ECE 398-MA
Introduction to Modern Communication with Python and SDR
Lab 5 – Pulse-Shaping

Noah Breit

March 13, 2025

1 Assignment 1

```
import numpy as np
import matplotlib.pyplot as plt

num_symbols = 64
sps = 16

############# PART 1 ##############
def generate_bpsk_symbols(num_symbols):
"""Generate BPSK symbols from random bits."""
np.random.seed(0)
bits = np.random.randint(0, 2, num_symbols)
symbols = np.where(bits == 1, 1, -1)
return symbols

def upsample_symbols(symbols, sps):
"""Upsample symbols by inserting zeros."""
up_sym = np.zeros(len(symbols) * sps)
up_sym[::sps] = symbols
return up_sym

def plot_upsampled_symbols(up_sym):
"""Plot upsampled symbols."""
plt.figure()
plt.stem(up_sym)
plt.title("Upsampled␣Symbols")
plt.show()
```

```
# part 1 driver
symbols = generate_bpsk_symbols(num_symbols)
up_sym = upsample_symbols(symbols, sps)
plot_upsampled_symbols(up_sym)
```

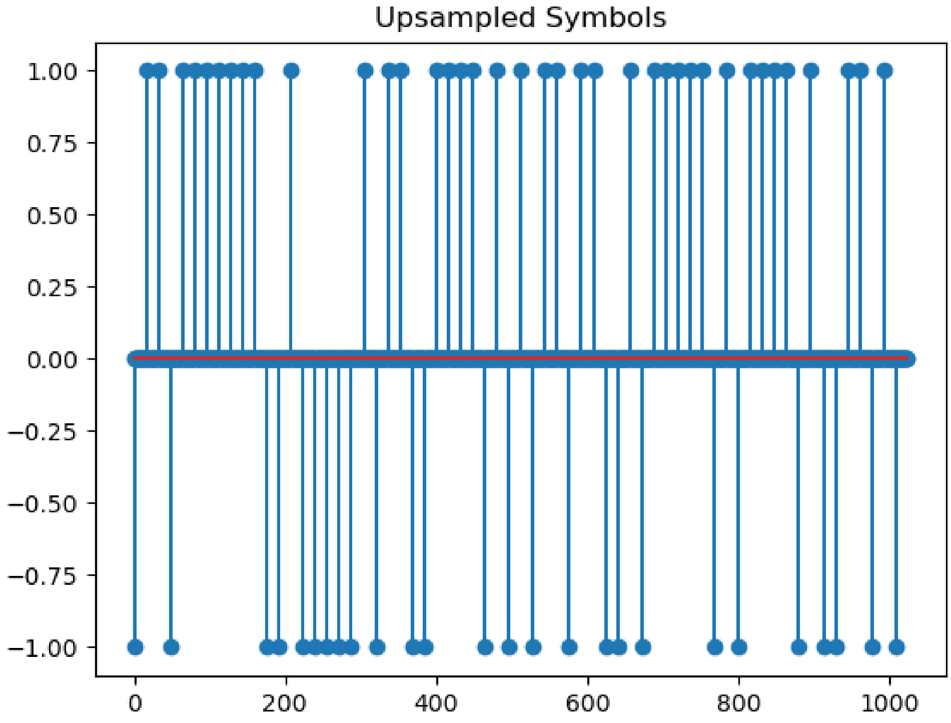

Figure 1: Upsampled BPSK Symbols

2 Assignment 2

```
############ PART 2 ################
def rcosfilter(N, alpha, Tb, Fs):
"""
Generates a raised cosine (RC) filter (FIR) impulse response.

Parameters
----------
N : int
Length of the filter in samples.
```

```
alpha : float
Roll off factor (Valid values are [0, 1]).

Tb : float
Symbol period.

Fs : float
Sampling Rate.

Returns
-------
h_rc : 1-D ndarray of floats
Impulse response of the raised cosine filter.
"""
t = ((np.arange(N) - N / 2))*1/float(Fs)
h_rc = np.sinc(t / Tb) * np.cos(np.pi * alpha * t / Tb) / (1 - 4 * alpha**2
# h_rc[np.abs(t) > Tb / (2 * alpha)] = 0  # Ensure filter is zero outside t
return h_rc

def rectangular_pulse(N):
"""Generate rectangular pulse."""
return np.ones(N)

def convolve_with_pulse(up_sym, pulse):
"""Convolve upsampled symbols with a pulse."""
return np.convolve(up_sym, pulse, mode='full')

def plot_time_domain(x):
"""Plot signal in time domain."""
plt.figure()
plt.plot(x)
plt.title("Time␣Domain")
plt.show()

def plot_frequency_domain(x):
"""Plot signal in frequency domain."""
plt.figure()
plt.plot(np.abs(np.fft.fft(x)))
plt.yscale('log')
plt.title("Frequency␣Domain")
plt.show()

def plot_eye_diagram(x, sps, numeye=2):
"""Plot eye diagram."""
plt.figure()
```

```
for k in range(len(x) // sps):
start_idx = k * sps - sps // 2
end_idx = (k + numeye) * sps + sps // 2
if start_idx < 0:
start_idx = 0
if end_idx > len(x):
break
plt.plot(x[start_idx:end_idx], color='gray', alpha=0.5, linewidth=1.5)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()

# part 2 driver
N = 15 * sps + 1
Tb = sps
Fs = 1

# Rectangular Pulse
rect_pulse = rectangular_pulse(sps)
rect_convolved = convolve_with_pulse(up_sym, rect_pulse)
plot_time_domain(rect_convolved)
plot_frequency_domain(rect_convolved)
plot_eye_diagram(rect_convolved, sps)

# Raised-cosine Pulses
alphas = [0, 0.5, 1]
for alpha in alphas:
rc_pulse = rcosfilter(N, alpha, Tb, Fs)
rc_convolved = convolve_with_pulse(up_sym, rc_pulse)
plot_time_domain(rc_convolved)
plot_frequency_domain(rc_convolved)
plot_eye_diagram(rc_convolved, sps)
```

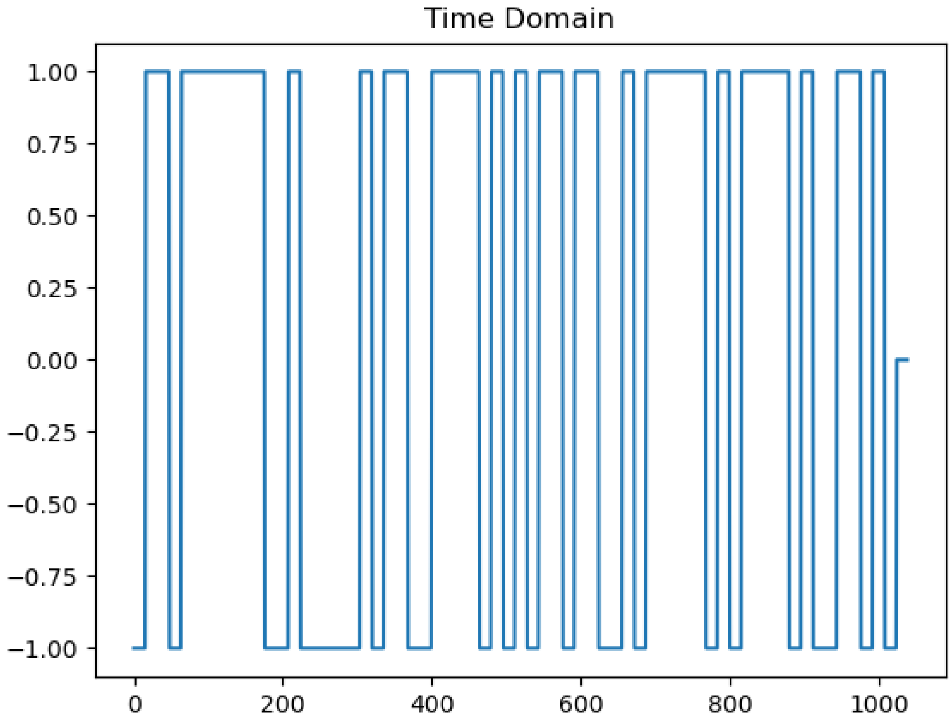

Figure 2: Rect Pulse – Time-Domain Signal

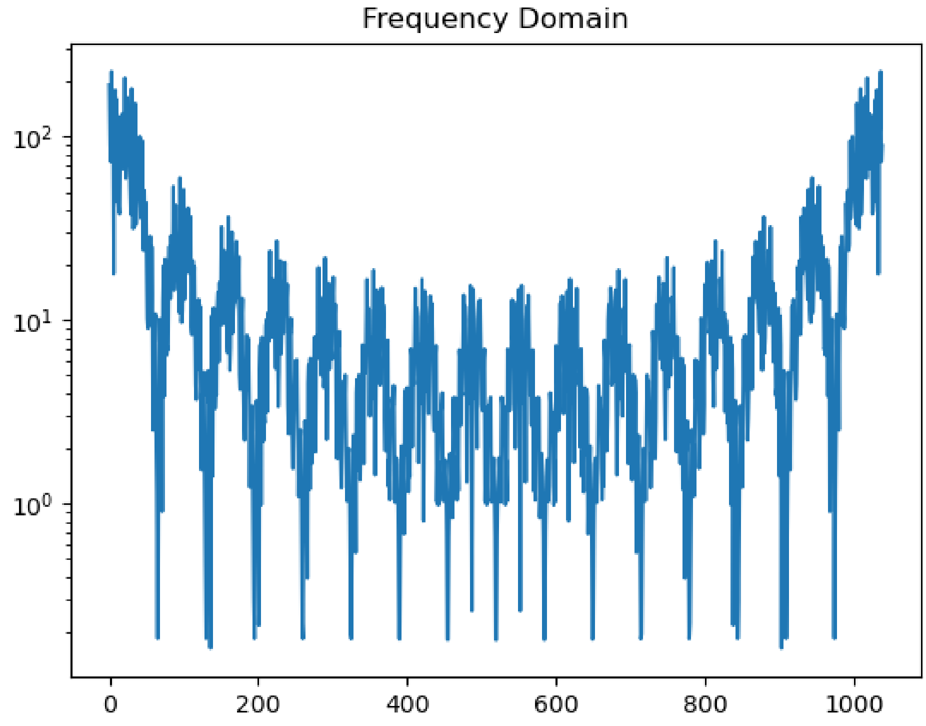

Figure 3: Rect Pulse – Freq-Domain Signal

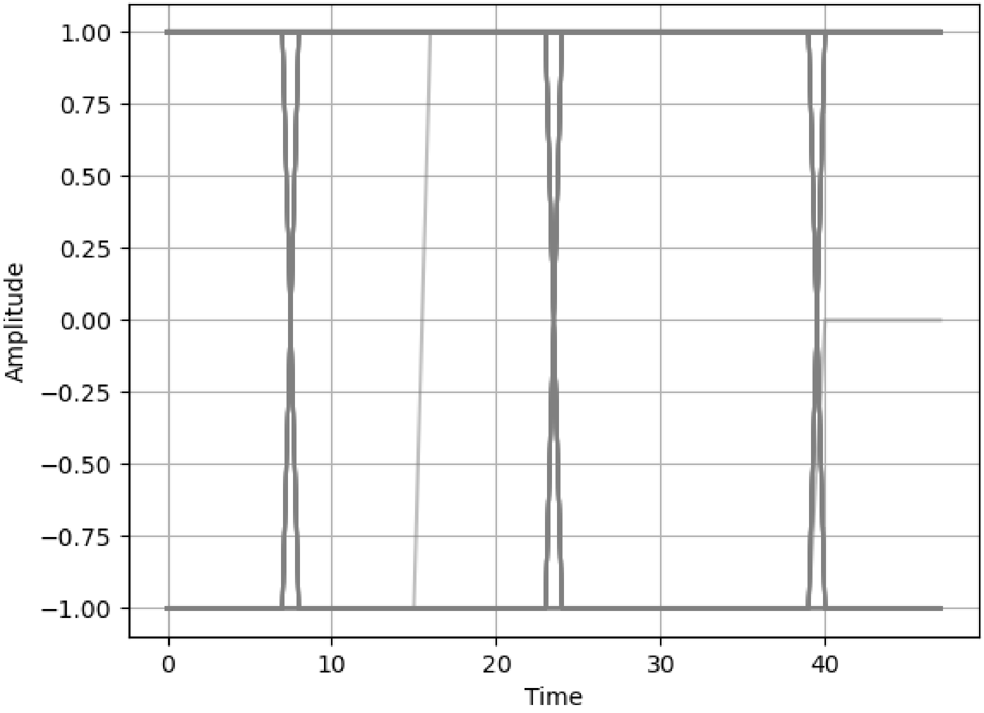

Figure 4: Rect Pulse – Eye Diagram

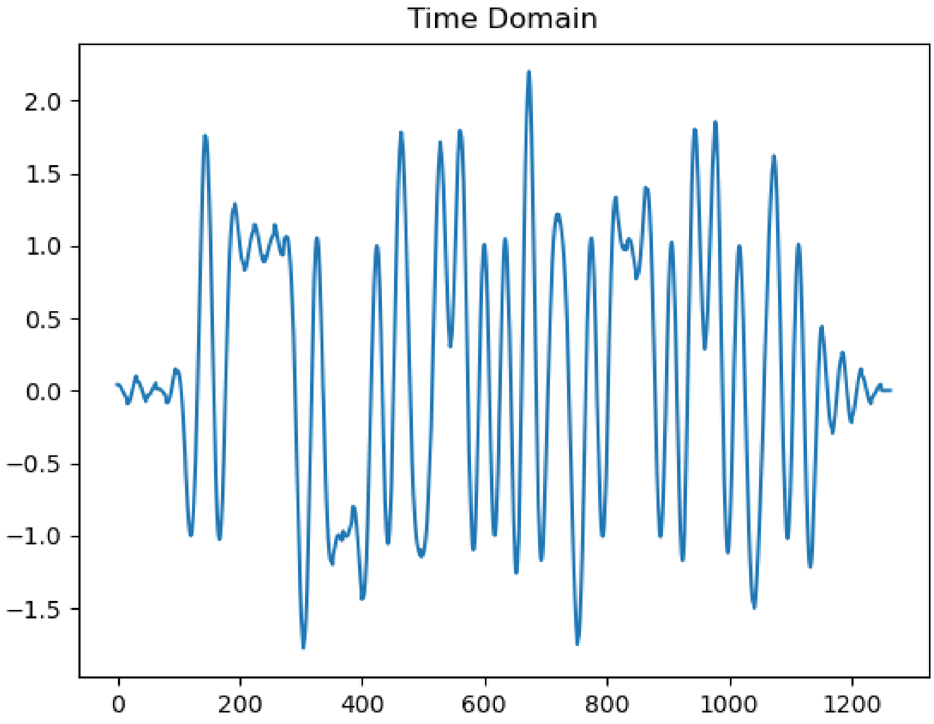

Figure 5: Cosine Pulse ALPHA=0 – Time-Domain Signal

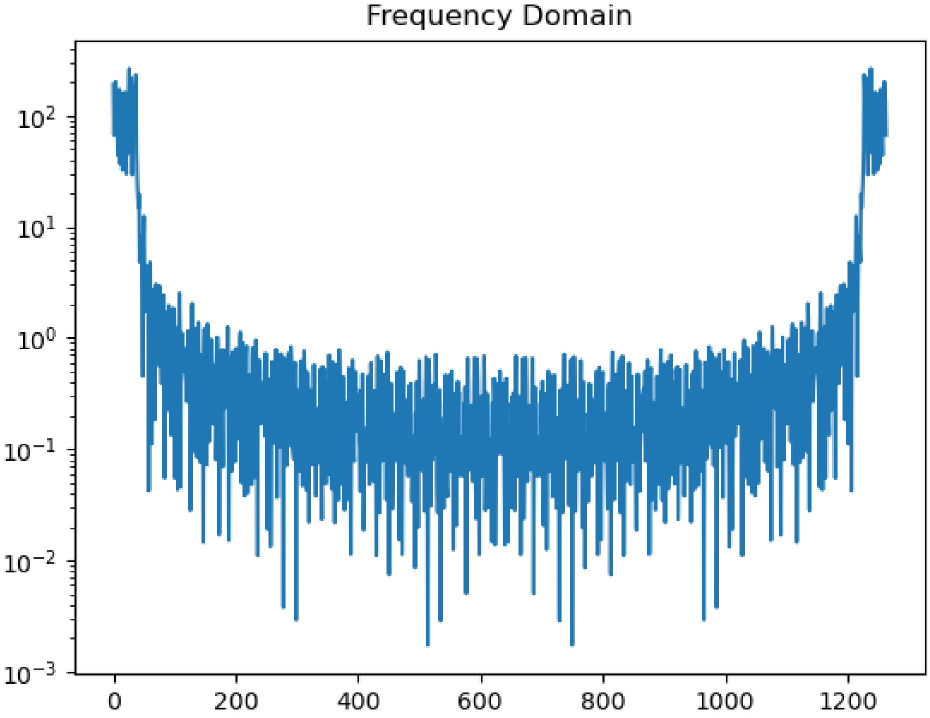

Figure 6: Cosine Pulse ALPHA=0 – Freq-Domain Signal

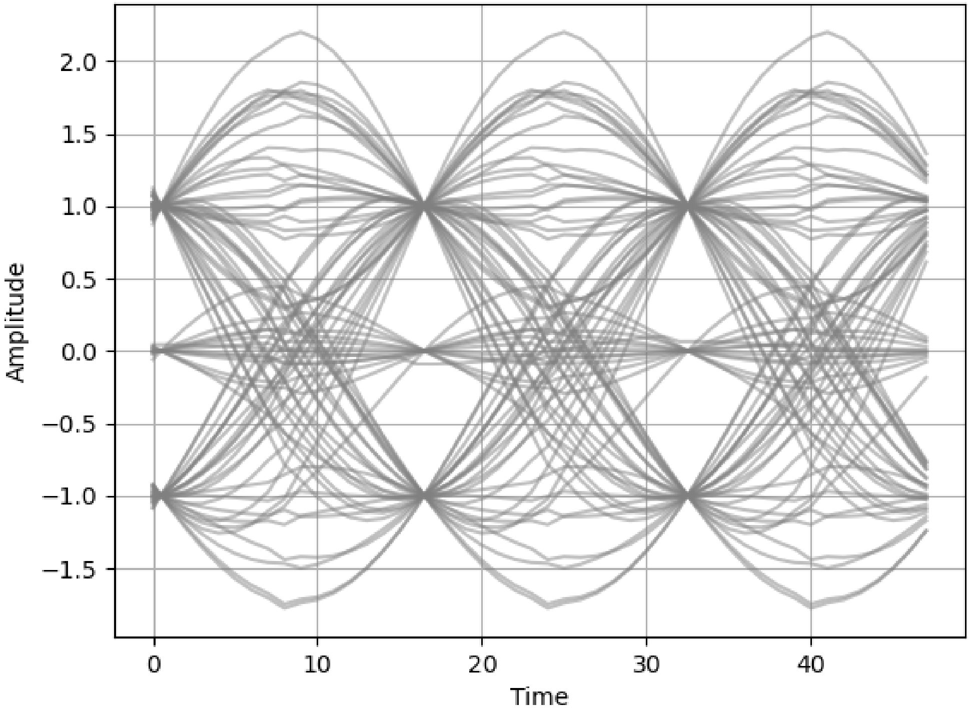

Figure 7: Cosine Pulse ALPHA=0 – Eye Diagram

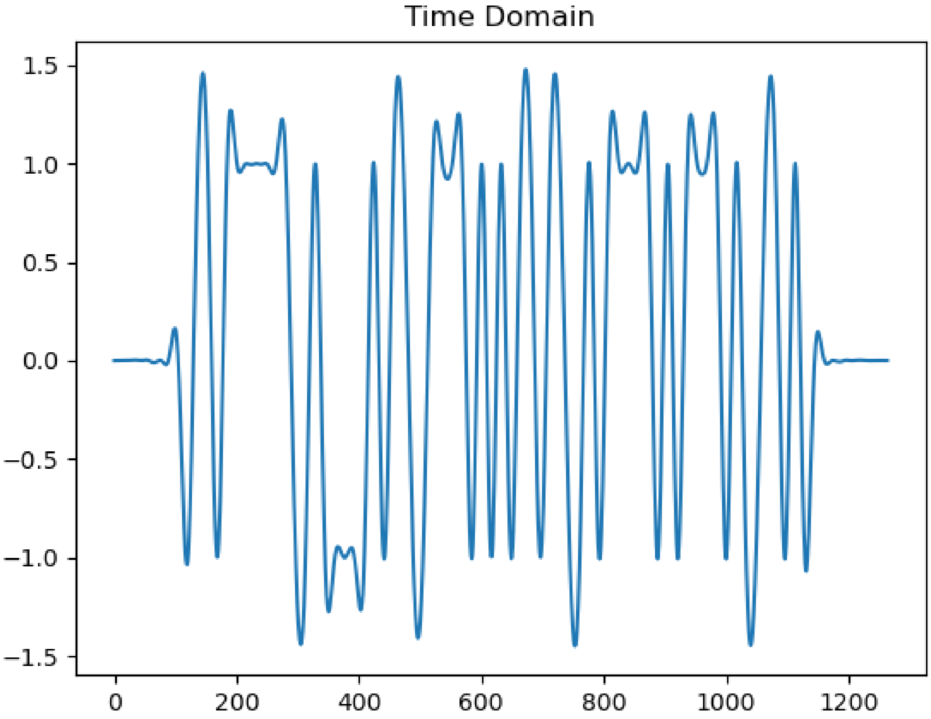

Figure 8: Cosine Pulse ALPHA=0.5 – Time-Domain Signal

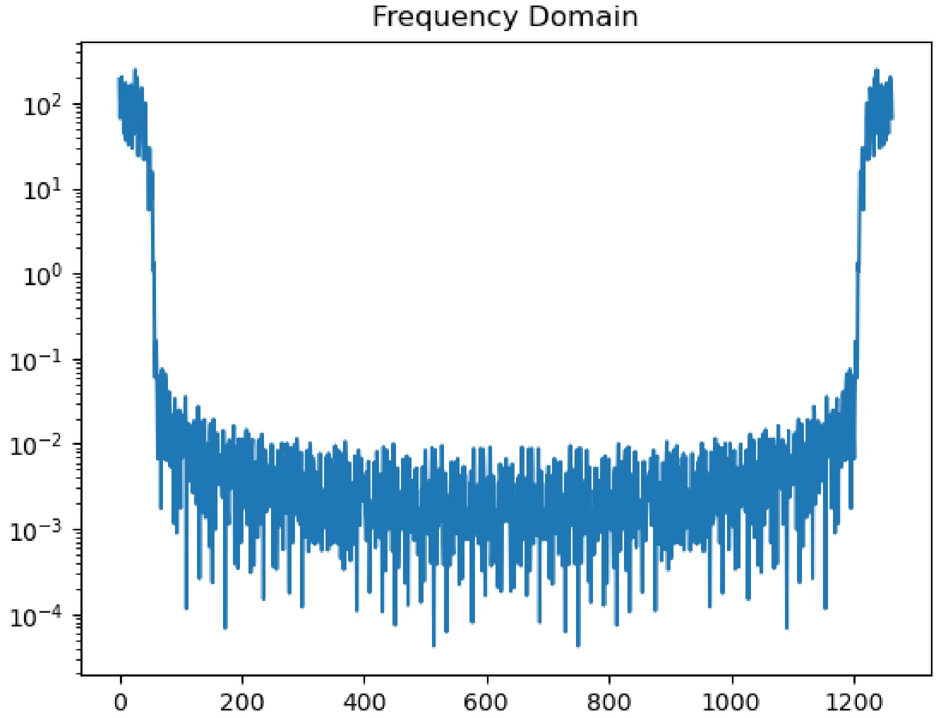

Figure 9: Cosine Pulse ALPHA=0.5 – Freq-Domain Signal

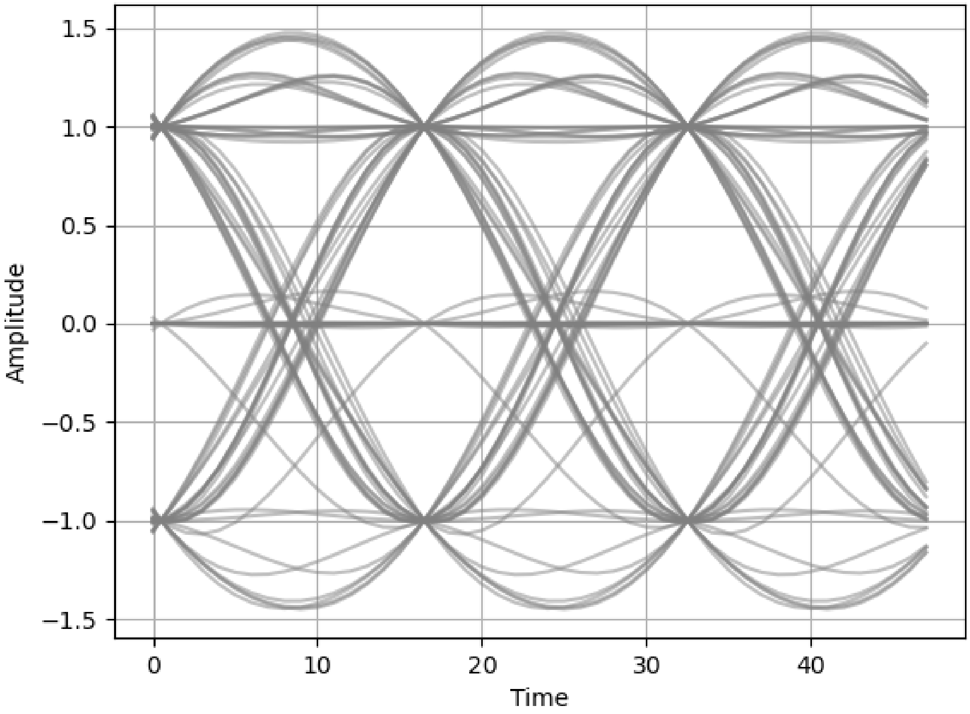

Figure 10: Cosine Pulse ALPHA=0.5 – Eye Diagram

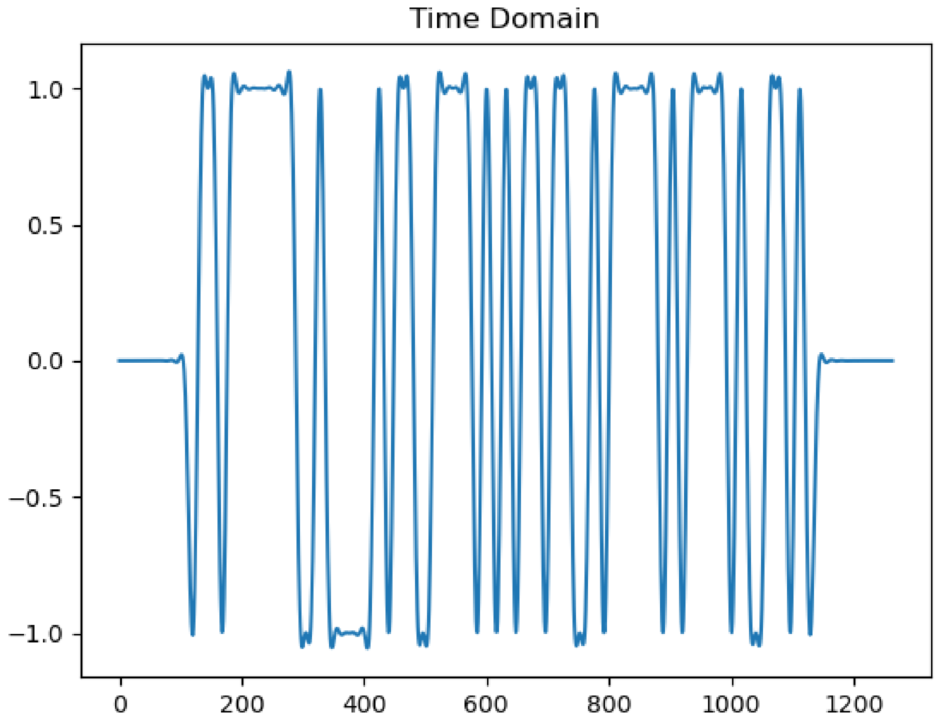

Figure 11: Cosine Pulse ALPHA=1 – Time-Domain Signal

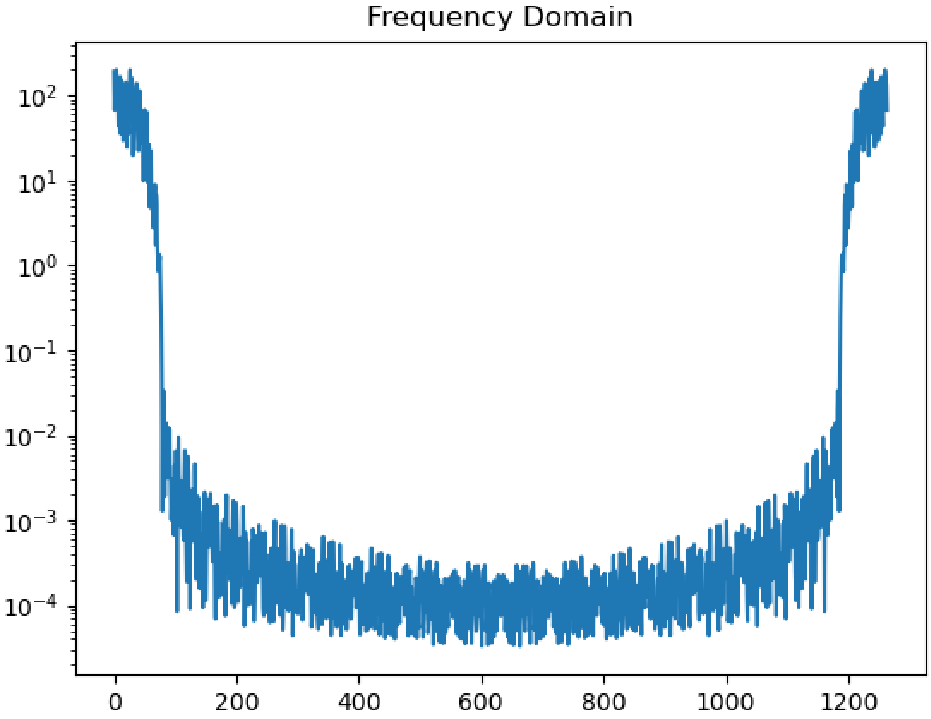

Figure 12: Cosine Pulse ALPHA=1 – Freq-Domain Signal

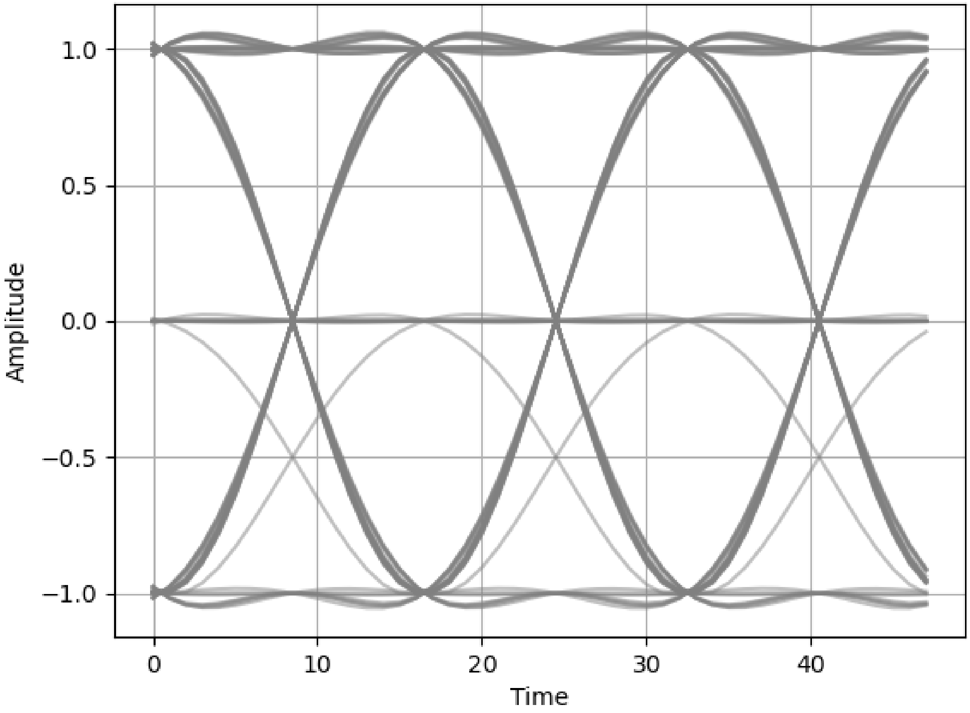

Figure 13: Cosine Pulse ALPHA=1 – Eye Diagram