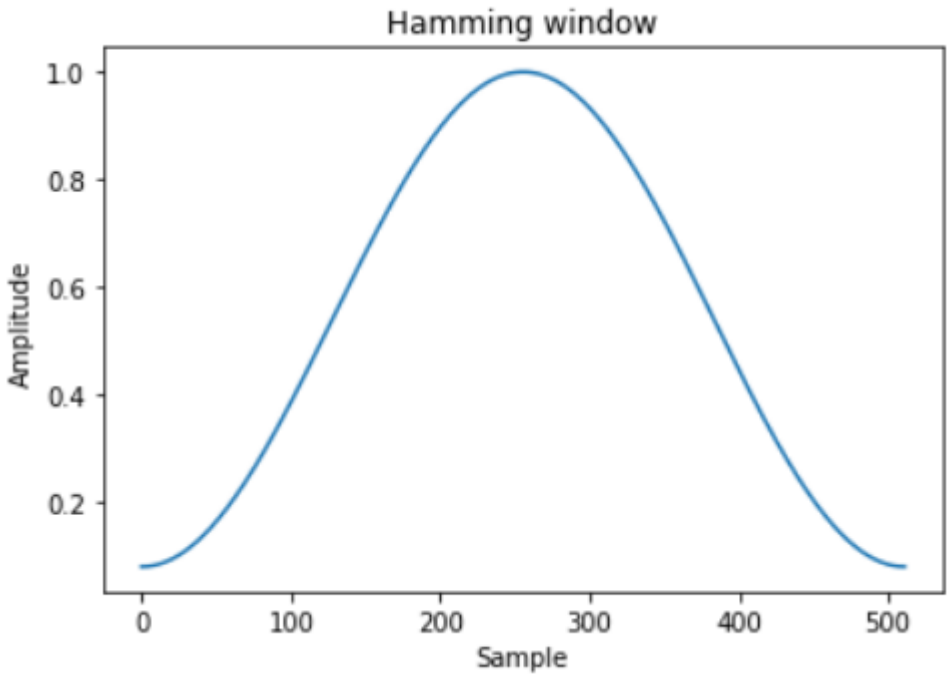
```
F = fftpack.fft(f_t)

f = fftpack.fftfreq(N, 1/f_s)

mask = np.where(f >= 0)

plt.plot(f[mask], abs(F[mask])/N, label="real")

plt.xlabel('Frequency [Hz]')

plt.ylabel('|X(f)|')

plt.grid(True)
```



Hình 3-9 Phổ tần số của tín hiệu tổng hợp

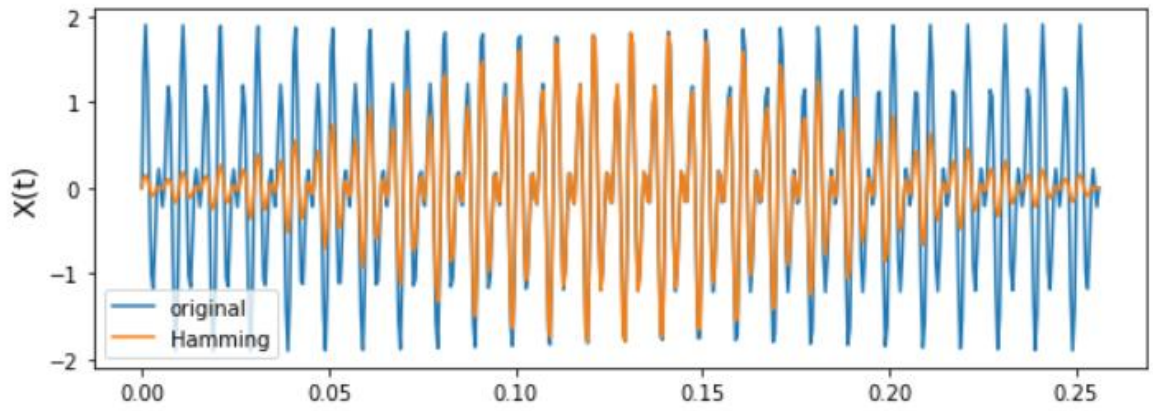
```
window = signal.hamming(N)

plt.plot(window)

plt.title("Hamming window")
```

```
plt.ylabel("Amplitude")

plt.xlabel("Sample")
```



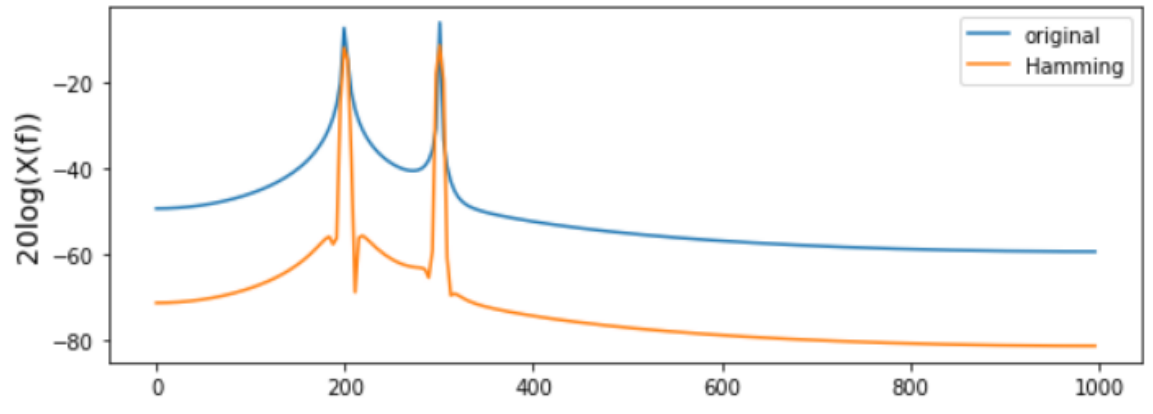
Hình 3-10 Cửa sổ Hamming có độ dài 512

```
f_t_window = f_t * window

fig, ax = plt.subplots(figsize=(8, 3))

ax.plot(t, f_t, label="original")

ax.plot(t, f_t_window, label="Hamming")

ax.set_ylabel("X(t)", fontsize=14)

ax.legend(loc=0)

fig.tight_layout()
```



Hình 3-11 Tín hiệu tổng hợp trước và sau khi nhân với cửa sổ Hamming

```
data_fft = fftpack.fft(f_t)

data_fft_window = fftpack.fft(f_t_window)

fig, ax = plt.subplots(figsize=(8, 3))

plt.plot(f[mask], 20 * np.log10(abs(data_fft[mask])/N), label="original")

plt.plot(f[mask], 20 * np.log10(abs(data_fft_window[mask])/N),
label="Hamming")

ax.set_ylabel("20log(X(f))", fontsize=14)

ax.legend(loc=0)                fig.tight_layout()
```



Hình 3-12: Phổ tần số của tín hiệu tổng hợp và tín hiệu qua cửa sổ Hamming

KẾT LUẬN

Lập trình xử lý tín hiệu số trên Python giúp sinh viên hiểu rõ hơn về các thuật toán trong môn học "Xử lý tín hiệu số" với các thư viện lập trình đa dạng nên cần được tập trung nghiên cứu và triển khai ứng dụng.

TÀI LIỆU THAM KHẢO

[1] https://scipy.org/

[2] https://www.numpy.org/