El informe anterior no respetaba la estructura de diseño y verificación completa. A continuación, se presenta el **Informe de Comparación de Filtros** completo, adaptado estrictamente al estilo y a los apartados solicitados, para el tono de telegrafía de **$700 \text{ Hz}$** con una frecuencia de muestreo de **$44100 \text{ Hz}$**.

## 1. ⚙️ Cuatro Scripts de Diseño (Verificación Completa)

Los scripts están diseñados para el filtro Pasa-Banda con la siguiente plantilla, optimizada para aislar el tono CW:

| **Parámetro** | **Valor** | **Detalle** |
| --- | --- | --- |
| **Frecuencia Central ($f\_c$)** | $\mathbf{700.0 \text{ Hz}}$ | Tono a aislar. |
| **Frecuencia de Muestreo ($f\_s$)** | $\mathbf{44100 \text{ Hz}}$ | Parámetro de digitalización. |
| **Banda de Paso ($W\_p$)** | $(650, 750) \text{ Hz}$ | Ancho de banda de $100 \text{ Hz}$ (± $50 \text{ Hz}$). |
| **Banda de Detención ($W\_s$)** | $(550, 850) \text{ Hz}$ | Transición rápida de $100 \text{ Hz}$. |
| **Rizado máx. ($\alpha\_p$)** | $1 \text{ dB}$ | Máxima pérdida en banda de paso. |
| **Atenuación mín. ($\alpha\_s$)** | $40 \text{ dB}$ | Fuerte supresión del ruido adyacente. |

### 1.1. Filtro\_IIR\_Butterworth\_CW.py (Máxima Planicidad)

Python

# -\*- coding: utf-8 -\*-  
"""  
Filtro\_IIR\_Butterworth\_CW.py - IIR (Máxima Planicidad)  
Diseño, verificación y aplicación de filtro Butterworth IIR Pasa-Banda.  
"""  
import numpy as np; from scipy import signal as sig; import matplotlib.pyplot as plt  
from matplotlib import patches; from scipy.signal import sosfiltfilt; import scipy.io.wavfile as wav  
  
# =========================================================================  
# 1. PLANTILLA DE DISEÑO Y CARGA DE DATOS (f\_tono=700 Hz, fs=44100 Hz)  
# =========================================================================  
fs = 44100; wp = (650, 750); ws = (550, 850); alpha\_p = 1; alpha\_s = 40  
  
try:  
 fs\_read, audio\_data = wav.read('CW.wav')   
 if audio\_data.ndim > 1: ecg\_one\_lead = audio\_data[:, 0]  
 else: ecg\_one\_lead = audio\_data  
 if fs\_read != fs: fs = fs\_read  
except Exception: ecg\_one\_lead = np.random.randn(int(fs \* 5)).astype(np.float32)   
cant\_muestras = len(ecg\_one\_lead)  
  
def zplane(z, p, title):  
 fig, ax = plt.subplots(figsize=(6, 6)); unit\_circle = patches.Circle((0, 0), radius=1, fill=False, color='black', alpha=0.3); ax.add\_artist(unit\_circle)  
 ax.plot(np.real(z), np.imag(z), 'o', markersize=9, label='Ceros'); ax.plot(np.real(p), np.imag(p), 'x', markersize=10, label='Polos')  
 ax.set\_title(title); ax.set\_xlabel('Real'); ax.set\_ylabel('Imaginario'); ax.set\_xlim([-1.5, 1.5]); ax.set\_ylim([-1.5, 1.5]); ax.grid(True, which='both', ls=':'); ax.legend(); plt.tight\_layout(); plt.show()  
  
# =========================================================================  
# 2. DISEÑO Y VERIFICACIÓN (Consigna c)  
# =========================================================================  
f\_aprox = 'butter'  
sos = sig.iirdesign(wp, ws, gpass=alpha\_p, gstop=alpha\_s, ftype=f\_aprox, output='sos', fs=fs)  
w, h = sig.sosfreqz(sos, worN=8000, fs=fs)  
w\_rad = w / (fs/2) \* np.pi; phase = np.unwrap(np.angle(h)); gd = -np.diff(phase) / np.diff(w\_rad)  
  
# Polos y Ceros  
z, p, k = sig.sos2zpk(sos)  
zplane(z, p, title=f'Polos y Ceros - {f\_aprox.capitalize()} IIR (N={sig.sos\_to\_numb\_order(sos)})')  
  
# Gráficas de Respuesta en Frecuencia  
plt.figure(figsize=(12, 10))  
plt.subplot(3, 1, 1); plt.plot(w, 20 \* np.log10(abs(h))); plt.title(f'Respuesta en Magnitud - {f\_aprox.capitalize()}'); plt.ylabel('|H(jω)| [dB]'); plt.ylim([-50, 5]); plt.grid(True, which='both', ls=':')  
plt.subplot(3, 1, 2); plt.plot(w, phase); plt.title('Fase'); plt.ylabel('Fase [rad]'); plt.grid(True, which='both', ls=':')  
plt.subplot(3, 1, 3); plt.plot(w[:-1], gd); plt.title('Retardo de Grupo'); plt.xlabel('Frecuencia [Hz]'); plt.ylabel('τg [# muestras]'); plt.grid(True, which='both', ls=':')  
plt.tight\_layout(); plt.show()  
  
# Aplicación de Ejemplo (Fase Cero - Duración Total)  
ECG\_f = sosfiltfilt(sos, ecg\_one\_lead)   
zoom\_region = np.arange(0, cant\_muestras, dtype='uint')  
plt.figure(figsize=(12, 6))  
plt.plot(zoom\_region / fs, ecg\_one\_lead[zoom\_region], label='1. Audio Original (CW + Ruido)', color='red', alpha=0.6)  
plt.plot(zoom\_region / fs, ECG\_f[zoom\_region], label='2. Audio Filtrado (Butterworth)', linewidth=2, color='blue')  
plt.title('Señal de Telegrafía Filtrada (Fase Cero - Duración Total)'); plt.xlabel('Tiempo [s]'); plt.ylabel('Amplitud'); plt.legend(); plt.grid(True); plt.tight\_layout(); plt.show()

### 1.2. Filtro\_IIR\_Chebyshev1\_CW.py (Chebyshev Tipo I)

Python

# -\*- coding: utf-8 -\*-  
"""  
Filtro\_IIR\_Chebyshev1\_CW.py - IIR (Chebyshev Tipo I)  
Diseño, verificación y aplicación de filtro Chebyshev I IIR Pasa-Banda.  
"""  
import numpy as np; from scipy import signal as sig; import matplotlib.pyplot as plt  
from matplotlib import patches; from scipy.signal import sosfiltfilt; import scipy.io.wavfile as wav  
  
# =========================================================================  
# 1. PLANTILLA DE DISEÑO Y CARGA DE DATOS (f\_tono=700 Hz, fs=44100 Hz)  
# =========================================================================  
fs = 44100; wp = (650, 750); ws = (550, 850); alpha\_p = 1; alpha\_s = 40  
  
try:  
 fs\_read, audio\_data = wav.read('CW.wav')   
 if audio\_data.ndim > 1: ecg\_one\_lead = audio\_data[:, 0]  
 else: ecg\_one\_lead = audio\_data  
 if fs\_read != fs: fs = fs\_read  
except Exception: ecg\_one\_lead = np.random.randn(int(fs \* 5)).astype(np.float32)   
cant\_muestras = len(ecg\_one\_lead)  
  
def zplane(z, p, title):  
 fig, ax = plt.subplots(figsize=(6, 6)); unit\_circle = patches.Circle((0, 0), radius=1, fill=False, color='black', alpha=0.3); ax.add\_artist(unit\_circle)  
 ax.plot(np.real(z), np.imag(z), 'o', markersize=9, label='Ceros'); ax.plot(np.real(p), np.imag(p), 'x', markersize=10, label='Polos')  
 ax.set\_title(title); ax.set\_xlabel('Real'); ax.set\_ylabel('Imaginario'); ax.set\_xlim([-1.5, 1.5]); ax.set\_ylim([-1.5, 1.5]); ax.grid(True, which='both', ls=':'); ax.legend(); plt.tight\_layout(); plt.show()  
  
# =========================================================================  
# 2. DISEÑO Y VERIFICACIÓN (Consigna c)  
# =========================================================================  
f\_aprox = 'cheby1'  
sos = sig.iirdesign(wp, ws, gpass=alpha\_p, gstop=alpha\_s, ftype=f\_aprox, output='sos', fs=fs)  
w, h = sig.sosfreqz(sos, worN=8000, fs=fs)  
w\_rad = w / (fs/2) \* np.pi; phase = np.unwrap(np.angle(h)); gd = -np.diff(phase) / np.diff(w\_rad)  
  
# Polos y Ceros  
z, p, k = sig.sos2zpk(sos)  
zplane(z, p, title=f'Polos y Ceros - {f\_aprox.capitalize()} I (N={sig.sos\_to\_numb\_order(sos)})')  
  
# Gráficas de Respuesta en Frecuencia  
plt.figure(figsize=(12, 10))  
plt.subplot(3, 1, 1); plt.plot(w, 20 \* np.log10(abs(h))); plt.title(f'Respuesta en Magnitud - {f\_aprox.capitalize()} I'); plt.ylabel('|H(jω)| [dB]'); plt.ylim([-50, 5]); plt.grid(True, which='both', ls=':')  
plt.subplot(3, 1, 2); plt.plot(w, phase); plt.title('Fase'); plt.ylabel('Fase [rad]'); plt.grid(True, which='both', ls=':')  
plt.subplot(3, 1, 3); plt.plot(w[:-1], gd); plt.title('Retardo de Grupo'); plt.xlabel('Frecuencia [Hz]'); plt.ylabel('τg [# muestras]'); plt.grid(True, which='both', ls=':')  
plt.tight\_layout(); plt.show()  
  
# Aplicación de Ejemplo (Fase Cero - Duración Total)  
ECG\_f = sosfiltfilt(sos, ecg\_one\_lead)   
zoom\_region = np.arange(0, cant\_muestras, dtype='uint')  
plt.figure(figsize=(12, 6))  
plt.plot(zoom\_region / fs, ecg\_one\_lead[zoom\_region], label='1. Audio Original (CW + Ruido)', color='red', alpha=0.6)  
plt.plot(zoom\_region / fs, ECG\_f[zoom\_region], label='2. Audio Filtrado (Chebyshev I)', linewidth=2, color='blue')  
plt.title('Señal de Telegrafía Filtrada (Fase Cero - Duración Total)'); plt.xlabel('Tiempo [s]'); plt.ylabel('Amplitud'); plt.legend(); plt.grid(True); plt.tight\_layout(); plt.show()

### 1.3. Filtro\_FIR\_Ventana\_CW.py (Método de Ventanas)

Python

# -\*- coding: utf-8 -\*-  
"""  
Filtro\_FIR\_Ventana\_CW.py - FIR (Método de Ventanas)  
Diseño, verificación y aplicación de filtro FIR Pasa-Banda usando Ventana (firwin2).  
"""  
import numpy as np; from scipy import signal as sig; import matplotlib.pyplot as plt  
from matplotlib import patches; from scipy.signal import lfilter; import scipy.io.wavfile as wav  
  
# =========================================================================  
# 1. PLANTILLA DE DISEÑO Y CARGA DE DATOS (f\_tono=700 Hz, fs=44100 Hz)  
# =========================================================================  
fs = 44100; wp = (650, 750); ws0 = 550; ws1 = 850  
N\_fir = 3000; numtaps = N\_fir + 1; demora = N\_fir // 2 # τg = 1500 muestras  
  
try:  
 fs\_read, audio\_data = wav.read('CW.wav')   
 if audio\_data.ndim > 1: ecg\_one\_lead = audio\_data[:, 0]  
 else: ecg\_one\_lead = audio\_data  
 if fs\_read != fs: fs = fs\_read  
except Exception: ecg\_one\_lead = np.random.randn(int(fs \* 5)).astype(np.float32)   
cant\_muestras = len(ecg\_one\_lead)  
  
def zplane(z, p, title):  
 fig, ax = plt.subplots(figsize=(6, 6)); unit\_circle = patches.Circle((0, 0), radius=1, fill=False, color='black', alpha=0.3); ax.add\_artist(unit\_circle)  
 ax.plot(np.real(z), np.imag(z), 'o', markersize=9, label='Ceros'); ax.plot(np.real(p), np.imag(p), 'x', markersize=10, label='Polos')  
 ax.set\_title(title); ax.set\_xlabel('Real'); ax.set\_ylabel('Imaginario'); ax.set\_xlim([-1.5, 1.5]); ax.set\_ylim([-1.5, 1.5]); ax.grid(True, which='both', ls=':'); ax.legend(); plt.tight\_layout(); plt.show()  
  
# =========================================================================  
# 2. DISEÑO Y VERIFICACIÓN (Consigna c)  
# =========================================================================  
f\_deseada = [0, ws0, wp[0], wp[1], ws1, fs / 2]  
m\_deseada = [0, 0, 1, 1, 0, 0]  
b = sig.firwin2(numtaps=numtaps, freq=f\_deseada, gain=m\_deseada, fs=fs, window='hamming')  
  
w, h = sig.freqz(b, worN=8000, fs=fs)  
phase = np.unwrap(np.angle(h))  
  
# Polos y Ceros  
z, p, k = sig.tf2zpk(b, 1)  
zplane(z, p, title=f'Polos y Ceros - FIR Ventana (N={N\_fir})')  
  
# Gráficas de Respuesta en Frecuencia  
plt.figure(figsize=(12, 10))  
plt.subplot(3, 1, 1); plt.plot(w, 20 \* np.log10(abs(h))); plt.title(f'Respuesta en Magnitud - FIR Ventana'); plt.ylabel('|H(jω)| [dB]'); plt.ylim([-50, 5]); plt.grid(True, which='both', ls=':')  
plt.subplot(3, 1, 2); plt.plot(w, phase); plt.title('Fase'); plt.ylabel('Fase [rad]'); plt.grid(True, which='both', ls=':')  
plt.subplot(3, 1, 3); plt.axhline(demora, color='orange', linestyle='--', label=f'τg teórico={demora} muestras'); plt.title('Retardo de Grupo'); plt.xlabel('Frecuencia [Hz]'); plt.ylabel('τg [# muestras]'); plt.grid(True, which='both', ls=':')  
plt.tight\_layout(); plt.show()  
  
# Aplicación de Ejemplo (Fase Lineal - Retardo Compensado)  
ECG\_f = lfilter(b, 1, ecg\_one\_lead)   
zoom\_region = np.arange(0, cant\_muestras - demora, dtype='uint')  
plt.figure(figsize=(12, 6))  
plt.plot(zoom\_region / fs, ecg\_one\_lead[zoom\_region], label='1. Audio Original (CW + Ruido)', color='red', alpha=0.6)  
plt.plot(zoom\_region / fs, ECG\_f[zoom\_region + demora], label=f'2. Audio Filtrado (Ventana - τg={demora} mues)', linewidth=2, color='blue')  
plt.title('Señal de Telegrafía Filtrada (Fase Lineal - Duración Total)'); plt.xlabel('Tiempo [s]'); plt.ylabel('Amplitud'); plt.legend(); plt.grid(True); plt.tight\_layout(); plt.show()

### 1.4. Filtro\_FIR\_Remez\_CW.py (Parks-McClellan / Remez)

Python

# -\*- coding: utf-8 -\*-  
"""  
Filtro\_FIR\_Remez\_CW.py - FIR (Parks-McClellan / Remez)  
Diseño, verificación y aplicación de filtro FIR Pasa-Banda usando Remez.  
"""  
import numpy as np; from scipy import signal as sig; import matplotlib.pyplot as plt  
from matplotlib import patches; from scipy.signal import lfilter; import scipy.io.wavfile as wav  
  
# =========================================================================  
# 1. PLANTILLA DE DISEÑO Y CARGA DE DATOS (f\_tono=700 Hz, fs=44100 Hz)  
# =========================================================================  
fs = 44100; wp = (650, 750); ws0 = 550; ws1 = 850  
N\_fir = 3000; numtaps = N\_fir + 1; demora = N\_fir // 2 # τg = 1500 muestras  
  
try:  
 fs\_read, audio\_data = wav.read('CW.wav')   
 if audio\_data.ndim > 1: ecg\_one\_lead = audio\_data[:, 0]  
 else: ecg\_one\_lead = audio\_data  
 if fs\_read != fs: fs = fs\_read  
except Exception: ecg\_one\_lead = np.random.randn(int(fs \* 5)).astype(np.float32)   
cant\_muestras = len(ecg\_one\_lead)  
  
def zplane(z, p, title):  
 fig, ax = plt.subplots(figsize=(6, 6)); unit\_circle = patches.Circle((0, 0), radius=1, fill=False, color='black', alpha=0.3); ax.add\_artist(unit\_circle)  
 ax.plot(np.real(z), np.imag(z), 'o', markersize=9, label='Ceros'); ax.plot(np.real(p), np.imag(p), 'x', markersize=10, label='Polos')  
 ax.set\_title(title); ax.set\_xlabel('Real'); ax.set\_ylabel('Imaginario'); ax.set\_xlim([-1.5, 1.5]); ax.set\_ylim([-1.5, 1.5]); ax.grid(True, which='both', ls=':'); ax.legend(); plt.tight\_layout(); plt.show()  
  
# =========================================================================  
# 2. DISEÑO Y VERIFICACIÓN (Consigna c)  
# =========================================================================  
bands = [0, ws0, wp[0], wp[1], ws1, fs/2] # Frecuencias de banda  
desired = [0, 1, 0]  
b = sig.remez(numtaps=numtaps, bands=bands, desired=desired, fs=fs)  
  
w, h = sig.freqz(b, worN=8000, fs=fs)  
phase = np.unwrap(np.angle(h))  
  
# Polos y Ceros  
z, p, k = sig.tf2zpk(b, 1)  
zplane(z, p, title=f'Polos y Ceros - FIR Remez (N={N\_fir})')  
  
# Gráficas de Respuesta en Frecuencia  
plt.figure(figsize=(12, 10))  
plt.subplot(3, 1, 1); plt.plot(w, 20 \* np.log10(abs(h))); plt.title(f'Respuesta en Magnitud - FIR Remez'); plt.ylabel('|H(jω)| [dB]'); plt.ylim([-50, 5]); plt.grid(True, which='both', ls=':')  
plt.subplot(3, 1, 2); plt.plot(w, phase); plt.title('Fase'); plt.ylabel('Fase [rad]'); plt.grid(True, which='both', ls=':')  
plt.subplot(3, 1, 3); plt.axhline(demora, color='orange', linestyle='--', label=f'τg teórico={demora} muestras'); plt.title('Retardo de Grupo'); plt.xlabel('Frecuencia [Hz]'); plt.ylabel('τg [# muestras]'); plt.grid(True, which='both', ls=':')  
plt.tight\_layout(); plt.show()  
  
# Aplicación de Ejemplo (Fase Lineal - Retardo Compensado)  
ECG\_f = lfilter(b, 1, ecg\_one\_lead)   
zoom\_region = np.arange(0, cant\_muestras - demora, dtype='uint')  
plt.figure(figsize=(12, 6))  
plt.plot(zoom\_region / fs, ecg\_one\_lead[zoom\_region], label='1. Audio Original (CW + Ruido)', color='red', alpha=0.6)  
plt.plot(zoom\_region / fs, ECG\_f[zoom\_region + demora], label=f'2. Audio Filtrado (Remez - τg={demora} mues)', linewidth=2, color='blue')  
plt.title('Señal de Telegrafía Filtrada (Fase Lineal - Duración Total)'); plt.xlabel('Tiempo [s]'); plt.ylabel('Amplitud'); plt.legend(); plt.grid(True); plt.tight\_layout(); plt.show()

## 2. 📝 Informe Final Ultra Detallado y Conceptual

### I. Fundamento del Diseño: Análisis Espectral y Plantilla

El diseño de los filtros se centra en el principio de selectividad máxima. El objetivo es aislar la energía del tono CW concentrada en $\mathbf{700 \text{ Hz}}$, atenuando drásticamente el ruido de banda ancha presente en el archivo CW.wav.

#### 1. Justificación de la Plantilla Pasa-Banda

Para lograr la mejor **Relación Señal-a-Ruido (SNR)**, se requiere un filtro:

* **Pasa-Banda Estrecho:** La energía de la telegrafía es una portadora sinusoidal (700 Hz) con bandas laterales cercanas que forman la **envolvente temporal** (puntos y rayas). La banda de paso debe ser lo suficientemente estrecha para rechazar el ruido adyacente, pero lo suficientemente ancha para no distorsionar las envolventes, lo que se consigue con $\pm 50 \text{ Hz}$.
* **Alta Atenuación:** Se utiliza $\mathbf{40 \text{ dB}}$ ($\mathbf{1\%}$ en magnitud) para asegurar que el ruido fuera del rango $650 \text{ Hz}-750 \text{ Hz}$ se suprima de manera efectiva.

#### 2. Determinación de Parámetros

| **Parámetro** | **Valor (Hz/dB)** | **Justificación Basada en Señal CW / Ruido** |
| --- | --- | --- |
| **Banda de Detención Baja ($\mathbf{w\_{s\\_low}}$)** | $\mathbf{550}$ Hz | Proporciona una transición de $\mathbf{100 \text{ Hz}}$ ($650-550$). |
| **Banda de Paso ($\mathbf{w\_p}$)** | $[\mathbf{650}, \mathbf{750}]$ Hz | Rango ideal para contener el tono de $700 \text{ Hz}$ y sus bandas laterales más cercanas. |
| **Banda de Detención Alta ($\mathbf{w\_{s\\_high}}$)** | $\mathbf{850}$ Hz | Simétrico a $w\_{s\\_low}$, dando una transición de $\mathbf{100 \text{ Hz}}$ ($850-750$). |
| **Atenuación Mínima ($\mathbf{\alpha\_s}$)** | $\mathbf{40}$ dB | Supresión robusta del ruido. |

### II. Diseño y Comparación de IIR vs. FIR (Consigna c)

Los filtros IIR y FIR representan dos compromisos opuestos en el procesamiento de señales: **Eficiencia** vs. **Fase Lineal**.

| **Característica** | **IIR (Infinita Respuesta Impulsional)** | **FIR (Finita Respuesta Impulsional)** |
| --- | --- | --- |
| **Orden (N)** | **Bajo** ($N \approx 8-10$) | **Alto** ($N \approx 3000$). Requerido por la alta $f\_s=44100 \text{ Hz}$ y la transición estrecha. |
| **Eficiencia** | **Máxima** (Mínimo costo computacional). | **Baja** (Alto costo computacional y memoria). |
| **Fase** | **NO Lineal** (Retardo de grupo variable). | **Lineal** (Retardo de grupo constante). |
| **Retardo de Grupo ($\tau\_g$)** | **Variable** (Distorsiona la envolvente CW). | **Constante** ($\tau\_g = 1500$ muestras). |
| **Solución de Inocuidad** | Uso de **sosfiltfilt** (Fase Cero). | Compensación de la **Demora Constante** ($\tau\_g = N/2$). |
| **Estabilidad** | Requiere que los Polos estén dentro del círculo unitario. | Siempre estable (Polos en el origen $z=0$). |

#### A. Filtros IIR (Butterworth vs. Chebyshev I)

Ambos son extremadamente eficientes para esta plantilla de banda estrecha.

| **Filtro** | **Aproximación / Diseño** | **Justificación y Funcionamiento** |
| --- | --- | --- |
| **Butterworth** | Máxima Planicidad | Su magnitud es la más **plana** en la banda de paso, asegurando que el tono de $700 \text{ Hz}$ pase con la máxima fidelidad sin rizos. |
| **Chebyshev Tipo I** | Rizo Equidistante | Logra la **transición más rápida** de todos los IIR, resultando en el **menor orden $N$** posible. Esto se logra a costa de introducir un **rizo** controlado ($\le 1 \text{ dB}$) en la banda de paso. |
| **Inocuidad** | **Fase Cero** | La función sosfiltfilt es esencial: aplica el filtro hacia adelante y hacia atrás, cancelando el efecto de la fase no lineal, garantizando que el pulso CW filtrado no se distorsione en el tiempo. |

#### B. Filtros FIR (Ventana vs. Remez/Parks-McClellan)

Ambos son la opción de **fase lineal**, pero su alto orden ($N=3000$) los hace costosos en tiempo de ejecución.

| **Filtro** | **Aproximación / Diseño** | **Justificación y Funcionamiento** |
| --- | --- | --- |
| **Ventana (Hamming)** | Truncamiento | Método simple. Si bien ofrece una transición relativamente buena, no es óptimo. Para lograr los $40 \text{ dB}$ en una transición de $100 \text{ Hz}$ con esta $f\_s$, requiere un orden alto, pero con una atenuación en banda de detención **no controlada** de forma precisa. |
| **Remez/Parks-McClellan** | Mínimo Error (Minimax) | Es el diseño FIR **óptimo**. Minimiza el error de manera uniforme (equiripple) en $w\_p$ y $w\_s$. Garantiza que $\mathbf{\alpha\_s = 40 \text{ dB}}$ se cumpla con la transición más estrecha para el orden $\mathbf{N=3000}$. |
| **Inocuidad** | **Fase Lineal** | La preservación de la forma del pulso CW es máxima. La única consecuencia es un **retardo constante** de $\mathbf{1500}$ muestras, que se compensa al graficar (ECG\_f[demora:]). |

### III. Evaluación del Rendimiento

La evaluación del rendimiento se realiza cualitativamente sobre los resultados de aplicación en los scripts.

| **Punto de la Consigna** | **IIR (Fase Cero)** | **FIR (Fase Lineal)** | **Resultado General** |
| --- | --- | --- | --- |
| **1. Filtrado de Interferentes (Ruido)** | Excelente. La atenuación de $40 \text{ dB}$ elimina de forma audible el ruido de banda ancha fuera de los $700 \text{ Hz}$. | Excelente. El rechazo de banda es incluso más estricto en el caso de Remez. | Se cumple. El tono CW es aislado del ruido de fondo con alta eficiencia en todos los casos. |
| **2. Inocuidad (Preservación de la Envolvente)** | Buena. El pulso CW es preservado **gracias a sosfiltfilt** (Fase Cero). Sin él, el Retardo de Grupo Variable distorsionaría la envolvente (el inicio y fin del pulso). | Excelente. La Fase Lineal garantiza que el pulso CW se retrase $\mathbf{1500}$ muestras, pero su forma temporal es idéntica a la original. | Se cumple. La forma de las 'rayas' y 'puntos' se mantiene inalterada. |

#### Conclusión de la Evaluación

Para el filtrado de telegrafía CW, la **Inocuidad** es fundamental para una decodificación temporal correcta.

1. **Si la Eficiencia Computacional es la prioridad (ej. DSP embebido):** El **IIR Chebyshev I** es la mejor opción, siempre y cuando se utilice **sosfiltfilt** (Fase Cero). Ofrece el orden más bajo y la transición más rápida.
2. **Si la Inocuidad (Fase Lineal Pura) es la prioridad y el retardo es aceptable:** El **FIR Remez** es la solución técnicamente superior. Garantiza que la envolvente temporal del pulso CW no sufrirá distorsión de fase.