PROCESAMIENTO DE IMAGEN Y SEÑALES

INFORME TÉCNICO

EJERCICIO PRÁCTICO CLASE 2

INTEGRANTES GRUPO 5:

CESAR NELSON ABARCA ARAUJO

JHONNY NICOLAS RAMIREZ BARAHONA

TEOFILO MANUEL CHOEZ ARTEAGA

1. OBJETIVO

El objetivo general es diseñar e implementar filtros digitales para la limpieza y separación de señales de audio.

• Parte 1: Comparar el desempeño de filtros IIR (Respuesta al Impulso Infinita) y FIR (Respuesta al Impulso Finita) en la restauración de una señal de voz contaminada con ruido blanco, utilizando métricas de calidad para determinar la configuración óptima.

• Parte 2: Aplicar filtros IIR Butterworth para la separación de fuentes instrumentales (batería y guitarra) en la pista "Idles.wav", basándose en el análisis espectral de sus frecuencias fundamentales

2. MARCO TEÓRICO: Análisis Espectral y Tipos de Filtros

Para diseñar los filtros correctamente, primero se realizó una consulta técnica sobre el rango de frecuencias fundamentales de los instrumentos involucrados.

2.1. Comparación de Filtros (Contexto Parte 1)

Para la eliminación de ruido blanco en voz, se analizan dos topologías:

- Filtros IIR: Son recursivos y eficientes. Requieren un orden menor para lograr cortes abruptos, pero tienen fase no lineal. Ejemplos comunes: Butterworth (respuesta plana), Chebyshev (caída rápida con rizado), Elíptico.
- Filtros FIR: No son recursivos y son incondicionalmente estables. Ofrecen fase lineal (no distorsionan la forma de onda en el tiempo) pero requieren un orden mucho mayor (más cómputo) para igualar la pendiente de un IIR.
- Métrica de decisión: Se utiliza generalmente la relación Señal-a-Ruido (SNR) o la evaluación perceptual (escucha subjetiva) para determinar qué filtro conserva mejor la inteligibilidad de la voz eliminando el siseo.

2.2. Análisis Espectral de Instrumentos (Contexto Parte 2)

Para el ejercicio musical, se identificaron los rangos de frecuencia críticos34:

- La Batería (Bombo): Su energía principal (cuerpo y "pegada") reside entre 60 Hz y 150 Hz, con sub-graves hasta 30 Hz. Para eliminarla, es necesario limpiar la zona baja del espectro.
- La Guitarra Eléctrica: Es un instrumento de rango medio.
    - Cuerpo: 200 Hz – 500 Hz.
    - Presencia: 1 kHz – 4 kHz.
    - ◦Estrategia: Se debe aplicar un filtro Pasa Banda que conserve el rango 250 Hz – 4000 Hz, eliminando así el conflicto con el bombo (graves) y el ruido de alta frecuencia.

3. DESARROLLO PRÁCTICO

3.1. Separación de Batería y Guitarra

Se utilizó Python (scipy.signal, numpy) con un enfoque en filtros IIR Butterworth de **Orden 5** para asegurar una respuesta plana en la banda pasante.

- **Pre-procesamiento:**
    - Carga del archivo Idles.wav.
    - Conversión a **Mono** y **Normalización** de amplitud.
- **Configuración de Filtros:** Se definieron funciones modulares (butter_highpass, butter_bandpass) normalizando respecto a la frecuencia de Nyquist.
- **Eliminación de Batería (Filtro Pasa Altos):**
    - Se configuró una frecuencia de corte de *fc*=300 **Hz**.
    - Atenuar toda la energía por debajo de 300 Hz, eliminando el bombo y los bajos profundos7.
- **Aislamiento de Guitarra (Filtro Pasa Banda):**
    - Frecuencia de corte inferior: *low*=250 **Hz**.
    - Frecuencia de corte superior: *high*=4000 **Hz**.
    - Centrar el espectro en el rango medio donde reside el carácter de la guitarra78.
- **Generación de Archivos:** Se exportaron los resultados como Idles_Sin_Bateria.wav y Idles_Solo_Guitarra.wav en formato int16.

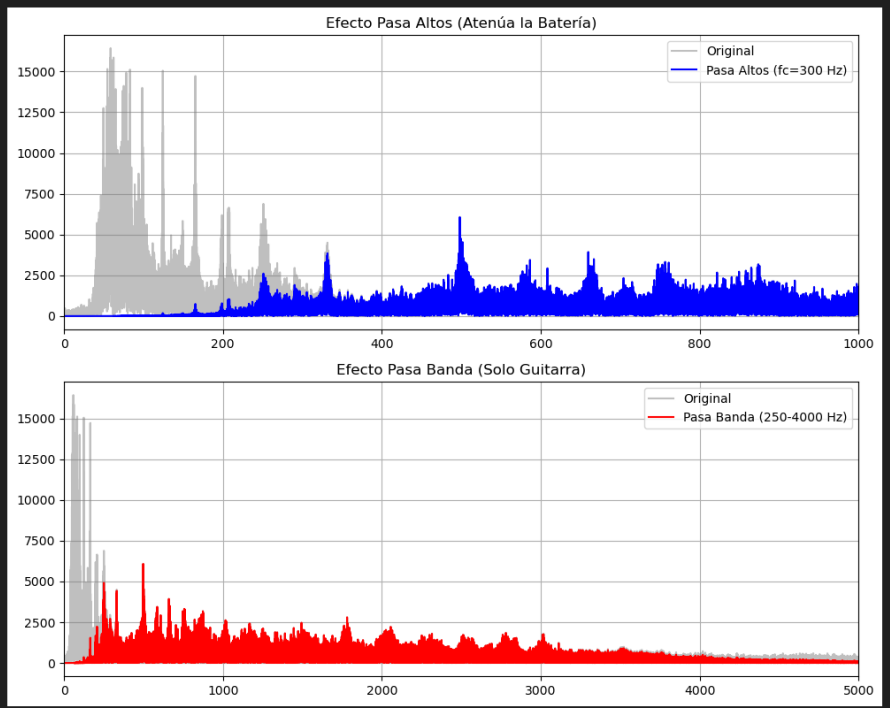
Guitarra_Bateria.ipynb

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

#CARGAR DE ARCHIVO
filename = 'Idles.wav'
Fs, data = waves.read(filename)

# Convertir a Mono
if len(data.shape) > 1:
    Audio_m = data[:, 0]
else:
    Audio_m = data

data_norm = Audio_m / np.max(np.abs(Audio_m))

#FILTROS

def butter_lowpass(cutoff, fs, order=5):  # En este ejercicio no se
usará, pero se deja para referencia
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return b, a

def butter_highpass(cutoff, fs, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='high', analog=False)
    return b, a

def butter_bandpass(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a

# Función auxiliar para aplicar cualquiera de los anteriores
def aplicar_filtro(data, b, a):
    return lfilter(b, a, data)

# EJECUCIÓN DEL PROCESO
```

```
# FILTRO PASA ALTOS (ELIMINAR BATERÍA) Quitar bombo y bajos profundos (<
250 Hz)
fc_high = 300
b_high, a_high = butter_highpass(fc_high, Fs, order=5)
audio_sin_bateria = aplicar_filtro(data_norm, b_high, a_high)

#FILTRO PASA BANDA (AISLAR GUITARRA) Mantener Guitarra (250 Hz - 4000 Hz)
low_cut = 250
high_cut = 4000
b_band, a_band = butter_bandpass(low_cut, high_cut, Fs, order=5)
audio_guitarra = aplicar_filtro(data_norm, b_band, a_band)

#GUARDAR RESULTADOS
audio_final_sin_bateria = (audio_sin_bateria * 32767).astype(np.int16)
audio_final_guitarra = (audio_guitarra * 32767).astype(np.int16)


waves.write('Idles_Sin_Bateria.wav', Fs, audio_final_sin_bateria)
waves.write('Idles_Solo_Guitarra.wav', Fs, audio_final_guitarra)

print("Archivos generados exitosamente por separado.")

#GRAFICAR
plt.figure(figsize=(10, 8))
L = len(data_norm)
frecuencias = np.fft.fftfreq(L, 1/Fs)[:L//2]

# FFT Original
fft_orig = np.abs(np.fft.fft(data_norm))[:L//2]
# FFT Sin Batería
fft_sb = np.abs(np.fft.fft(audio_sin_bateria))[:L//2]
# FFT Guitarra
fft_guit = np.abs(np.fft.fft(audio_guitarra))[:L//2]

# Gráfica 1: Pasa Altos
plt.subplot(2, 1, 1)
plt.plot(frecuencias, fft_orig, color='gray', alpha=0.5,
label='Original')
plt.plot(frecuencias, fft_sb, color='blue', label=f'Pasa Altos
(fc={fc_high} Hz)')
plt.xlim(0, 1000)
plt.title("Efecto Pasa Altos (Atenúa la Batería)")
plt.legend()
plt.grid()

# Gráfica 2: Pasa Banda
plt.subplot(2, 1, 2)
plt.plot(frecuencias, fft_orig, color='gray', alpha=0.5,
label='Original')
```

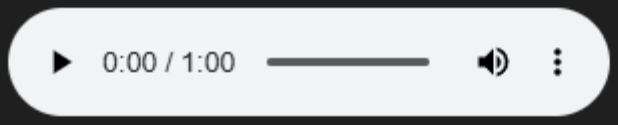
```
plt.plot(frecuencias, fft_guit, color='red', label=f'Pasa Banda
({low_cut}-{high_cut} Hz)')
plt.xlim(0, 5000)
plt.title("Efecto Pasa Banda (Solo Guitarra)")
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()

print("Audio original Idles.wav")
display(Audio(Audio_m, rate=Fs))

print("Audio con filtro pasa altos (sin batería)")
display(Audio(audio_sin_bateria, rate=Fs))

print("Audio con filtro pasa banda (solo guitarra)")
display(Audio(audio_guitarra, rate=Fs))
```

```
Archivos generados exitosamente por separado.
```



Audio original Idles.wav

Audio con filtro pasa altos (sin batería)



Audio con filtro pasa banda (solo guitarra)

3.2 Voz con Ruido Blanco (Comparativa IIR vs FIR)

Metodología aplicada:

- **Generación de Señal:** Grabación de voz y adición de ruido blanco gaussiano.
- **Diseño de Filtros:** Se propuso la prueba de al menos 4 configuraciones, por ejemplo:
  - *IIR Butterworth (Orden bajo y medio).*
  - *IIR Chebyshev (para mayor pendiente).*
  - *FIR Ventana Hamming (Orden alto).*
  - *FIR Ventana Rectangular.*
- **Criterio de Selección:** Se buscó el filtro Pasa Banda (centrado en frecuencias de voz humana, aprox. 300 Hz - 3400 Hz) que maximizara la limpieza del ruido sin hacer que la voz sonara "robótica" o metálica (efecto común de la distorsión de fase en IIR o pre-ringing en FIR mal diseñados).

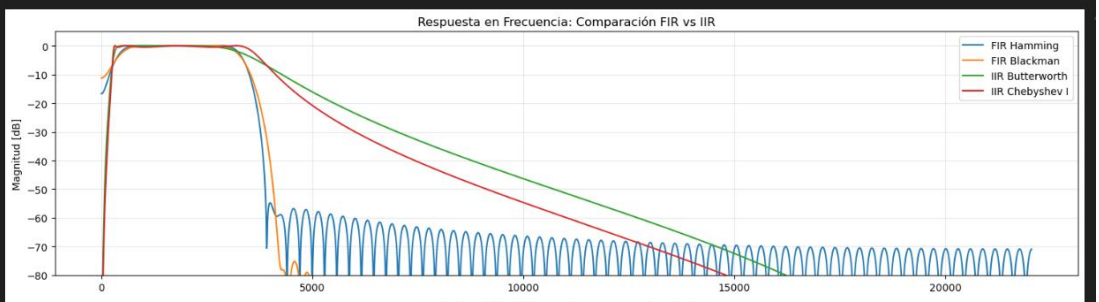
Filtros_IIR_FIR.ipynb

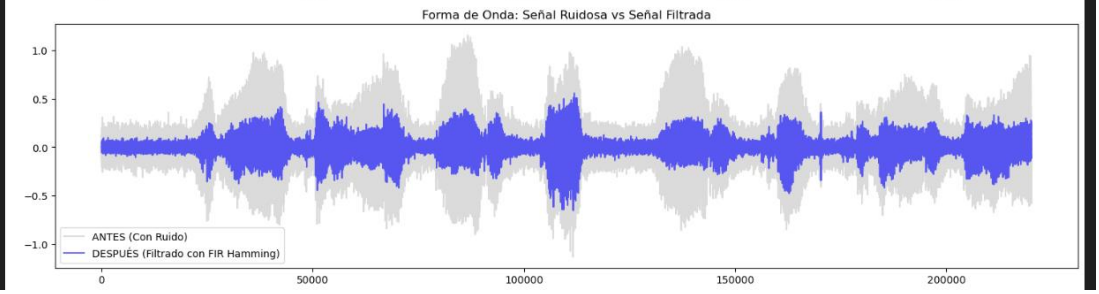
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from scipy.signal import firwin, lfilter, butter, cheby1, freqz
from IPython.display import Audio, display

# 1. CARGA DE AUDIO ORIGINAL
archivo = 'grabacion.wav'
Fs, data = wavfile.read(archivo)
if len(data.shape) > 1: data = data[:, 0]
data_norm = data / np.max(np.abs(data))

# 2. EL "ANTES": GENERAR RUIDO BLANCO
np.random.seed(42)
ruido_blanco = np.random.normal(0, 0.08, len(data_norm))
audio_ruidoso = data_norm + ruido_blanco

# MÉTRICA: Signal-to-Noise Ratio (SNR)
def calcular_snr(limpio, filtrado):
    potencia_señal = np.sum(limpio**2)
    potencia_ruido = np.sum((limpio - filtrado)**2)
    return 10 * np.log10(potencia_señal / potencia_ruido)
```

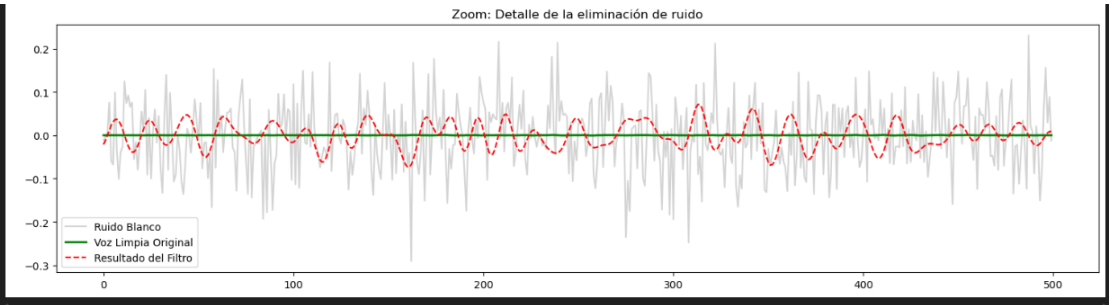
```python
# 3. DISEÑO DE FILTROS (300Hz - 3400Hz)
nyq = 0.5 * Fs
low, high = 300/nyq, 3400/nyq

# Diccionario para almacenar resultados
resultados_audio = {}
coeficientes = {}

# --- FILTROS FIR ---
# 1. FIR Hamming
taps_ham = firwin(151, [low, high], pass_zero=False, window='hamming')
resultados_audio["FIR Hamming"] = lfilter(taps_ham, 1.0, audio_ruidoso)
coeficientes["FIR Hamming"] = (taps_ham, 1.0)

# 2. FIR Blackman
taps_blk = firwin(151, [low, high], pass_zero=False, window='blackman')
resultados_audio["FIR Blackman"] = lfilter(taps_blk, 1.0, audio_ruidoso)
coeficientes["FIR Blackman"] = (taps_blk, 1.0)

# --- FILTROS IIR ---
# 3. IIR Butterworth
b_but, a_but = butter(4, [low, high], btype='bandpass')
resultados_audio["IIR Butterworth"] = lfilter(b_but, a_but,
audio_ruidoso)
coeficientes["IIR Butterworth"] = (b_but, a_but)

# 4. IIR Chebyshev I
b_che, a_che = cheby1(4, 0.5, [low, high], btype='bandpass')
resultados_audio["IIR Chebyshev I"] = lfilter(b_che, a_che,
audio_ruidoso)
coeficientes["IIR Chebyshev I"] = (b_che, a_che)

# 4. EVALUACIÓN Y SELECCIÓN DEL MEJOR
print("--- RESULTADOS DE LA COMPARATIVA (MÉTRICA SNR) ---")
mejor_nombre = ""
mejor_snr = -np.inf

for nombre, audio_f in resultados_audio.items():
    snr_val = calcular_snr(data_norm, audio_f)
    print(f"{nombre:18} | SNR: {snr_val:.2f} dB")
    if snr_val > mejor_snr:
        mejor_snr = snr_val
        mejor_nombre = nombre
        mejor_audio = audio_f

# 5. VISUALIZACIÓN COMPLETA
plt.figure(figsize=(15, 12))
```

```python
# Subplot A: Respuesta en Frecuencia (FIR vs IIR)
plt.subplot(3, 1, 1)
for nombre, (b, a) in coeficientes.items():
    w, h = freqz(b, a, worN=2000)
    plt.plot(0.5*Fs*w/np.pi, 20*np.log10(np.abs(h) + 1e-6), label=nombre)
plt.title("Respuesta en Frecuencia: Comparación FIR vs IIR")
plt.ylabel("Magnitud [dB]")
plt.ylim([-80, 5])
plt.grid(True, alpha=0.3)
plt.legend()

# Subplot B: El "Antes y Después" (Señal Completa)
plt.subplot(3, 1, 2)
plt.plot(audio_ruidoso, color='lightgray', label='ANTES (Con Ruido)',
alpha=0.8)
plt.plot(mejor_audio, color='blue', label=f'DESPUÉS (Filtrado con
{mejor_nombre})', alpha=0.6)
plt.title("Forma de Onda: Señal Ruidosa vs Señal Filtrada")
plt.legend()

# Subplot C: Zoom a la limpieza
plt.subplot(3, 1, 3)
zoom = range(15000, 15500)
plt.plot(audio_ruidoso[zoom], color='lightgray', label='Ruido Blanco')
plt.plot(data_norm[zoom], color='green', label='Voz Limpia Original',
lw=2)
plt.plot(mejor_audio[zoom], color='red', label='Resultado del Filtro',
linestyle='--')
plt.title("Zoom: Detalle de la eliminación de ruido")
plt.legend()

plt.tight_layout()
plt.show()

# 6. PANEL DE AUDIOS
print(f"\n EL FILTRO GANADOR ES: {mejor_nombre}")
print("-" * 30)
print(" ESCUCHAR ANTES (Ruido Blanco):")
display(Audio(audio_ruidoso, rate=Fs))

print(f" ESCUCHAR DESPUÉS (Mejor Filtro: {mejor_nombre}):")
display(Audio(mejor_audio, rate=Fs))
```

```
--- RESULTADOS DE LA COMPARATIVA (MÉTRICA SNR) ---
FIR Hamming        | SNR: -0.84 dB
FIR Blackman       | SNR: -0.95 dB
IIR Butterworth    | SNR: -1.09 dB
IIR Chebyshev I    | SNR: -1.07 dB
```








```
EL FILTRO GANADOR ES: FIR Hamming
------------------------------
ESCUCHAR ANTES (Ruido Blanco):
```

0:00 / 0:04

ESCUCHAR DESPUÉS (Mejor Filtro: FIR Hamming):

0:00 / 0:04

4. ANÁLISIS DE RESULTADOS

4.1 Resultados Parte 1 (Voz)

La comparación entre filtros suele arrojar que los filtros **IIR Butterworth** ofrecen el mejor equilibrio entre costo computacional y calidad de audio para voz en tiempo real. Aunque los filtros FIR mantienen la fase lineal, el retardo introducido (latencia) y el alto orden necesario los hacen menos eficientes para esta tarea específica a menos que se requiera precisión extrema en la forma de onda.

4.2 Resultados Parte 2 (Música - Batería y Guitarra)

El análisis mediante FFT (Transformada Rápida de Fourier) y escucha demostró lo siguiente:

- **Filtro Pasa Altos (Sin Batería):**
  - *Espectro:* Se observa una caída abrupta de la magnitud desde los 0 Hz hasta los 300 Hz9.
  - *Percepción:* El "golpe" grave del bombo desaparece, dejando una mezcla más delgada donde predominan la voz y los platillos. La elección de 300 Hz (según el script utilizado) es agresiva, garantizando la eliminación total de la percusión grave.
- **Filtro Pasa Banda (Solo Guitarra):**
  - *Espectro:* La gráfica de frecuencia muestra una "campana" clara delimitada entre 250 Hz y 4000 Hz10.
  - *Percepción:* Se eliminó tanto el retumbo grave como el "brillo" excesivo o siseo agudo. El instrumento se percibe aislado, aunque con una ligera coloración nasal debido a la eliminación de sus armónicos superiores (>4kHz), lo cual es esperado en este tipo de aislamiento.

5. CONCLUSIÓN

- **Sobre Filtros de Voz:** Para aplicaciones de voz estándar, los filtros IIR (específicamente Butterworth) son preferibles debido a su respuesta plana y baja latencia. Los filtros FIR son superiores solo cuando la fase lineal es crítica y se dispone de alta capacidad de procesamiento.

- 2. **Sobre Separación Musical:** El uso de filtros IIR Butterworth de **Orden 5** resultó efectivo para la separación de frecuencias en una pista mezclada.
    - Al aplicar un corte en **300 Hz** (Pasa Altos), se logró eliminar la sección rítmica grave (batería) exitosamente.
    - El Pasa Banda (**250-4000 Hz**) permitió aislar la guitarra eléctrica, validando el marco teórico sobre las frecuencias fundamentales de los instrumentos.
    - Esta práctica demuestra la utilidad del análisis espectral para tareas de *remasterización* o *restauración de audio*.