Lab 3

Objective

To investigae the spetral properties of waeveforms and understand the impact of amplitude modulations

Fourier transform and spectral analysis

Establish communication channel, W1 - generate sinusoidal waveform (1kHz, 1V, 0V offset), Read via CH1

Communication Channel set up

In [ ]:
```
# communication channel setup
import serial
import serial.tools.list_ports
import numpy as np
import plotly.graph_objs as go

VID = 61525
PID = 38912

device = None

ports = serial.tools.list_ports.comports()
for p in ports:
    if p.vid == VID and p.pid == PID:
        try:
            device = serial.Serial(p.device)
        except serial.SerialException:
            print('Reconnect the controller unit.')

if device is None:
    raise Exception('No suitable device detected.')
```

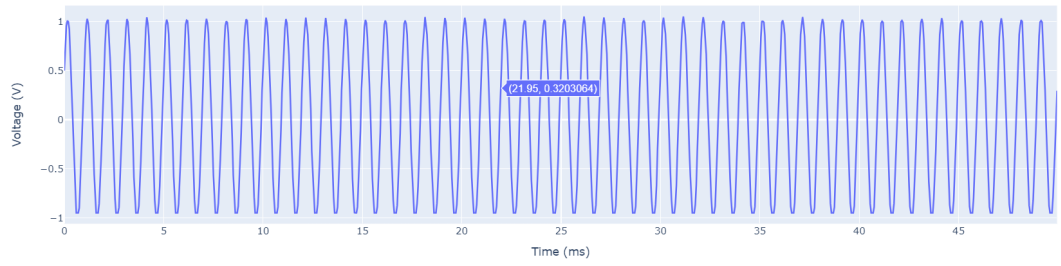
Waveform generation

In [ ]:
```
# waveform generation
ns = 64
freq = 1000
amp = 1
offset = 0
cmd_gensin = 's100'
cmd_gensin += str(ns).zfill(3) + str(freq).zfill(7) + str(int(amp*100)).zfill(4)
device.write(bytes(cmd_gensin, 'utf-8'))
```

Out[ ]: 23

Waveform detection and plotting

In [ ]:
```
# waveform detection and plotting
fs = 20000
c2 = 0
gain1 = 138
dco1 = 130
c1 = 0
gain2 = 138
dco2 = 130
cmd_readosc = 'm1' + str(fs).zfill(6) + str(c1) + str(gain1).zfill(3) + str(dco1
bytedata = bytearray(4000)
device.reset_input_buffer()
device.write(bytes(cmd_readosc, 'utf-8'))
device.readline()
device.readinto(bytedata)
data = np.frombuffer(bytedata, dtype='uint16').reshape((2, 1000))
t = np.arange(1000)/fs
sig = 2*(data[0, :] - np.mean(data[0, :]))/np.ptp(data[0, :]) # adjust
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=sig))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (V)')
```



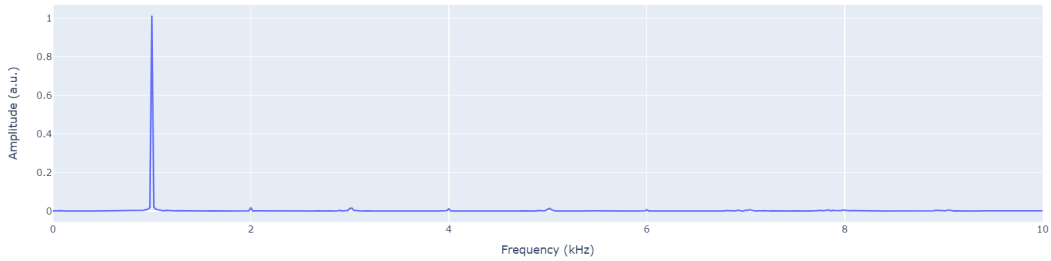
Plotting of sinusoidal voltage signal taken from CH1

rfft() - used to compute the fast Fourier transform for a real number array. Returns an array of complex numbers where thier magnitude and phase represent the amplitude and phase of the corresponding frequency component.

rfftfreq() - create a frequency array based on the sampling frequency and the number of samples

abs() - functions for magnitutdes.

In [ ]:
```
f = np.fft.rfftfreq(len(t), d=1/fs)  # frequency array
spec = np.abs(np.fft.rfft(sig))/len(f) # spectral amplitudes
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```



Generating triangular (W1) and sawtooth (W2) waveform

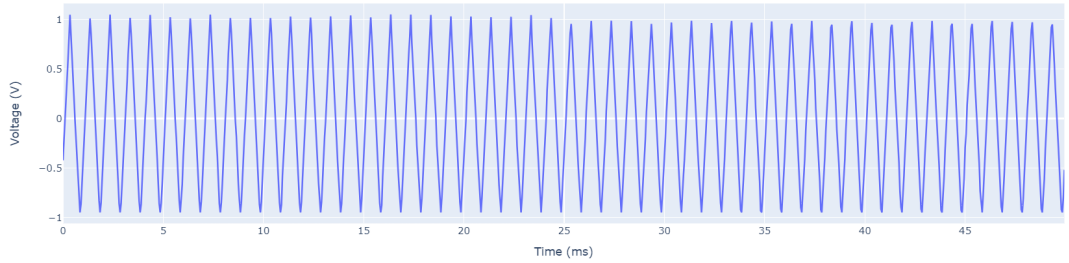
Generating Triangular Waveform (W1)

In [ ]:
```
# waveform generation
ns = 64             #samples per cycle
freq = 1000         #freq
amp = 1             #amplitude
offset = 0          #dc offset
cmd_gensin = 's110' #triangular waveform at W1
cmd_gensin += str(ns).zfill(3) + str(freq).zfill(7) + str(int(amp*100)).zfill(4)
device.write(bytes(cmd_gensin, 'utf-8'))
```

Out[ ]: 23

Plotting of voltage signal taken from CH1

In [ ]:
```
# waveform detection and plotting
fs = 20000
c2 = 0
gain1 = 138
dco1 = 130
c1 = 0
gain2 = 138
dco2 = 130
cmd_readosc = 'm1' + str(fs).zfill(6) + str(c1) + str(gain1).zfill(3) + str(dco1
bytedata = bytearray(4000)
device.reset_input_buffer()
device.write(bytes(cmd_readosc, 'utf-8'))
device.readline()
device.readinto(bytedata)
data = np.frombuffer(bytedata, dtype='uint16').reshape((2, 1000))
t = np.arange(1000)/fs
sig = 2*(data[0, :] - np.mean(data[0, :]))/np.ptp(data[0, :]) # adjust
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=sig))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (V)')
```



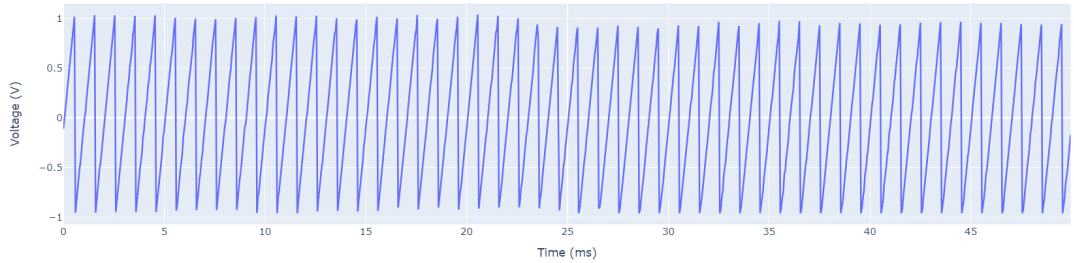
In [ ]:
```
f = np.fft.rfftfreq(len(t), d=1/fs)  # frequency array
spec = np.abs(np.fft.rfft(sig))/len(f) # spectral amplitudes
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```

Generating Sawtooth Waveform (W2)

In [ ]:
```
# waveform generation
ns = 64              #samples per cycle
freq = 1000          #freq
amp = 1              #amplitude
offset = 0           #dc offset
cmd_gensin = 's211' #sawtooth waveform at W2
cmd_gensin += str(ns).zfill(3) + str(freq).zfill(7) + str(int(amp*100)).zfill(4)
device.write(bytes(cmd_gensin, 'utf-8'))
```

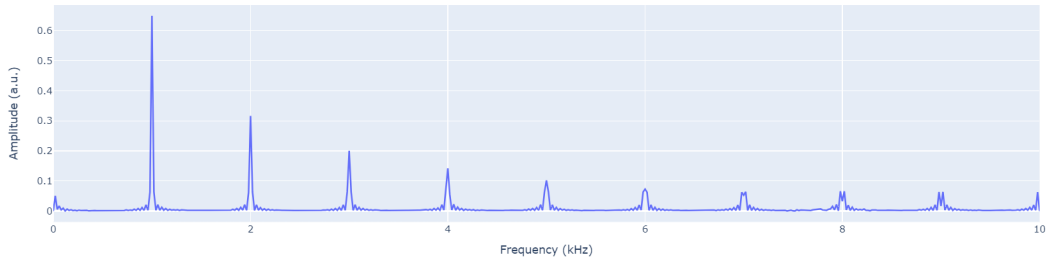
Out[ ]: 23

In [ ]:
```
bytedata = bytearray(4000)
device.reset_input_buffer()
device.write(bytes(cmd_readosc, 'utf-8'))
device.readline()
device.readinto(bytedata)
data = np.frombuffer(bytedata, dtype='uint16').reshape((2, 1000))
t = np.arange(1000)/fs
sig = 2*(data[1, :] - np.mean(data[1, :]))/np.ptp(data[1, :]) # adjust
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=sig))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (V)')
```



In [ ]:
```
f = np.fft.rfftfreq(len(t), d=1/fs)  # frequency array
spec = np.abs(np.fft.rfft(sig))/len(f) # spectral amplitudes
fig = go.Figure()
```

```
fig.add_trace(go.Scatter(x=f*1e-3, y=spec))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```



Similarities and Differences of Sinusoidal, Triangular and Sawtooth waveform spectra

Similarities

- All waveforms are periodic and have fundamental frequency

Differences

Sinusoidal

- Single peak at 1kHZ (Fundamental Frequency)
- No harmonic present in a pure sinusoidal waveform

Triangular

- The triangular waveform has odd harmonics that decrease in magnitude at a rate of $1/n^2$, where n is the harmonic number.
- Its spectrum consists of the fundamental frequency and odd harmonics with decreasing amplitudes.

Sawtooth

- Unlike the triangular waveform, the sawtooth waveform includes both odd and even harmonics.
- Its spectrum consists of the fundamental frequency and all integer harmonics, both odd and even, with decreasing amplitudes as the harmonic number increases.

Ammplitude modulation and its influence on spectrum

Generating Sinusoidal Waveform

In [ ]:
```
cmd_gensin = 's100'
cmd_gensin += str(ns).zfill(3) + str(freq).zfill(7) + str(int(amp*100)).zfill(4)
device.write(bytes(cmd_gensin, 'utf-8'))
```

Out[ ]: 23

Detection at CH1

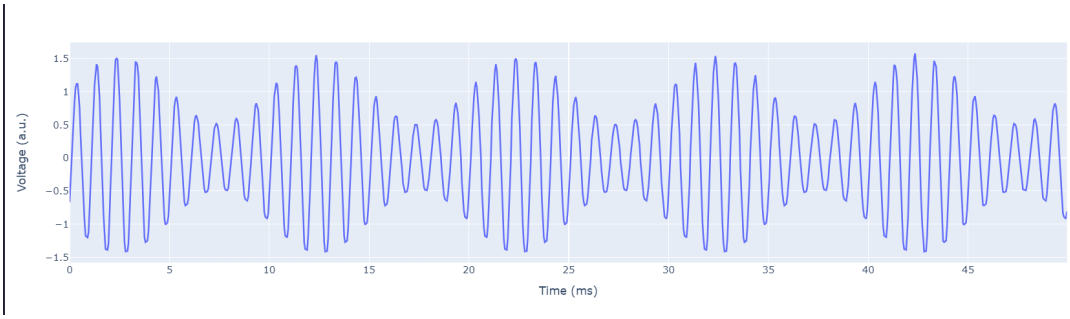
In [ ]:
```
# waveform detection
device.reset_input_buffer()
device.write(bytes(cmd_readosc, 'utf-8'))
device.readline()
device.readinto(bytedata)
data=np.frombuffer(bytedata, dtype='uint16').reshape((2, 1000))
t = np.arange(1000)/fs
sig = 2*(data[0, :] - np.mean(data[0, :]))/np.ptp(data[0, :])
```

Amplitude Modulation

Modulation Freq = 100Hz

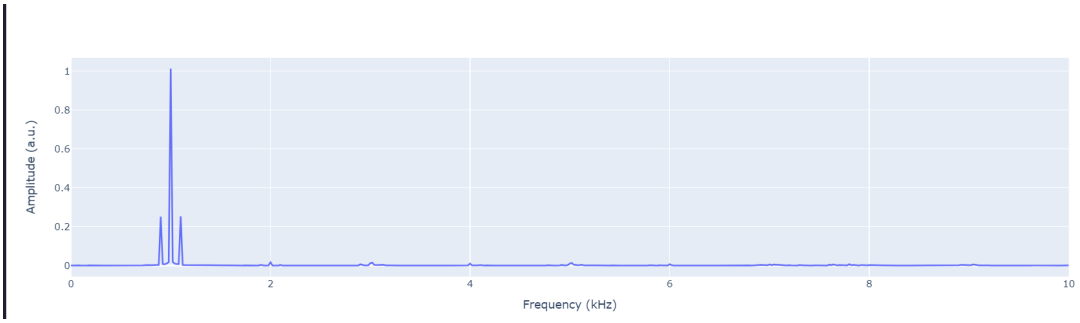
Modulation Depth = 0.5

In [ ]:
```
# amplitude modulation
mod_freq = 100
mod_depth = 0.5
mod = sig*(1 + mod_depth*np.sin(2*np.pi*mod_freq*t))
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=mod))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (a.u.)')
```



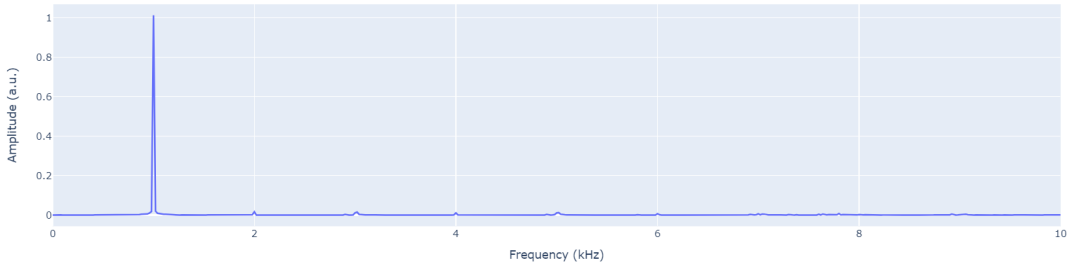
Spectral Analysis

In [ ]:
```
# spectral analysis
spec_mod = abs(np.fft.rfft(mod))/len(f)
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec_mod))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```

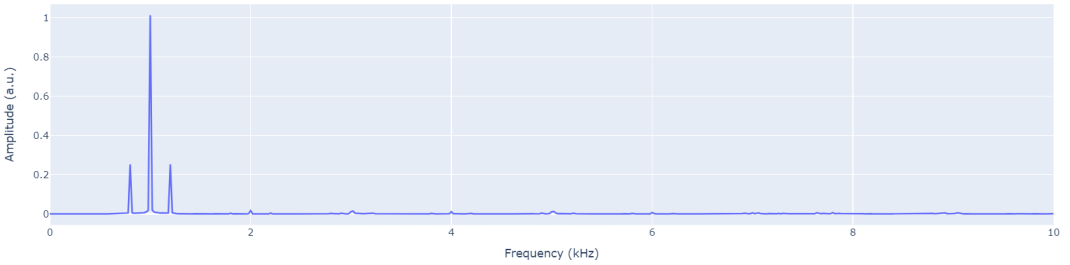


Change in modulation freq and depth

In [ ]:
```
mod_freq = 0    #frequency = 0Hz
mod_depth = 0.5
mod = sig*(1 + mod_depth*np.sin(2*np.pi*mod_freq*t))
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=mod))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (a.u.)')
# spectral analysis
spec_mod = abs(np.fft.rfft(mod))/len(f)
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec_mod))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```

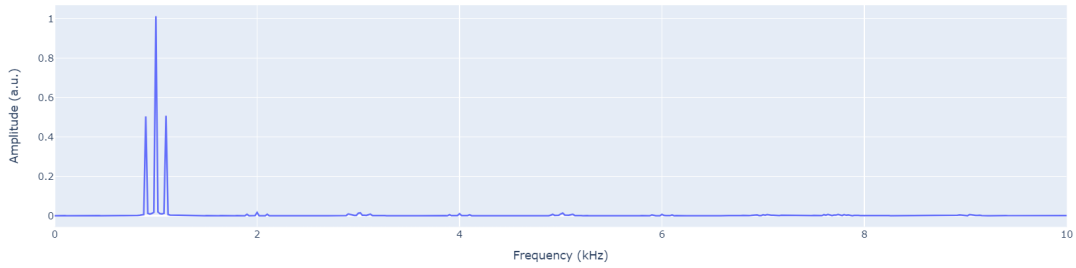


In [ ]:
```
mod_freq = 200   #frequency = 200Hz
mod_depth = 0.5
mod = sig*(1 + mod_depth*np.sin(2*np.pi*mod_freq*t))
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=mod))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (a.u.)')
# spectral analysis
spec_mod = abs(np.fft.rfft(mod))/len(f)
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec_mod))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```

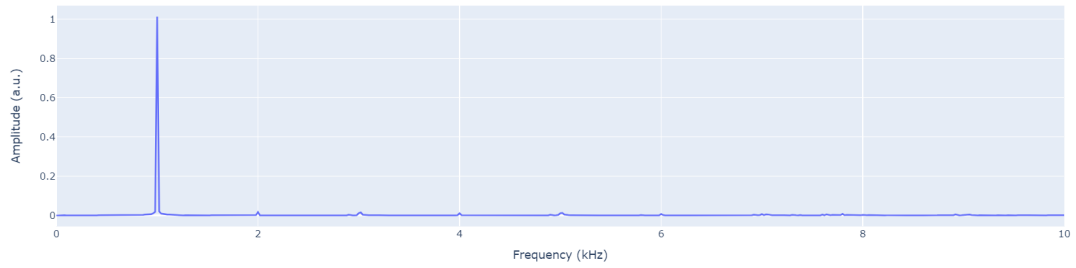


In [ ]:
```
mod_freq = 100    #frequency = 100Hz
mod_depth = 1   #depth = 1
mod = sig*(1 + mod_depth*np.sin(2*np.pi*mod_freq*t))
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=mod))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (a.u.)')
# spectral analysis
spec_mod = abs(np.fft.rfft(mod))/len(f)
fig = go.Figure()
```

```
fig.add_trace(go.Scatter(x=f*1e-3, y=spec_mod))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```



In [ ]:
```
mod_freq = 100   #frequency = 100Hz
mod_depth = 0    #depth = 0
mod = sig*(1 + mod_depth*np.sin(2*np.pi*mod_freq*t))
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=mod))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (a.u.)')
# spectral analysis
spec_mod = abs(np.fft.rfft(mod))/len(f)
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec_mod))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```



Observation in the effects of Modulation Frequency and Modulation Depth

Modulation Frequency

- Higher frequency:
  - broader bandwidth of modulated signal resulting in side bands further apart in the spectra
- Lower Frequency:
  - Lower modulation frequencies concentrate the sidebands closer to the carrier frequency

Modulation Depth

- Increasing the modulation depth increases the amplitude of the sidebands relative to the carrier.
  - This results in a more pronounced presence of the sidebands in the spectrum.
- At very high modulation depths, the modulation signal can distort the carrier waveform, leading to over-modulation effects and distortion in the demodulated signal.

To summarise, both modulation frequency and modulation depth influence the spectral properties of an AM signal. The modulation frequency regulates the spacing and distribution of the sidebands, whilst the modulation depth controls the relative amplitudes of the carrier and sideband. Balancing these factors is critical for efficient spectrum consumption and accurate signal reproduction in AM systems.

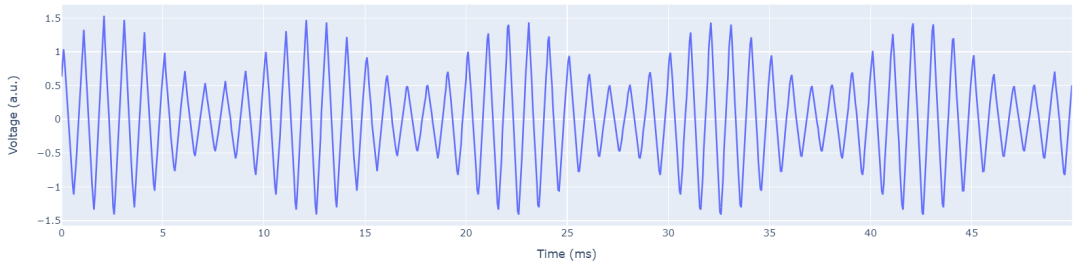
Amplitude Modulation on Triangular and Sawtooth

Triangular Waveform Amplitude Modulation (W1, CH1)

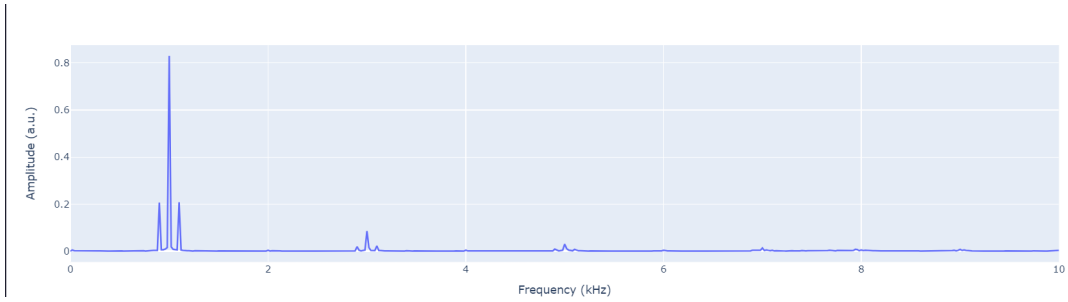
In [ ]:
```
cmd_gensin = 's110' #Triangular at W1
cmd_gensin += str(ns).zfill(3) + str(freq).zfill(7) + str(int(amp*100)).zfill(4)
device.write(bytes(cmd_gensin, 'utf-8'))

# waveform detection
device.reset_input_buffer()
device.write(bytes(cmd_readosc, 'utf-8'))
device.readline()
device.readinto(bytedata)
data=np.frombuffer(bytedata, dtype='uint16').reshape((2, 1000))
t = np.arange(1000)/fs
sig = 2*(data[0, :] - np.mean(data[0, :]))/np.ptp(data[0, :]) #CH1

# amplitude modulation
mod_freq = 100
mod_depth = 0.5
mod = sig*(1 + mod_depth*np.sin(2*np.pi*mod_freq*t))
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=mod))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (a.u.)')
```



In [ ]:
```
# spectral analysis
spec_mod = abs(np.fft.rfft(mod))/len(f)
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec_mod))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```



Triangular Waveform Amplitude Modulation (W1, CH1)

In [ ]:
```
cmd_gensin = 's211' #Sawtooth at W2
cmd_gensin += str(ns).zfill(3) + str(freq).zfill(7) + str(int(amp*100)).zfill(4)
device.write(bytes(cmd_gensin, 'utf-8'))

# waveform detection
device.reset_input_buffer()
device.write(bytes(cmd_readosc, 'utf-8'))
device.readline()
device.readinto(bytedata)
data=np.frombuffer(bytedata, dtype='uint16').reshape((2, 1000))
t = np.arange(1000)/fs
sig = 2*(data[1, :] - np.mean(data[1, :]))/np.ptp(data[1, :]) #CH2

# amplitude modulation
mod_freq = 100
mod_depth = 0.5
mod = sig*(1 + mod_depth*np.sin(2*np.pi*mod_freq*t))
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=mod))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (a.u.)')
```

In [ ]:
```
# spectral analysis
spec_mod = abs(np.fft.rfft(mod))/len(f)
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec_mod))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```

When sinusoidal, triangular, and sawtooth waveforms are employed as modulation signals in amplitude modulation (AM), they have various effects on the time-domain signals and the modulated signals' spectra.

Time Signal Plots:

1. **Sinusoidal Modulation**:
   - When a sinusoidal waveform is used for modulation, the resulting AM signal exhibits smooth variations in amplitude over time.
   - The amplitude of the carrier signal varies sinusoidally in accordance with the modulating sinusoid.
   - The time signal appears as a smooth waveform with amplitude variations following the envelope of the modulating sinusoid.
2. **Triangular Modulation**:

- Triangular modulation results in an AM signal where the carrier amplitude varies in a triangular pattern.
- The modulation waveform influences the linear rise and fall of the carrier amplitude.

3. **Sawtooth Modulation**:

- Sawtooth modulation leads to an AM signal where the carrier amplitude follows a sawtooth pattern.
- The carrier amplitude rises linearly and then abruptly falls before rising again in a repetitive manner.

Spectral Plots:

1. **Sinusoidal Modulation**:

- The spectral plot of sinusoidal modulation exhibits a central carrier peak surrounded by two symmetrical sidebands.

2. **Triangular Modulation**:

- The spectrum of triangular modulation consists of the carrier peak surrounded by sidebands.
- Sidebands are distributed according to the harmonics of the triangular modulation waveform.
- The spectrum contains odd harmonics due to the nature of the triangular waveform.

3. **Sawtooth Modulation**:

- The spectral plot of sawtooth modulation also shows the carrier peak and sidebands.
- Unlike the triangular modulation, the sawtooth modulation produces both odd and even harmonics.
- The spectrum consists of a rich harmonic structure due to the sawtooth waveform.

Similarities:

- All modulation waveforms produce AM signals with a carrier and sidebands in their spectra.
- The carrier frequency remains unchanged in all cases, while the sidebands represent the modulation frequency and its harmonics.

Differences:

- The shape of the modulation waveform influences the pattern of amplitude variations in the time signal.
- The spectral content of the modulated signal varies depending on the waveform used for modulation, with different distributions of harmonics and sidebands.

In conclusion, whereas sinusoidal, triangular, and sawtooth waveforms produce comparable AM signals in terms of carrier and sidebands, their time-domain and spectral properties differ due to the modulation waveforms' unique shapes and harmonic content.

Open-ended Question

Generate a 1 kHz sinusoidal signal with 8 samples per cycle. Measure the signal using CH1 at a sampling frequency of 100 kHz. Repeat the measurement with a 20 kHz sinusoidal signal with the same number of samples per cycle. Comment on the results.

1kHz Sinusoidal

In [ ]:
```
# waveform generation
ns = 8
freq = 1000
amp = 1
offset = 0
cmd_gensin = 's100' #W1
cmd_gensin += str(ns).zfill(3) + str(freq).zfill(7) + str(int(amp*100)).zfill(4)
device.write(bytes(cmd_gensin, 'utf-8'))
```

Out[ ]: 23

In [ ]:
```
# waveform detection and plotting
fs = 100000
c2 = 0
gain1 = 138
dco1 = 130
c1 = 0
gain2 = 138
dco2 = 130
cmd_readosc = 'm1' + str(fs).zfill(6) + str(c1) + str(gain1).zfill(3) + str(dco1
bytedata = bytearray(4000)
device.reset_input_buffer()
device.write(bytes(cmd_readosc, 'utf-8'))
device.readline()
device.readinto(bytedata)
data = np.frombuffer(bytedata, dtype='uint16').reshape((2, 1000))
t = np.arange(1000)/fs
sig = 2*(data[0, :] - np.mean(data[0, :]))/np.ptp(data[0, :]) # adjust
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=sig))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (V)')
```

In [ ]:
```
# amplitude modulation
mod_freq = 100
mod_depth = 0.5
mod = sig*(1 + mod_depth*np.sin(2*np.pi*mod_freq*t))
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=mod))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (a.u.)')
```

In [ ]:
```
# spectral analysis
spec_mod = abs(np.fft.rfft(mod))/len(f)
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec_mod))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```

20kHz Sinusoidal

In [ ]:
```
# waveform generation
ns = 8
freq = 20000
amp = 1
offset = 0
cmd_gensin = 's200' #W2
cmd_gensin += str(ns).zfill(3) + str(freq).zfill(7) + str(int(amp*100)).zfill(4)
device.write(bytes(cmd_gensin, 'utf-8'))
```

Out[ ]: 23

In [ ]:
```
# waveform detection and plotting
fs = 100000
c2 = 0
gain1 = 138
dco1 = 130
c1 = 0
gain2 = 138
dco2 = 130
cmd_readosc = 'm1' + str(fs).zfill(6) + str(c1) + str(gain1).zfill(3) + str(dco1
bytedata = bytearray(4000)
device.reset_input_buffer()
device.write(bytes(cmd_readosc, 'utf-8'))
device.readline()
device.readinto(bytedata)
data = np.frombuffer(bytedata, dtype='uint16').reshape((2, 1000))
t = np.arange(1000)/fs
sig = 2*(data[1, :] - np.mean(data[1, :]))/np.ptp(data[1, :]) # adjust
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=sig))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (V)')
```

In [ ]:
```
# amplitude modulation
mod_freq = 100
mod_depth = 0.5
mod = sig*(1 + mod_depth*np.sin(2*np.pi*mod_freq*t))
fig = go.Figure()
fig.add_trace(go.Scatter(x=t*1e3, y=mod))
fig.update_layout(xaxis_title='Time (ms)', yaxis_title='Voltage (a.u.)')
```

In [ ]:
```
# spectral analysis
spec_mod = abs(np.fft.rfft(mod))/len(f)
fig = go.Figure()
fig.add_trace(go.Scatter(x=f*1e-3, y=spec_mod))
fig.update_layout(xaxis_title='Frequency (kHz)', yaxis_title='Amplitude (a.u.)')
```

The Nyquist-Shannon sampling theorem states that to accurately reconstruct a signal, the sampling frequency must be at least twice the highest frequency present in the signal. . So, a sampling frequency of 100 kHz should be more than sufficient to capture the 1 kHz / 20kHz sine wave without aliasing.

With only 8 samples per cycle, There is not enough capturing points to accurately represent the sinusoidal waveform. This leads to distortion and an inaccurate depiction of the sine wave. With such few samples per cycle, any attempt to reconstruct the waveform between the sampled points would introduce interpolation errors, further distorting the shape of the waveform.