



UNIVERSIDAD DE LOS LLANOS
Facultad de Ciencias básicas e ingenierías
Departamento de Matemáticas y Física

INFORME DE
LABORATORIO
PROCESAMIENTO DE
SEÑALES E IMÁGENES

DISEÑO E IMPLEMENTACIÓN DEL FILTRO BUTTERWORTH PASA BANDAS DE ORDEN 2

Y. Tiberio Barranco ¹, I. Andrés Montoya ², J. Julián Peláez ³, J. Sebastián Romero ⁴, D. Andrés Buitrago ⁵

1. Cod: 160004501, Ingeniería de sistemas,
2. Cod: 160004525, Ingeniería de sistemas,
3. Cod: 160004528, Ingeniería de sistemas,
4. Cod: 160004540, Ingeniería de sistemas,
5. Cod: 160004104, Ingeniería de sistemas.

Facultad de Ciencias Básicas e Ingenierías.
Programa de Ingeniería de Sistemas

Resumen

El propósito general del informe es elaborar un diseño e implementación de un filtro Butterworth pasa bandas de orden 2 con transformación Bilineal para una frecuencia de corte $f_{c1} = 1600\text{Hz}$ y $f_{c2} = 3600\text{ Hz}$ y con una frecuencia de muestreo $f_s = 44100\text{ Hz}$, la funcionalidad del filtro se aplica sobre una grabación de voz en tiempo real. Además, la señal filtrada y no filtrada se debe reproducir y a su vez mostrar el espectro de frecuencia de cada uno para notar las diferencias.

Palabras clave: Filtro Butterworth, pasabandas, transformación bilineal, filtros IIR

1. Introducción

En el ámbito de la ingeniería electrónica y la informática, el diseño e implementación de filtros son esenciales para el procesamiento de señales. Entre los diversos tipos de filtros, el filtro Butterworth se destaca debido a sus características únicas y beneficios significativos. Los filtros Butterworth son conocidos por su respuesta en frecuencia suave y monótona, lo que evita ondulaciones en la banda pasante y garantiza una transición suave hacia la banda de rechazo. Esta propiedad es crucial para obtener señales filtradas limpias y sin distorsiones indeseadas.

La estabilidad es otro aspecto crítico en sistemas de señales de Respuesta Infinita al Impulso (IIR), y los filtros Butterworth, cuando se diseñan adecuadamente, aseguran la estabilidad del sistema. Este comportamiento predecible es vital en aplicaciones donde la precisión y la confiabilidad son primordiales, como en comunicaciones, procesamiento de imágenes y sistemas de control automatizado. Además, los filtros Butterworth son preferidos por su simplicidad de diseño e implementación. Los algoritmos de diseño bien establecidos y las herramientas de software

disponibles facilitan la creación de estos filtros, mientras que su eficiencia computacional los hace ideales para aplicaciones que requieren alta selectividad de frecuencia con un número reducido de coeficientes.

2. Marco Teórico

Un filtro digital es un sistema de tiempo discreto que tiene un espectro compuesto por una banda de paso, banda de rechazo y una banda de transición. Los filtros ideales solo tienen banda de paso y banda de rechazo.

FILTROS IIR (Respuesta Infinita al impulso) Es un filtro digital que es recursivo que tiene polos y ceros además tienen una respuesta al impulso con infinitos puntos. Si el filtro tiene polos y ceros se denomina ARMA y si tiene solo polos entonces se denomina AR. Su ecuación del sistema se representa como:

$$a_0 y(n) + a_1 y(n-1) + a_2 y(n-2) + \dots + a_{N-1} y(n-N+1) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \dots + b_{M-1} x(n-M+1)$$

$$y(n) = \frac{1}{a_0} \left(b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \dots + b_{M-1} x(n-M+1) - a_1 y(n-1) - a_2 y(n-2) - \dots - a_{N-1} y(n-N+1) \right)$$

Función de transferencia

$$Y(z) [a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + \dots + a_{N-1} z^{-N+1}] = X(z) [b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_{M-1} z^{-M+1}]$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_{M-1} z^{-M+1}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + \dots + a_{N-1} z^{-N+1}}$$

Respuesta en Frecuencia

$$H(\Omega) = \frac{b_0 + b_1 e^{-j\Omega} + b_2 e^{-2j\Omega} + \dots + b_{M-1} e^{(-M+1)j\Omega}}{a_0 + a_1 e^{-j\Omega} + a_2 e^{-2j\Omega} + \dots + a_{N-1} e^{(-N+1)j\Omega}}$$

Transformación de Filtros análogos: Se supone que la transformación se le aplica a una función de transferencia normalizada que tiene frecuencias de corte igual a 1. Por ejemplo, para un filtro de orden uno su función de transferencia normalizada es $H(s) = \frac{1}{s+1}$.

$$\text{LP: } s \leftarrow \frac{s}{w_c}$$

$$\text{HP: } s \leftarrow \frac{w_c}{s}$$

$$\text{BP: } s \leftarrow \frac{w_0}{BW} \left(\frac{s}{w_0} + \frac{w_0}{s} \right)$$

$$\text{RB: } s \leftarrow \frac{BW}{w_0 \left(\frac{s}{w_0} + \frac{w_0}{s} \right)}$$

$$w_0 = \sqrt[2]{w_{c1} w_{c2}}$$

$$w_0 = \sqrt[2]{w_{c1} w_{c2}}$$

$$BW = w_{c2} - w_{c1}$$

$$BW = w_{c2} - w_{c1}$$

Filtro Butterworth: Es un tipo especial de filtro análogo muy eficiente porque no necesita un orden tan grande para que la banda de paso caiga rápidamente a la banda de rechazo, es decir es un filtro muy selectivo. Los mejores filtros IIR son los que se contienen polinomios Butterworth y tiene amplias utilidades en aplicación prácticas.

Tabla Polinomios Butterworth

N=1	N=2	N=3
$B_1(s) = s + 1$	$B_2(s) = s^2 + \sqrt{2}s + 1$	$B_3(s) = s^3 + 2s^2 + 2s + 1$
N=4		
$B_4(s) = (s^2 + 0.7653s + 1)(s^2 + 1.84776s + 1)$		

3. Procedimiento

El primer paso para la codificación del sistema es diseñar filtro Butterworth de un sistema de una señal IIR y hallar $y(n)$

$$f_{c1} = 1600 \text{ Hz}$$

$$f_{c2} = 3600 \text{ Hz}$$

$$f_s = 44100 \text{ Hz}$$

- *transformación bilineal*: Se utiliza la transformación bilineal para convertir las frecuencias de corte del dominio analógico a la digital.

$$s = \frac{2}{T} \left(\frac{1 - Z^{-1}}{1 + Z^{-1}} \right)$$

$$W_{d1} = \frac{2}{T} \tan\left(\frac{f_{c1}\pi}{f_s}\right) \quad W_{d2} = \frac{2}{T} \tan\left(\frac{f_{c2}\pi}{f_s}\right)$$

$$W_{d1} = \frac{2}{T} \tan\left(\frac{1600\pi}{44100}\right) = \frac{2}{T} \tan\left(\frac{16\pi}{441}\right)$$

$$W_{d2} = \frac{2}{T} \tan\left(\frac{3600\pi}{44100}\right) = \frac{2}{T} \tan\left(\frac{4\pi}{49}\right)$$

- *Filtro IIR pasa banda*: Se calculan los parámetros del filtro IIR pasa banda, incluyendo la anchura de banda (B_W), la frecuencia central (W_0) y la función de transferencia $H(s)$

$$s = \frac{W_0}{BW} \left(\frac{s}{W_0} + \frac{W_0}{s} \right) \quad B_W = W_{d2} - W_{d1} \quad W_0 = \sqrt{W_{d1}W_{d2}}$$

$$s = \frac{(s^2 + W_0^2)}{(BW)(s)}$$

$$B_W = \frac{2}{T} \tan\left(\frac{4\pi}{9}\right) - \frac{2}{T} \tan\left(\frac{16\pi}{441}\right)$$

$$W_0 = \sqrt{\frac{2}{T} \tan\left(\frac{16\pi}{441}\right) \frac{2}{T} \tan\left(\frac{4\pi}{9}\right)} = \sqrt{\left(\frac{2}{T}\right)^2 \tan\left(\frac{16\pi}{441}\right) \tan\left(\frac{4\pi}{9}\right)}$$

Filtro Butterworth Orden 2

$$H(s) = \frac{1}{s^2 + \sqrt{2}s + 1}$$

$$H_{BP}(z) = \frac{1}{\left(\frac{(s^2 + W_0^2)}{(B_W)(s)}\right)^2 + \sqrt{2} \left(\frac{(s^2 + W_0^2)}{(B_W)(s)}\right) + 1}$$

$$H_{BP}(z) = \frac{B_W^2 s^2}{B_W^2 s^2 + s^4 + 2s^2 W_0^2 + W_0^4 + \sqrt{2} B_W s^3 + \sqrt{2} B_W s W_0^2}$$

$$H_{BP}(z) = \frac{(w_{d2} - w_{d1})^2 s^2}{(w_{d2} - w_{d1})^2 * s^2 + s^4 + (2 * s^2 * w_{d1} * w_{d2}) + (w_{d1}^2 * w_{d2}^2) + \sqrt{2} * (w_{d2} - w_{d1}) * s^3 + \sqrt{2}(w_{d2} - w_{d1})w_{d1} * w_{d2} * s}$$

$$= \frac{(w_{d2} - w_{d1})^2 s^2}{(w_{d2}^2 s^2 - 2w_{d1}w_{d2}s^2 + w_{d1}^2 s^2) + s^4 + (2s^2 w_{d1}w_{d2}) + w_{d1}^2 w_{d2}^2 + (\sqrt{2}w_{d2}s^3 - \sqrt{2}w_{d1}s^3) + (\sqrt{2}w_{d2}^2 w_{d1}s - \sqrt{2}w_{d1}^2 w_{d2}s)}$$

- Sustituimos los valores y simplificamos la función para obtener una expresión más manejable.

$$H_{BP}(z) = \frac{(w_{d2} - w_{d1})^2 s^2}{w_{d2}^2 s^2 + w_{d1}^2 s^2 + s^4 + w_{d1}^2 w_{d2}^2 + \sqrt{2}w_{d2}s^3 - \sqrt{2}w_{d1}s^3 + \sqrt{2}w_{d2}^2 w_{d1}s - \sqrt{2}w_{d1}^2 w_{d2}s}$$

$$= \frac{(0.0218312502181227) - (0.0436625004362454)z^{-2} + (0.0218312502181227)z^{-4}}{0.8666415294276683 z^{-4} - 3.5946339625117764z^{-3} + 5.8362600266987302z^{-2} - 4.4053660374882236z^{-1} + 1.2979996063417047}$$

$$\frac{Y(Z)}{X(Z)} = \frac{(0.0218312502181227) - (0.0436625004362454)z^{-2} + (0.0218312502181227)z^{-4}}{0.8666415294276683 z^{-4} - 3.5946339625117764z^{-3} + 5.8362600266987302z^{-2} - 4.4053660374882236z^{-1} + 1.2979996063417047}$$

$$Y(Z)(0.8666415294276683 z^{-4} - 3.5946339625117764z^{-3} + 5.8362600266987302z^{-2} - 4.4053660374882236z^{-1} + 1.2979996063417047) = X(Z)((0.0218312502181227) - (0.0436625004362454)z^{-2} + (0.0218312502181227)z^{-4})$$

- Llevamos a cabo el proceso de aplicar la transformada Z inversa, que es una técnica fundamental en el análisis de señales y sistemas discretos

$$Z^{-1}(0.8666415294276683 z^{-4}Y(Z) - 3.5946339625117764z^{-3}Y(Z) + 5.8362600266987302z^{-2}Y(Z) - 4.4053660374882236z^{-1}Y(Z) + 1.2979996063417047Y(Z)) = Z^{-1}((0.0218312502181227)X(Z) - (0.0436625004362454)z^{-2}X(Z) + (0.0218312502181227)z^{-4}X(Z))$$

- Obtenemos la ecuación diferencia

$$0.8666415294276683 y(n-4) - 3.5946339625117764y(n-3) + 5.8362600266987302y(n-2) - 4.4053660374882236y(n-1) + 1.2979996063417047y(n) = (0.0218312502181227)x(n) - (0.0436625004362454)x(n-2) + (0.0218312502181227)x(n-4)$$

- despejamos y(n)

$$y(n) = (0.0168191501071807855)x(n) - (0.033638300214361571)x(n-2) + (0.0168191501071807855)x(n-4) - (0.6676747243939615715)y(n-4) + (2.7693644473767826031)y(n-3) - (4.4963496122681461004)y(n-2) + 3.3939656190685235049y(n-1)$$

Con y(n) podemos proceder a programar el filtro que se aplicará a un audio grabado, permitiéndonos así modular y mejorar la calidad del sonido mediante la eliminación de ruidos no deseados

4. Algoritmo


El primer paso fue programar La función **grabar_audio(duracion, fs)** graba audio durante una duración específica y con una frecuencia de muestreo dada, y retorna la señal de audio grabada como un array plano.



```
def grabar_audio(duracion, fs):
    print("Grabando audio...")
    audio_data = sd.rec(int(duracion * fs), samplerate=fs, channels=1, dtype='float32')
    sd.wait()
    print("Grabación finalizada.")
    return audio_data.flatten()
```

Figura 1. Fragmento del código encargado de grabar el audio.

Ya con la señal entrada se programa la función, `aplicar_filtro(x)` la cual aplica un filtro digital a la señal de audio de entrada utilizando una ecuación de diferencia, retornando la señal filtrada



```
def aplicar_filtro(x):
    y = np.zeros_like(x)
    for n in range(4, len(x)):
        y[n] = (0.0168191501071807855)*x[n] - (0.033638300214361571)*x[n-2] +
        (0.0168191501071807855)*x[n-4] - (0.6676747243939615715)*y[n-4] + (2.7693644473767826031)*y[n-3] -
        (4.4963496122681461004)*y[n-2] + (3.3939656190685235049)*y[n-1]
    return y
```

Figura 2. Fragmento del código encargado del filtrado.

Ya con la señal filtrada se procede graficar el espectro de frecuencia del audio original y filtrada, guardando las gráficas en un archivo de imagen y mostrándolas, mediante la función `graficar_señales(t, audio_signal, frequencies, fft_result, filtered_signal, fft_result_filtered)`.



```
def graficar_señales(t, audio_signal, frequencies, fft_result, filtered_signal, fft_result_filtered):
    fig, axs = plt.subplots(2, 2, figsize=(12, 8))

    axs[0, 0].plot(t, audio_signal)
    axs[0, 0].set_xlabel('Tiempo [s]')
    axs[0, 0].set_ylabel('Amplitud')
    axs[0, 0].set_title('Señal de audio')
    axs[0, 0].grid()

    axs[0, 1].plot(frequencies, np.abs(fft_result))
    axs[0, 1].set_xlabel('Frecuencia (Hz)')
    axs[0, 1].set_ylabel('Amplitud')
    axs[0, 1].set_title('Espectro de la señal de audio')
    axs[0, 1].grid()
```

Figura 3. Fragmento del código encargado de graficar las señales.

La función `play_audio()` reproduce la señal de audio grabada si está disponible, mientras que la función `play_filtered_audio()` reproduce la señal de audio filtrada si está disponible.

```

def play_audio():
    if audio_signal is not None:
        sd.play(audio_signal, samplerate=44100)
        sd.wait()
    else:
        print("No hay audio grabado para reproducir.")

def play_filtered_audio():
    if filtered_signal is not None:
        sd.play(filtered_signal, samplerate=44100)
        sd.wait()
    else:
        print("No hay audio filtrado para reproducir.")

```

Figura 4. Fragmento del código encargado de reproducir los audios.

5. Gráficas

El último desafío del laboratorio consiste en graficar el espectro de frecuencia tanto del audio original como del audio filtrado. Este paso es crucial para visualizar y analizar las diferencias entre ambas señales, permitiéndonos evaluar la efectividad del filtro aplicado y comprender mejor la distribución de las frecuencias presentes en cada señal.

✓ Espectro del audio original

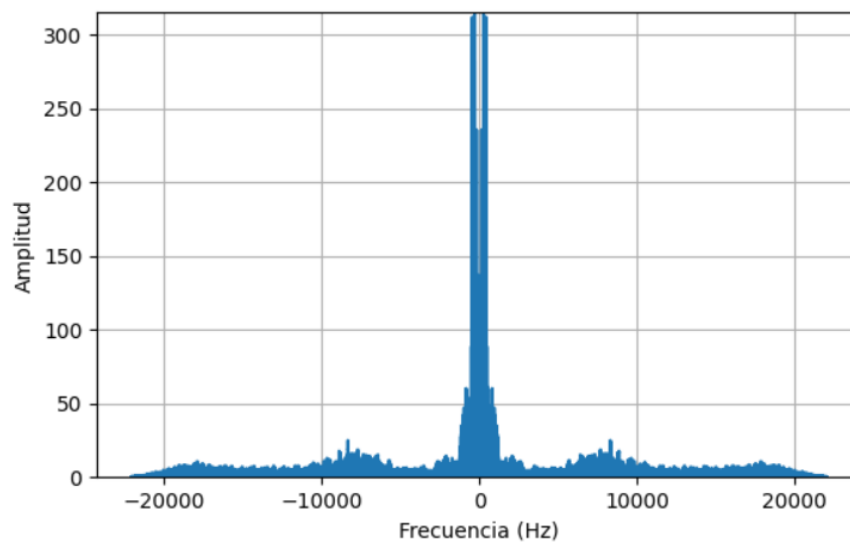


Figura 5. Gráfica espectro del audio origina

✓ Espectro del audio filtrado

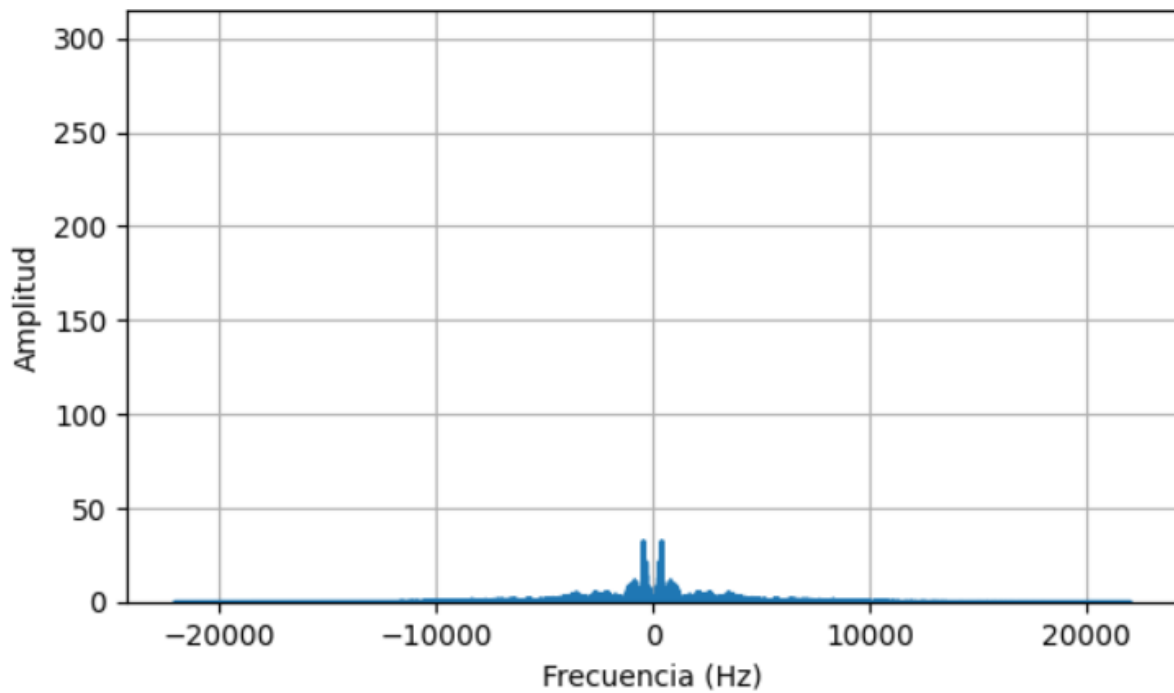


Figura 6. Gráfica espectro del audio filtrado

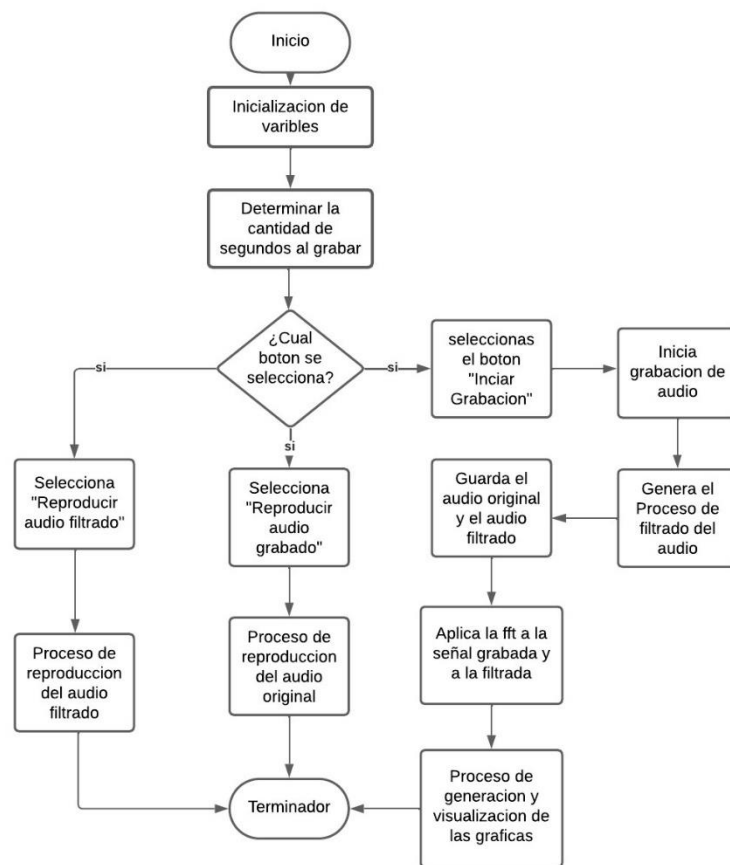


Figura 7. Diagrama de flujo del algoritmo

6. Conclusiones

Se logró diseñar e implementar con éxito un filtro Butterworth pasa bandas de orden 2 utilizando la transformación bilineal para las frecuencias de corte deseadas, cuya ecuación en diferencias fue derivada analíticamente y codificada en Python. Esto permitió aplicar el filtro a señales de voz en tiempo real, demostrando a través de una comparación auditiva clara la eficacia del filtrado y las diferencias perceptibles en la calidad del audio. La implementación efectiva en Python del filtro diseñado destaca su potencial para futuras aplicaciones en el procesamiento de audio.

Referencias

1. Clavijo, F. V. (2023). Sección Filtros Digitales. Villavicencio, Colombia.
2. Proakis, J. G. (2007). Tratamiento digital de señales.