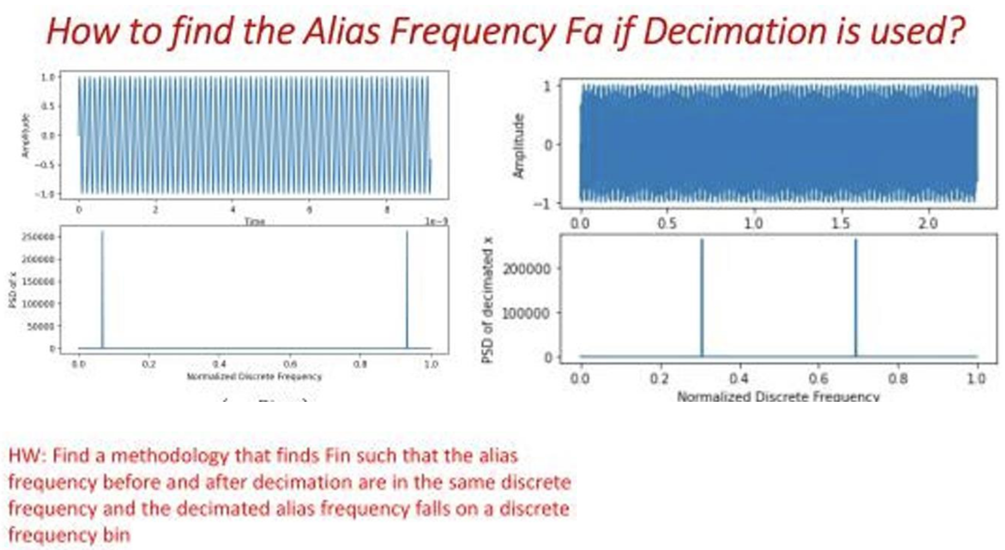
ECEN 610: MIXED SIGNAL INTERFACE HW.1

Name: Yu-Hao Chen   UIN: 435009528   Section: 601

Instructor: Sebastian Hoyos

TA: Haotian



- **Decimation** is the process of **reducing the sampling rate Fs** of a digital signal.
- When a signal is downsampled (Decimation), its frequency components may exceed the new Nyquist frequency:

  F'nyquist= Fs/2M |M=decimate factor

  This causes high-frequency components to **fold back (alias) into the Nyquist range**, resulting in aliasing.

  FsD= Fs/M

  Nyquist D= FsD/2

  Ex: Fs=10k Fin:3k and 6k Nyquist BW= 5k

  3k no alias, but 6k alias back to 4k

  After decimation M=2 FsD=5k Nyquist BW= 2.5k

  Now 3k alias back to 2k, and (6k alias to 4k) and then back to 1k

  Original:

```
0      1k    2k    3k    4k    5k    6k    7k    8k    9k    10k
|------|-----|-----|-----|-----|-----|-----|-----|-----|------|
   1k          3k                6k
```

  Decimate:

```
0      1k    2k    3k    4k    5k
|------|-----|-----|-----|-----|
   1k    2k
```

- Bin: When we perform Fast Fourier Transform (FFT) on the signal, the frequency axis will be divided into N bins, each bin represents a fixed frequency range.

  Ex: Fs=10k Hz FFT N :1024

  Bin: Δf= 10k/1024= 9.77Hz

  Bin1: 0Hz Bin2: 9.77Hz Bin3: 19.54Hz …Bin1024

- FFT can only resolve discrete frequency points. These frequency points are determined by FFT bins(No matter before decimating or after). The bin intervals are:

  Bin size= Fs/N

  If alias frequency fa does not fall exactly on a certain bin, then it will have a frequency error in the FFT spectrum.

  Therefore, we use:

  *Fa= round(Fa/Bin size) * Bin size*

  Align to the nearest FFT bin so that the FFT can parse it correctly.

  Ex: Fs: 1000Hz, FFT N: 16, M=2 (FsD=500Hz ND=8), Fin:1200Hz

  Original Fa: |1200- 1*1000|= 200Hz

  FFT Bin: Fs/N= 62.5Hz

  Normalized Fa: around(Fa/Bin size) *bin size= 187.5Hz (Bin3)

  FaD= Fa mod FsD= 200 mod 500= 200Hz

  Still need to normalized again.

- **Math for doing this**

  1. Fa=|Fin-kfs|, k=round(fin/fs)
  2. FaD= Fa mod FsD
  3. Normalized Bin: Bin=round(Fa/Bin size) * Bin size (for both Fa FaD)

     (Fa normalize and FaD normalized should be the same)

- ✧ After downclocking, the FFT Bin spacing (frequency resolution) remains unchanged.
- ✧ After downclocking, the number of FFT Bins is reduced, but the frequency point spacing remains the same.
- ✧ If alias frequency Fa corresponding bin remains unchanged, it will still fall into the same frequency bin after downscaling to ensure correct alignment.

- **Simulation**

  Fs: 1000Hz, Fin: 1200Hz, FFT N:882, M=2

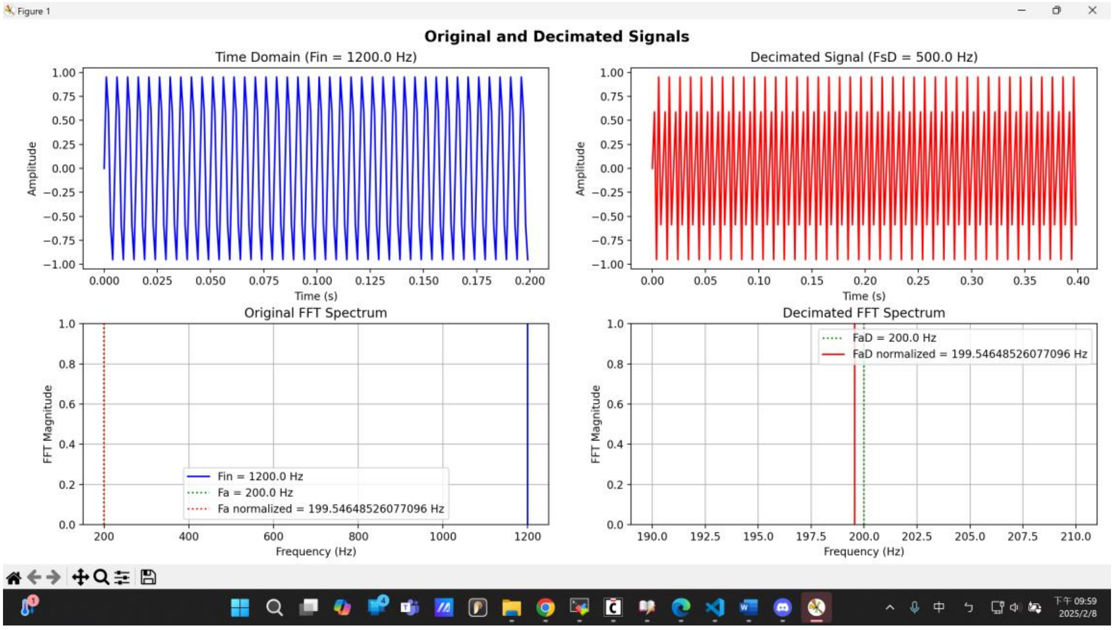
  Estimation: fa=|fin-kfs|=200, k=round(fin/fs)=1

  Bin size: Fs/N= 1000/822= 1.13379Hz

  Normalized Fa= round(Fs/ Bin size) *Bin sized= 199.5468Hz

```
k : 1
Fa (original): 200.0 Hz,  normalized Fa = 200.00000000000003 Hz
FaD = 200.0 Hz, normalized FaD = 200.00000000000003 Hz
PS C:\Users\harri\OneDrive\桌面\TAMU\ECEN 607 Advanced Analog circuit\lab1> & C:/Users/harri/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harri/OneDrive/桌面/TAMU/ECEN 610 Mixed-signal/code/LAB1/Q1/hw1.py"
Fs (Hz): 1000
Fin (Hz): 1200
FFT discrete point N: 882
Decimation Factor M: 2

outcome:
Fs: 1000.0 Hz, FsD (decimated) = 500.0 Hz
Fin: 1200.0 Hz
k : 1
Fa (original): 200.0 Hz,  normalized Fa = 199.54648526077096 Hz
FaD = 200.0 Hz, normalized FaD = 199.54648526077096 Hz
```



Code for Fa and normalized

C: > Users > harri > OneDrive > 桌面 > TAMU > ECEN 610 Mixed-signal > code > LAB1 > Q1 > hw1.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft

Fs = float(input("Fs (Hz): "))
Fin = float(input("Fin (Hz): "))
N = int(input("FFT discrete point N: "))  # EX: N=100 fin(Hz) cut into x[1-100] discrete point x[y]= fin/N * y
M = int(input("Decimation Factor M: "))  # Decimation Factor EX: Fs/M into FsD

# After decoimate FsD and ND
FsD = Fs / M
ND = N // M  # 降頻後的 FFT 點數

# Fa (original)
k = round(Fin / Fs) # closest kFs
Fa = abs(Fin - k * Fs) # Frequnece alias back to Nyquist BW

# FaD (mod to insure between FsD)
FaD = Fa % FsD

# bin size
bin_size = Fs / N
bin_size_D = FsD / ND

# Fix alias freq · fit FFT bin
normalized_Fa = round(Fa / bin_size) * bin_size
normalized_FaD = round(FaD/ bin_size_D) * bin_size_D
```

FFT and pre-plotting

```python

# time
t = np.arange(N) / Fs  # original time
t_decimated = np.arange(ND) / FsD  # D time

# Orignial Fin
x_original = np.sin(2 * np.pi * Fin * t)
x_alias = np.sin(2 * np.pi * normalized_Fa * t)  # 修正後 alias 頻率

# decimated freq
x_decimated = x_original[::M] # FsD= Fs/M

# FFT [:N//2]slicing
X_original = np.abs(fft(x_original))  # fft abs conjurgrate for Magnitude not Phase
X_decimated = np.abs(fft(x_decimated))  # fft output [-Fs/2- Fs/2] only keep the 0- Fs/2

# fft (0, Fs/N, Fs/2N ... {N/(2-1)Fs}/N, -Fs/2 )
# [:N//2] for the first half ( to show nyquist BW)
# [N//2:] for the second half ( negative part )

freqs = np.fft.fftfreq(N, d=1/Fs)  #  FFT freq
freqs_decimated = np.fft.fftfreq(ND, d=1/FsD)  # D FFT freq
```

Plotting

```python

# Print
print("\noutcome:")
print(f"Fs: {Fs} Hz, FsD (decimated) = {FsD} Hz")
print(f"Fin: {Fin} Hz")
print(f"k : {k}")
print(f"Fa (original): {Fa} Hz,  normalized Fa = {normalized_Fa} Hz")
print(f"FaD = {FaD} Hz, normalized FaD = {normalized_FaD} Hz")

# Plot
fig, axes = plt.subplots(2, 2, figsize=(12, 8))

# Upper left: original signal (time domain)
axes[0, 0].plot(t[:200], x_original[:200], color='blue')
axes[0, 0].set_title(f"Time Domain (Fin = {Fin} Hz)")
axes[0, 0].set_xlabel("Time (s)")
axes[0, 0].set_ylabel("Amplitude")

# Upper right: signal after downconversion (time domain)
axes[0, 1].plot(t_decimated[:200], x_decimated[:200], color='red')
axes[0, 1].set_title(f"Decimated Signal (Fs' = {FsD} Hz)")
axes[0, 1].set_xlabel("Time (s)")
axes[0, 1].set_ylabel("Amplitude")

# Bottom left: original FFT spectrum
#axes[1, 0].plot(freqs, X_original, label=f"Fin = {Fin} Hz", color='blue')
axes[1, 0].axvline(Fin, color='blue', label=f"Fin = {Fin} Hz")  # 原始頻率標示
axes[1, 0].axvline(Fa, color='green', linestyle=":", label=f"Fa = {Fa} Hz")
axes[1, 0].axvline(normalized_Fa, color='red', linestyle=":", label=f"Fa normalized = {normalized_Fa} Hz")  # Alias 頻率標
axes[1, 0].set_xlabel("Frequency (Hz)")
axes[1, 0].set_ylabel("FFT Magnitude")
axes[1, 0].set_title("Original FFT Spectrum")
axes[1, 0].legend()
```

```python
axes[1, 0].legend()
axes[1, 0].grid()

# Bottom right: FFT spectrum after downconversion
#axes[1, 1].plot(freqs_decimated, X_decimated, label=f"Fa' = {normalized_FaD} Hz", color='red')
axes[1, 1].axvline(Fa, color='green', linestyle=":", label=f"FaD = {FaD} Hz")
axes[1, 1].axvline(normalized_FaD, color='red',  label=f"FaD normalized = {normalized_FaD} Hz")
axes[1, 1].set_xlabel("Frequency (Hz)")
axes[1, 1].set_ylabel("FFT Magnitude")
axes[1, 1].set_title("Decimated FFT Spectrum")
axes[1, 1].legend()
axes[1, 1].grid()

fig.suptitle("Original and Decimated Signals", fontsize=14, fontweight='bold')

plt.tight_layout()
plt.show()
```