Universidad Nacional de Entre Ríos

Facultad de Ingeniera

Tarea análisis Tiempo-Frecuencia

Señales y Sistemas

Nazarena Emilce Romero
nazarena.romero@ingenieria.uner.edu.ar

Andres Antonio Venialgo
andres.venialgo@ingenieria.uner.edu.ar

2. Señal chirp

```python
import numpy as np
import scipy.io.wavfile as wavfile
import matplotlib.pyplot as plt
import scipy.signal
from IPython.display import Audio, display

# Parámetros de la señal
duration = 8.0  # segundos
fs = 44100 # Hz frecuencia de muestreo
t = np.linspace(0, duration, int(fs * duration), endpoint=False) # vector de tiempo

# Frecuencias para la chirp cuadratica: empieza alta, baja al centro, vuelve alta
f_high = 15000.0 # frecuencia en los extremos (Hz)
f_low  = 500.0   # frecuencia mínima en el centro (Hz)

half = duration / 2
# coeficiente cuadrático a tal que f(0)=f_high, f(half)=f_low, f(duration)=f_high
a = (f_high - f_low) / (half ** 2)

# frecuencia instantánea f(t) = a*(t-half)^2 + f_low
f_t = a * (t - half) ** 2 + f_low

# fase: integral de f(t) -> (t) = 2 * ( a*(t-half)^3/3 + f_low * t )
phase = 2 * np.pi * (a * (t - half) ** 3 / 3.0 + f_low * t)

# señal
audio_signal = 0.9 * np.sin(phase)  # amplitud <1 para evitar clipping
audio_signal_float = audio_signal / np.max(np.abs(audio_signal))
audio_signal_int16 = np.int16(audio_signal_float * 32767)

# guardar WAV
wavfile.write("chirp_u_shape.wav", fs, audio_signal_int16)

# Visualización: forma de onda (decimada) + zoom + espectrograma
decimate_target = 5000
N = audio_signal_float.size
dec_step = max(1, int(N / decimate_target))
t_dec = t[::dec_step]
sig_dec = audio_signal_float[::dec_step]

# zoom en ventana pequeña para ver ciclo
zoom_start = 3.9          # segundos
zoom_dur = 0.05           # 50 ms
i0 = int(zoom_start * fs)
i1 = int(min(N, i0 + zoom_dur * fs))
t_zoom = t[i0:i1]
sig_zoom = audio_signal_float[i0:i1]

fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(10, 9), gridspec_kw={'height_ratios':[1,1,1.2]})
ax1.plot(t_dec, sig_dec, lw=0.5)
ax1.set_title("Forma de onda (completa, decimada)")
```

```
ax1.set_xlim(0, duration)
ax1.set_ylabel("Amplitud")

ax2.plot(t_zoom, sig_zoom, lw=0.8)
ax2.set_title(f"Zoom {zoom_start:.3f}s → {zoom_start+zoom_dur:.3f}s")
ax2.set_ylabel("Amplitud")

frequencies, times, Sxx = scipy.signal.spectrogram(audio_signal_float, fs=fs, nperseg=1024)
pcm = ax3.pcolormesh(times, frequencies, 10 * np.log10(Sxx + 1e-12), shading='gouraud')
ax3.set_ylim(0, min(16000, fs/2))
ax3.set_title("Espectrograma (U esperado: alta-baja-alta)")
ax3.set_xlabel("Tiempo (s)")
ax3.set_ylabel("Frecuencia (Hz)")
fig.colorbar(pcm, ax=ax3, label='dB')

plt.tight_layout()
plt.show()

display(Audio(data=audio_signal_float, rate=fs))
```

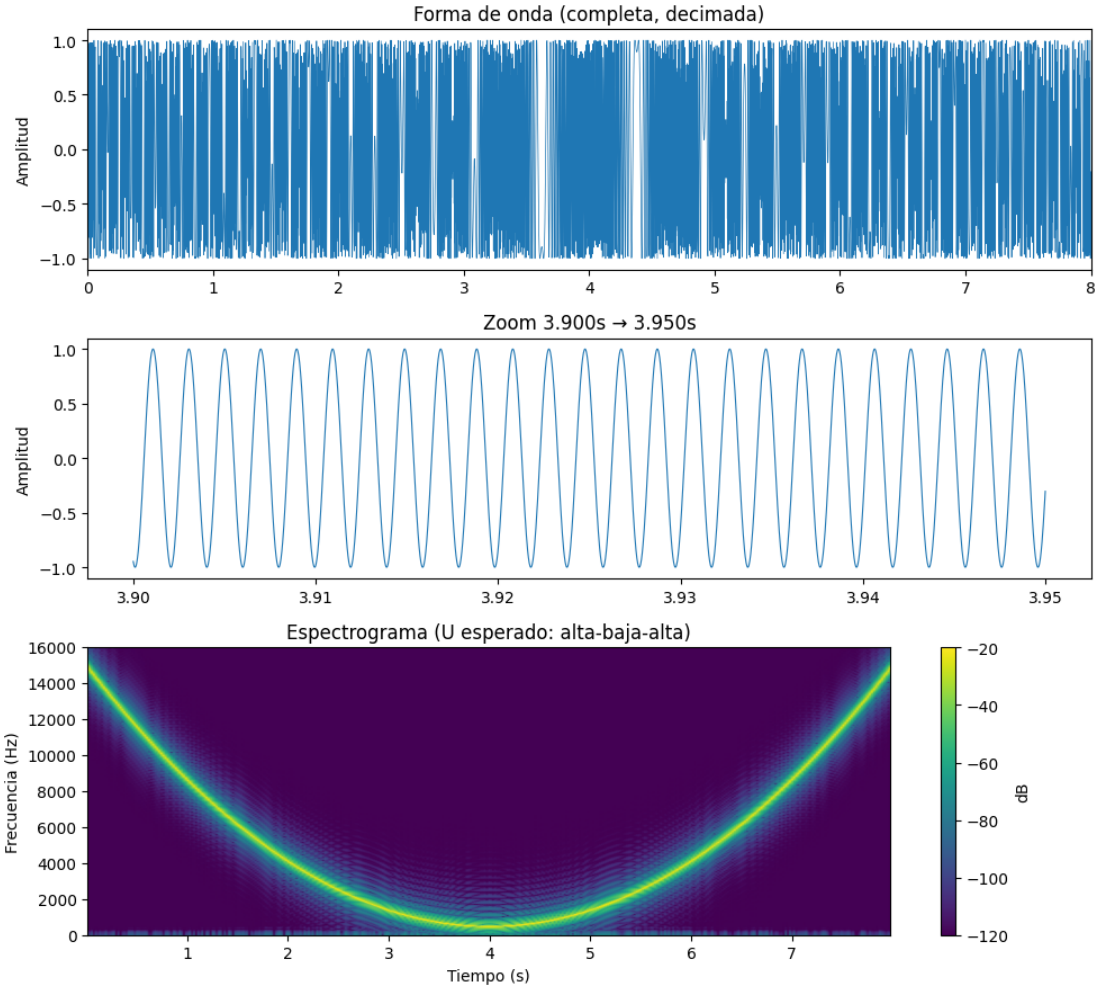

Figura 1: a)Audiograma; b)Zoom; c)Espectograma