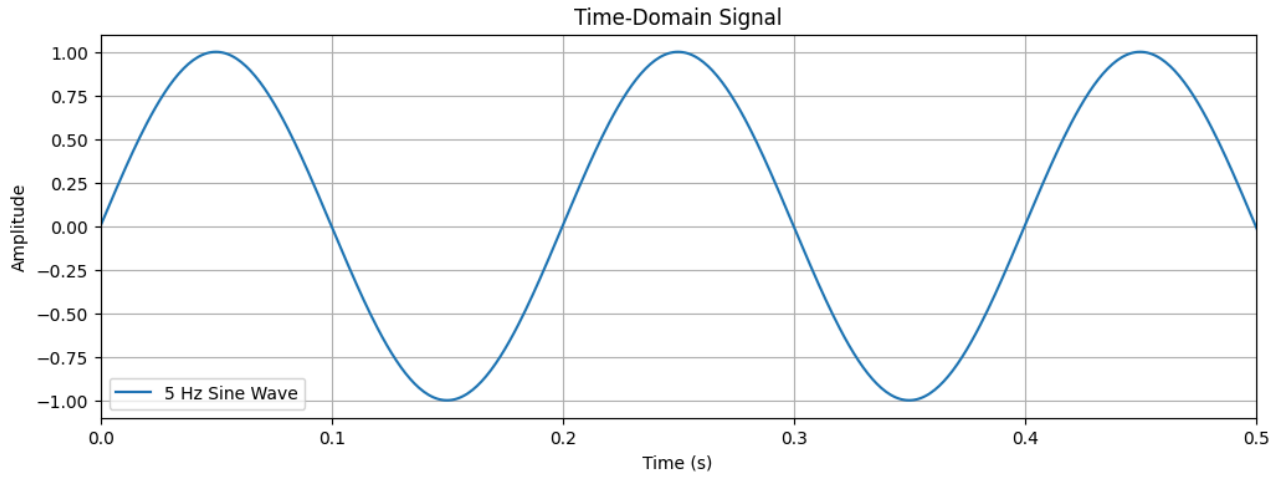
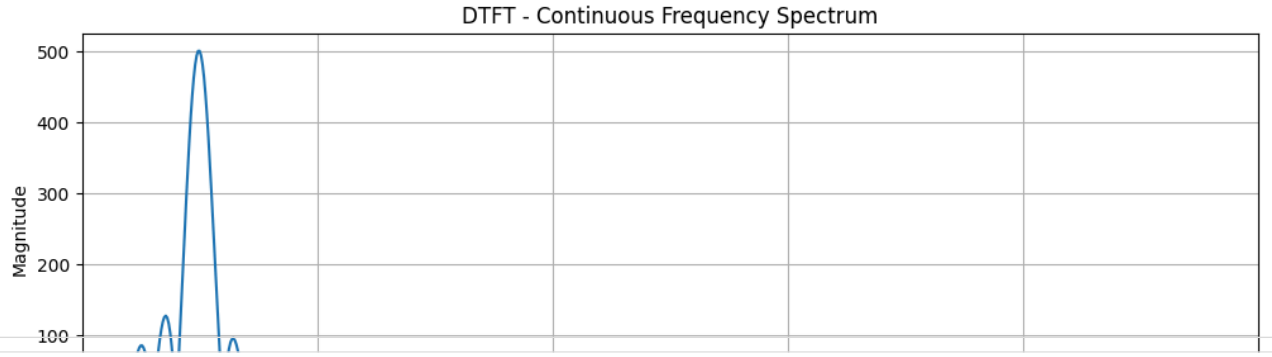
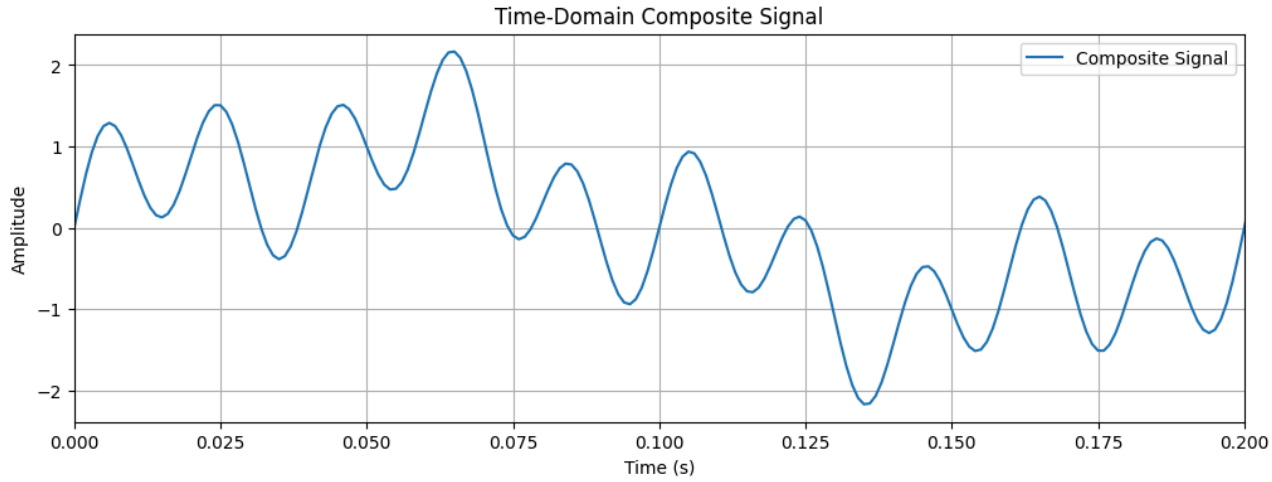
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz

# Parameters
fs = 1000       # Sampling frequency (Hz)
f = 5           # Signal frequency (Hz)
duration = 1    # seconds
N = fs * duration  # Number of samples


# (a) Generate sine wave
t = np.arange(N) / fs
x = np.sin(2 * np.pi * f * t)

# (b) Plot time-domain waveform
plt.figure(figsize=(12, 4))
plt.plot(t, x, label="5 Hz Sine Wave")
plt.title("Time-Domain Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.xlim(0, 0.5)  # zoom into 0.5s for clarity
plt.grid(True)
plt.legend()
plt.show()

# (c) Compute DTFT (approximated using freqz)
w, h = freqz(x, worN=8000)  # DTFT
frequencies = w * fs / (2 * np.pi)

plt.figure(figsize=(12, 4))
plt.plot(frequencies, np.abs(h))
plt.title("DTFT - Continuous Frequency Spectrum")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0, 50)  # show only first 50 Hz
plt.grid(True)
plt.show()

# (d) Compute DFT (using FFT)
X = np.fft.fft(x)
freqs = np.fft.fftfreq(len(X), 1/fs)

plt.figure(figsize=(12, 4))
plt.stem(freqs[:N//2], np.abs(X[:N//2]))
plt.title("DFT - Discrete Frequency Spectrum")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0, 50)  # show only up to 50 Hz
plt.grid(True)
plt.show()
```

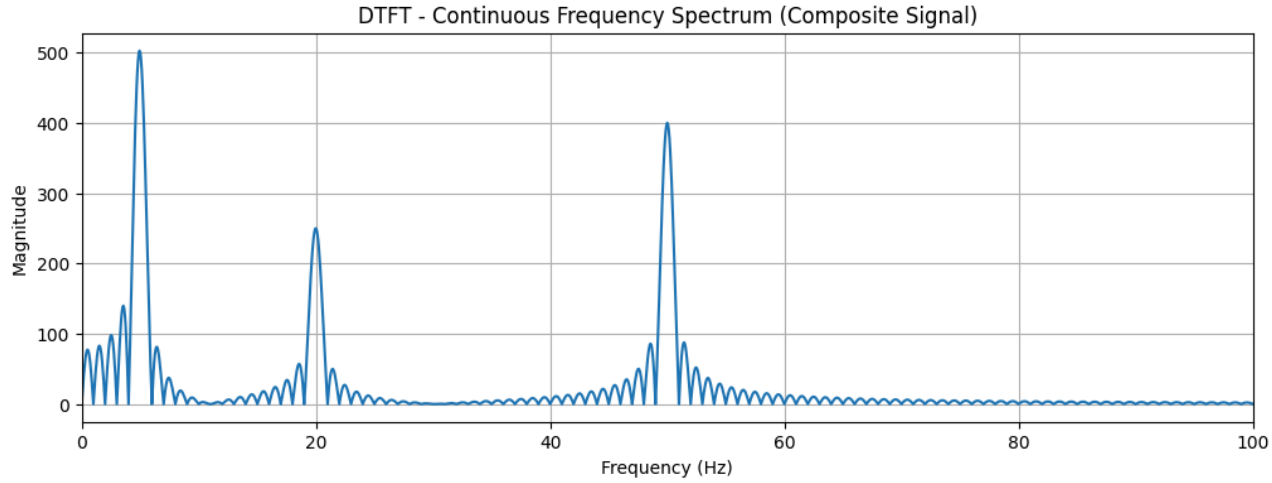


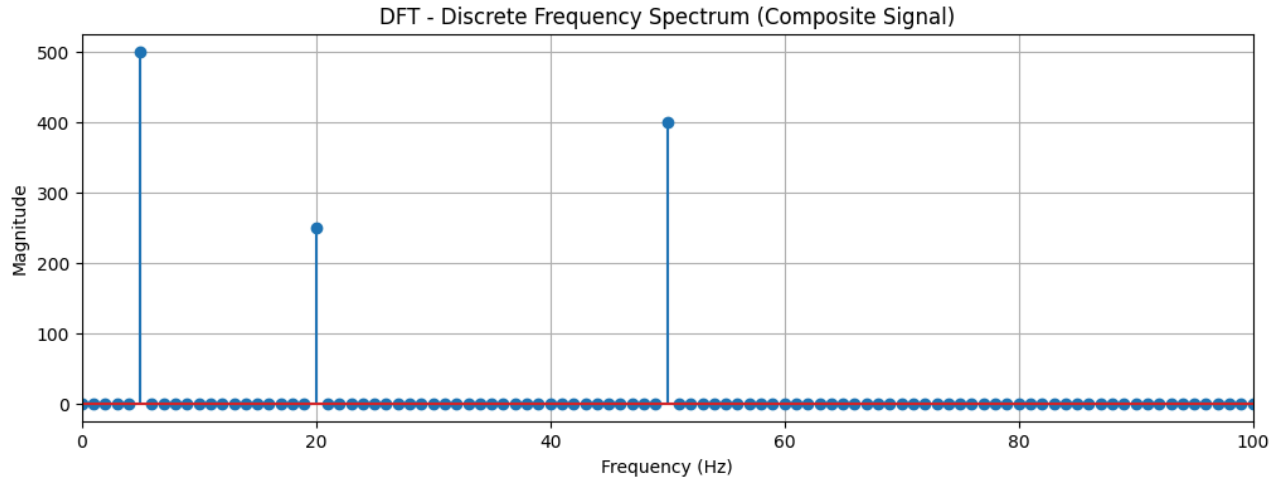



```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz

# Parameters
fs = 1000       # Sampling frequency (Hz)
duration = 1    # seconds
N = fs * duration  # Number of samples
t = np.arange(N) / fs

# (a) Generate composite signal
x = np.sin(2 * np.pi * 5 * t) + 0.5 * np.sin(2 * np.pi * 20 * t) + 0.8 * np.sin(2 * np.pi * 50 * t)

# (b) Plot time-domain waveform
plt.figure(figsize=(12, 4))
plt.plot(t, x, label="Composite Signal")
plt.title("Time-Domain Composite Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.xlim(0, 0.2)  # zoom into 0.2s for clarity
plt.grid(True)
plt.legend()
plt.show()

# (c) Compute DTFT (approximated using freqz)
w, h = freqz(x, worN=8000)  # DTFT
frequencies = w * fs / (2 * np.pi)

plt.figure(figsize=(12, 4))
plt.plot(frequencies, np.abs(h))
plt.title("DTFT - Continuous Frequency Spectrum (Composite Signal)")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0, 100)  # show first 100 Hz for clarity
plt.grid(True)
plt.show()

# (d) Compute DFT (using FFT)
X = np.fft.fft(x)
freqs = np.fft.fftfreq(len(X), 1/fs)

plt.figure(figsize=(12, 4))
plt.stem(freqs[:N//2], np.abs(X[:N//2]))
plt.title("DFT - Discrete Frequency Spectrum (Composite Signal)")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0, 100)  # show up to 100 Hz
```

```
plt.grid(True)
plt.show()
```

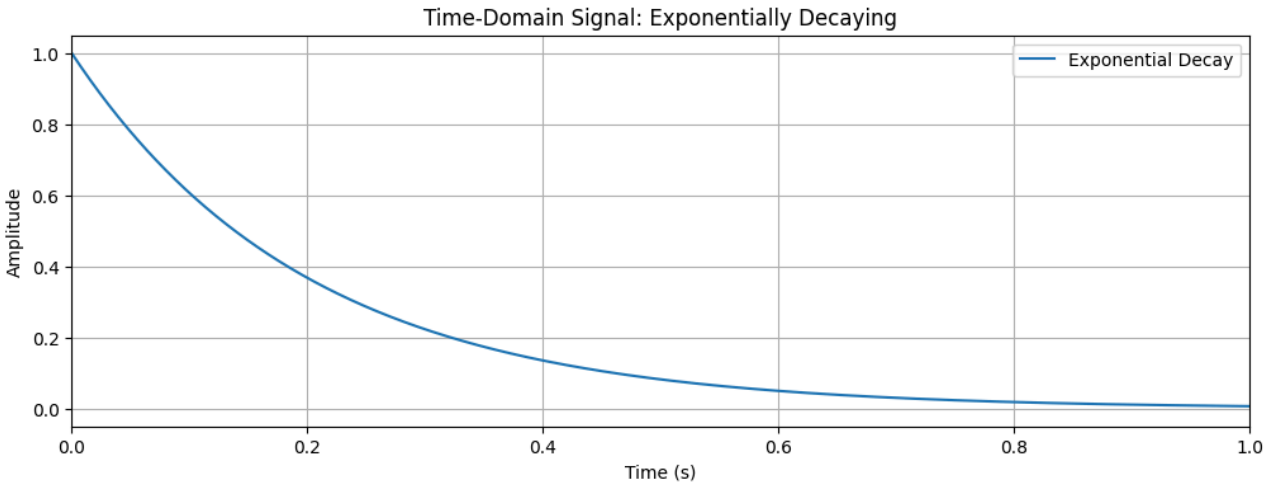


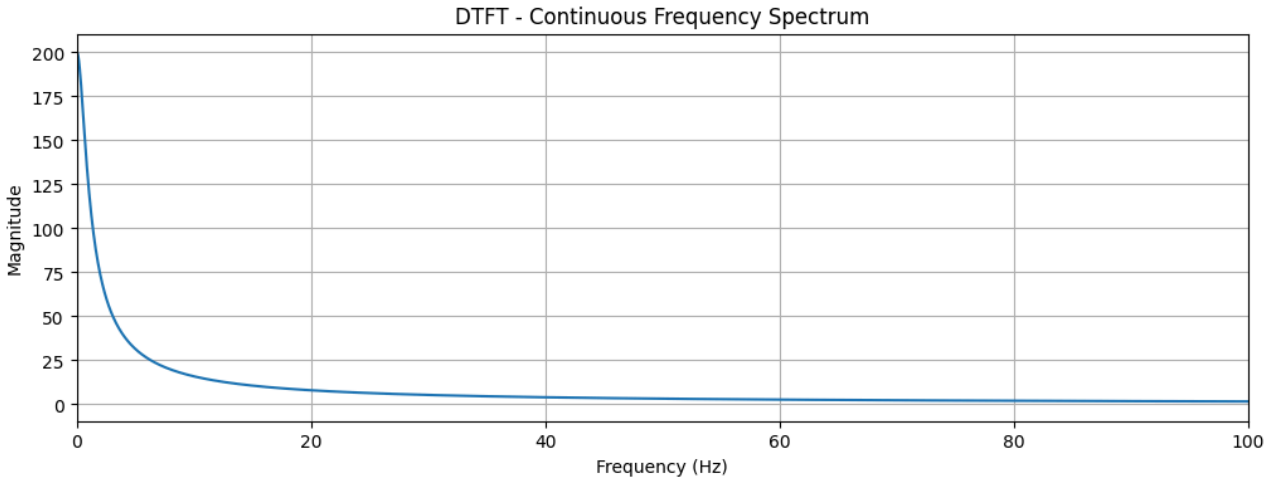


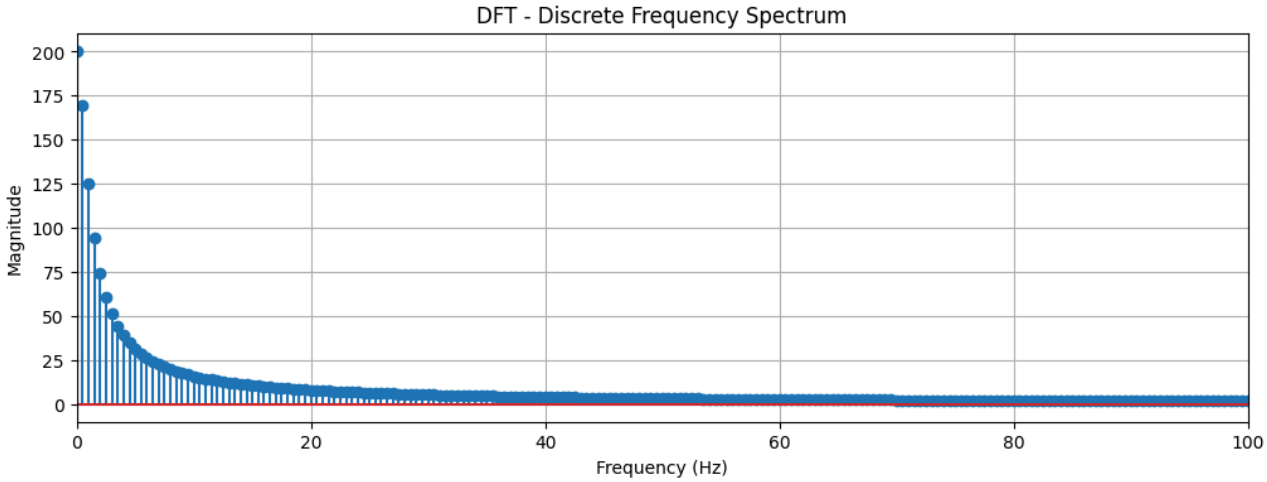



```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz

# Parameters
fs = 1000        # Sampling frequency (Hz)
duration = 2     # seconds
N = fs * duration
t = np.arange(N) / fs
alpha = 5        # Decay constant

# (a) Generate exponentially decaying signal
x = np.exp(-alpha * t)

# (b) Plot time-domain waveform
plt.figure(figsize=(12, 4))
```

```
plt.plot(t, x, label="Exponential Decay")
plt.title("Time-Domain Signal: Exponentially Decaying")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.xlim(0, 1)  # zoom for clarity
plt.grid(True)
plt.legend()
plt.show()

# (c) Compute DTFT (approximated using freqz)
w, h = freqz(x, worN=8000)
frequencies = w * fs / (2 * np.pi)

plt.figure(figsize=(12, 4))
plt.plot(frequencies, np.abs(h))
plt.title("DTFT - Continuous Frequency Spectrum")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0, 100)  # focus on low frequencies
plt.grid(True)
plt.show()

# (d) Compute DFT (using FFT)
X = np.fft.fft(x)
freqs = np.fft.fftfreq(len(X), 1/fs)

plt.figure(figsize=(12, 4))
plt.stem(freqs[:N//2], np.abs(X[:N//2]))
plt.title("DFT - Discrete Frequency Spectrum")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0, 100)  # zoom into low-frequency range
plt.grid(True)
plt.show()
```








```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz

# Parameters
fs = 1000        # Sampling frequency (Hz)
duration = 2     # total time window (seconds)
N = fs * duration
t = np.arange(N) / fs

pulse_width = 0.2  # Pulse width in seconds

# (a) Generate rectangular pulse
x = np.zeros_like(t)
x[(t >= 0) & (t <= pulse_width)] = 1

# (b) Plot time-domain waveform
plt.figure(figsize=(12, 4))
plt.plot(t, x, label="Rectangular Pulse")
```

```
plt.plot(t, x, label='Rectangular Pulse')
plt.title("Time-Domain Signal: Rectangular Pulse")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.xlim(0, 0.5)  # zoom into first 0.5s
plt.grid(True)
plt.legend()
plt.show()

# (c) Compute DTFT (approximated using freqz)
w, h = freqz(x, worN=8000)
frequencies = w * fs / (2 * np.pi)

plt.figure(figsize=(12, 4))
plt.plot(frequencies, np.abs(h))
plt.title("DTFT - Continuous Frequency Spectrum")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0, 200)  # show first 200 Hz
plt.grid(True)
plt.show()

# (d) Compute DFT (using FFT)
X = np.fft.fft(x)
freqs = np.fft.fftfreq(len(X), 1/fs)

plt.figure(figsize=(12, 4))
plt.stem(freqs[:N//2], np.abs(X[:N//2]))
plt.title("DFT - Discrete Frequency Spectrum")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.xlim(0, 200)  # zoom into low-frequency range
plt.grid(True)
plt.show()
```