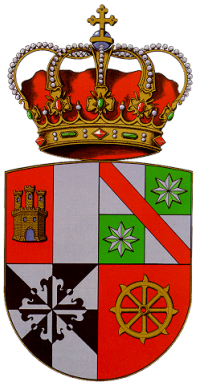

Procesado de Señal:

PRÁCTICA 6

# Diseño de filtros IIR

Ángel Fernández Arroyo

Rafael Martín San Martín

Angel.Fernandez17@alu.uclm.es

Rafael.Martin3@alu.uclm.es

# Índice



# Índice de figuras

# 1. Introducción

El propósito de esta práctica es aprender a diseñar filtros IIR. Para ello, generaremos una señal que contenga varias componentes armónicas y diseñaremos un filtro IIR que nos permita aislar una componente de interés. La práctica se divide en dos partes: el cálculo de los coeficientes del filtro mediante un script de MATLAB y la experimentación con el DSC TMS320F28335 de Texas Instruments.

# 2. Cálculo de los coeficientes del filtro IIR

Para la realización de esta práctica, se va a utilizar como base una señal cuyo contenido armónico se encuentra en la Tabla 2.

| Frecuencia (armónico) | Amplitud |
|---|---|
| 0 Hz (DC) | 0,25 |
| 50 Hz (h = 1) | 1 |
| 150 Hz (h = 3) | 0,5 |
| 250 Hz (h = 5) | 0,3 |
| 350 Hz (h = 7) | 0,25 |

Tabla 1: Componentes armónicas de la señal a filtrar.

La señal de interés para nuestra aplicación corresponde a la componente armónica de 50 Hz. En esta ocasión, se desea utilizar un filtro IIR paso banda para atenuar las componentes no deseadas. El primer paso a realizar consiste en el diseño de un filtro paso bajo analógico y después, se transformará dicho filtro en el dominio de tiempo discreto para, finalmente, realizar su transformación a un filtro paso banda.

En la Tabla 2 se recogen las especificaciones necesarias para el filtro paso bajo, que posteriormente se discretizará siguiendo la aproximación bilineal de Tustin. Y finalmente se traducirá en un filtro paso banda con las especificaciones descritas en la Tabla 3.

| Parámetro | Valor |
|---|---|
| $f_p$ (Hz) | 65 |
| $f_s$ (Hz) | 135 |
| $f_{muestreo}$ (KHz) | 1 |
| $\delta_p$ | 0,1 |
| $\delta_s$ | 0,01 |

Tabla 2: Parámetros de diseño del filtro paso bajo.

| Parámetro | Valor |
|---|---|
| $f_{p1}$ (Hz) | 20 |
| $f_{p2}$ (Hz) | 80 |
| $f_{muestreo}$ (KHz) | 1 |
| $\delta_p$ | 0,1 |
| $\delta_s$ | 0,01 |

Tabla 3: Parámetros de diseño del filtro paso banda.

El siguiente código recoge las intrucciones utilizadas en Matlab para el diseño de dicho filtro con los pasos descritos anteriormente.

```
% Butterworth filter continuous
[N, w_c] = buttord(w_p, w_s, delta_p_db, delta_s_db, 's');
[Hs_n, Hs_d] = butter(N, w_c, 's');
```

```
Hs = tf(Hs_n,Hs_d);

% Dicretize filter, tustin method
Fs = 1e3; Ts = 1/Fs;
Hz = c2d(Hs, Ts, 'tustin');

% Low pass to band pass filter
[Gz_n, Gz_d] = iirlp2bp(Hz.Num{1}, Hz.Den{1}, f_p*2/Fs, [20, 80]*2/Fs);
Gz = tf(Gz_n, Gz_d, Ts);

w_test = 2*pi*(1:0.01:500);

[resp_cont, resp_cont_ang] = bode(Hs, w_test);
resp_cont = squeeze(resp_cont); resp_cont_ang = squeeze(resp_cont_ang);
[resp_disc_lp, resp_disc_lp_ang] = bode(Hz, w_test);
resp_disc_lp = squeeze(resp_disc_lp); resp_disc_lp_ang = squeeze(resp_disc_lp_ang);
[resp_disc_bp, resp_disc_bp_ang] = bode(Gz, w_test);
resp_disc_bp = squeeze(resp_disc_bp); resp_disc_bp_ang = squeeze(resp_disc_bp_ang);
```

Tras el diseño, la respuesta en frecuencia del filtro se adjunta en la Figura 1. Se puede comprobar como el filtro paso banda discreto cumple con las especificaciones definidas anteriormente.

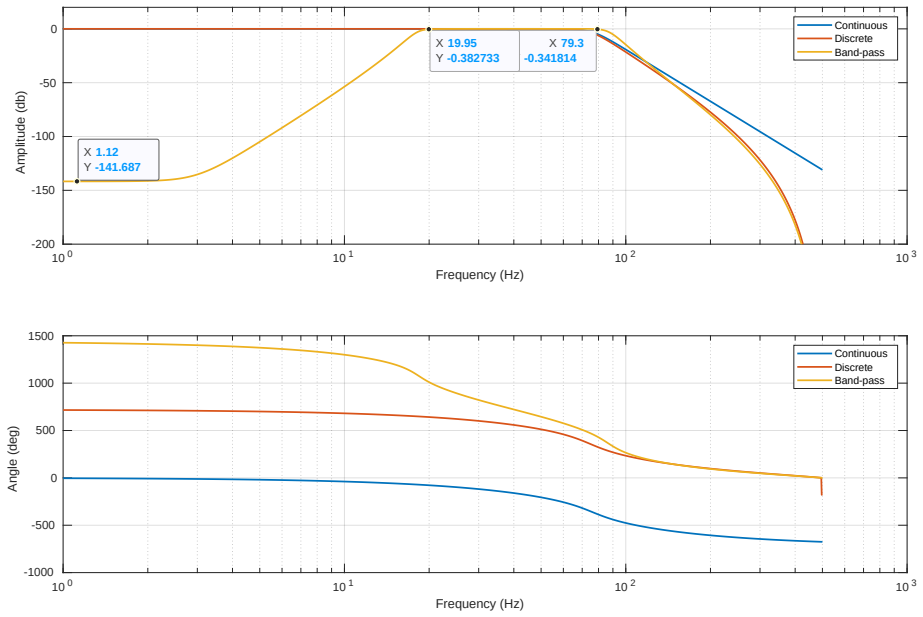


Figura 1: Respuesta en frecuencia del filtro diseñado: Analógico, Discreto, Discreto paso banda

# 3. Experimentación con el DSC

## 3.1. Diseña en Simulink la señal con el filtro IIR.

Tras la implementación en Simulink (véase Figura 2) se ha obtenido la señal contenida en la Figura 3. Dónde en la primera gráfica se observa la señal generada con las componentes armónicas descritas en la tabla 2, en la segunda de ellas se observa la señal fundamental que se desea obtener tras el filtrado y, finalmente, la señal filtrada.

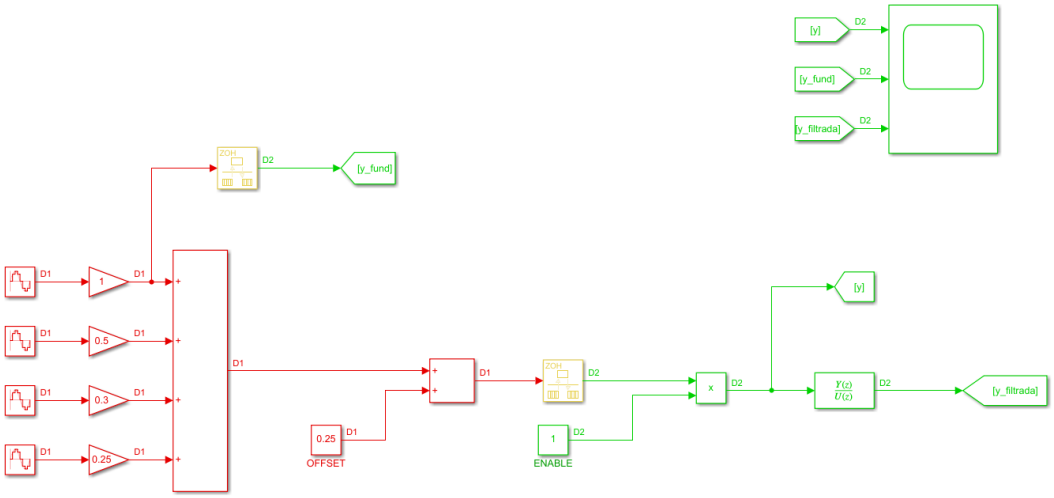


Figura 2: Implementación Simulink

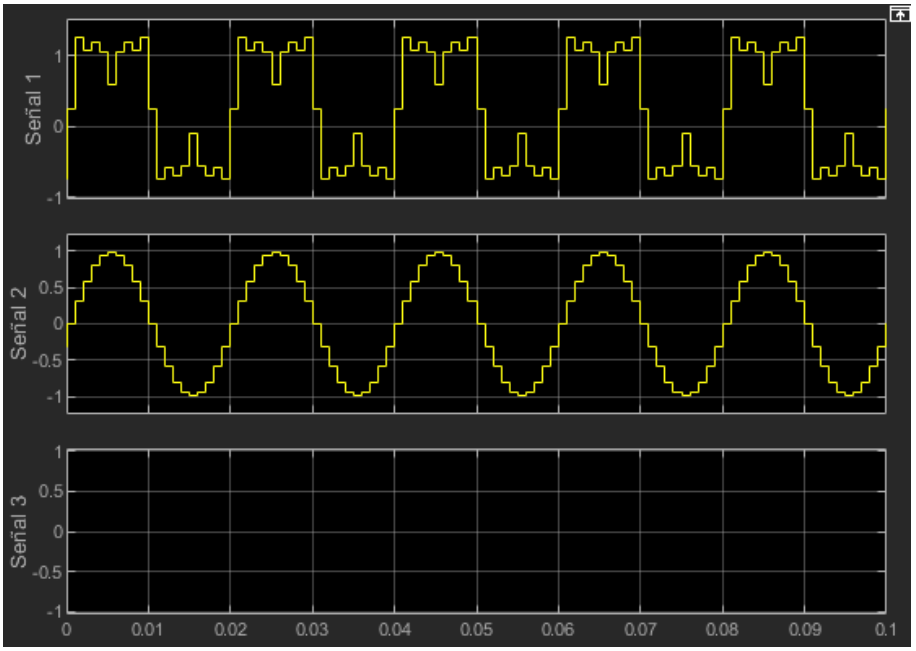


Figura 3: Señal obtenida del DSC

En este caso, no se observa ningún valor en la señal ya que, tras la discretización, el filtro se ha vuelto inestable y no converge a ningún valor. Esto es debido a que el dispositivo con el que se implementa no tiene suficiente precisión (32 bits) para representar los coeficientes y esto provoca pequeños errores en la localización de los polos, saliendose del circulo unidad y provocando que el filtro no converja.

## 3.2. Cuantización del filtro.

Para solucionar el problema descrito anteriormente, se puede hacer uso de filter designer (véase Figura 4) una herramienta que cuantiza el filtro, los parámetos del filtro serán los recogidos en la Tabla 4. Posteriormente se ajustará a precisión simple, que usará la especificación descrita en IEEE-754 para representar los coeficientes de forma que evitará polos fuera del circulo unidad y solucionando el problema anteriormente descrito.

| Parámetro | Valor | Parámetro | Valor |
|---|---|---|---|
| $f_{s1}$ (Hz) | 1 | $f_{p1}$ (Hz) | 20 |
| $f_{s2}$ (Hz) | 100 | $f_{p2}$ (Hz) | 80 |
| $\alpha_s$ (dB) | 80 | $\alpha_p$ (dB) | 0,01 |
| $f_{muestreo}$ (KHz) | 1 | | |

Tabla 4: Especificaciones de diseño para el filtro en FilterDesigner.

La aplicación devolverá un bloque que se introducirá en el diagrama de simulink, sustituyendo el bloque del filtro.

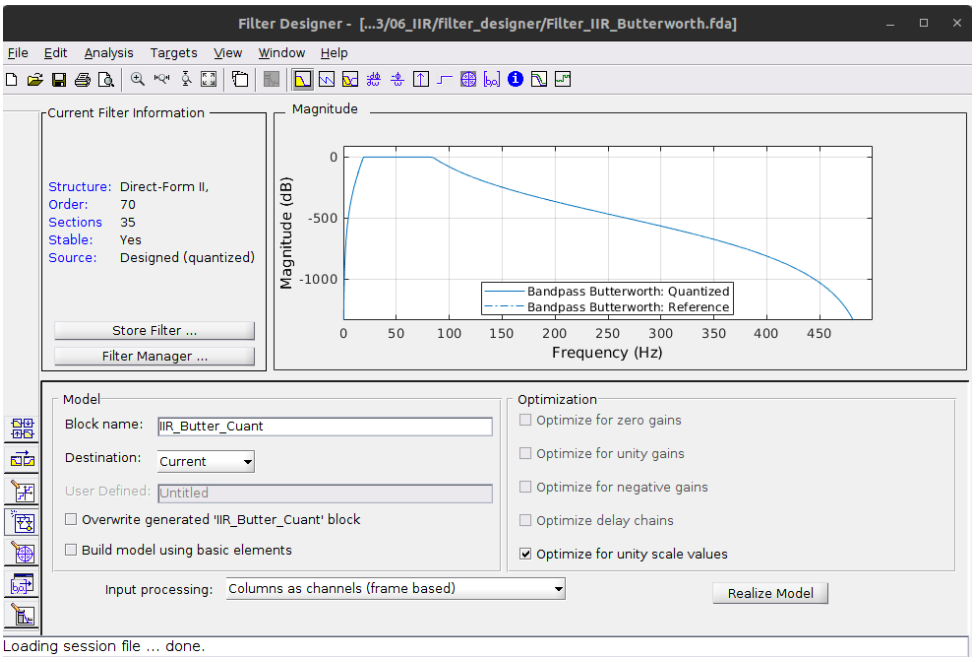


Figura 4: FilterDesigner

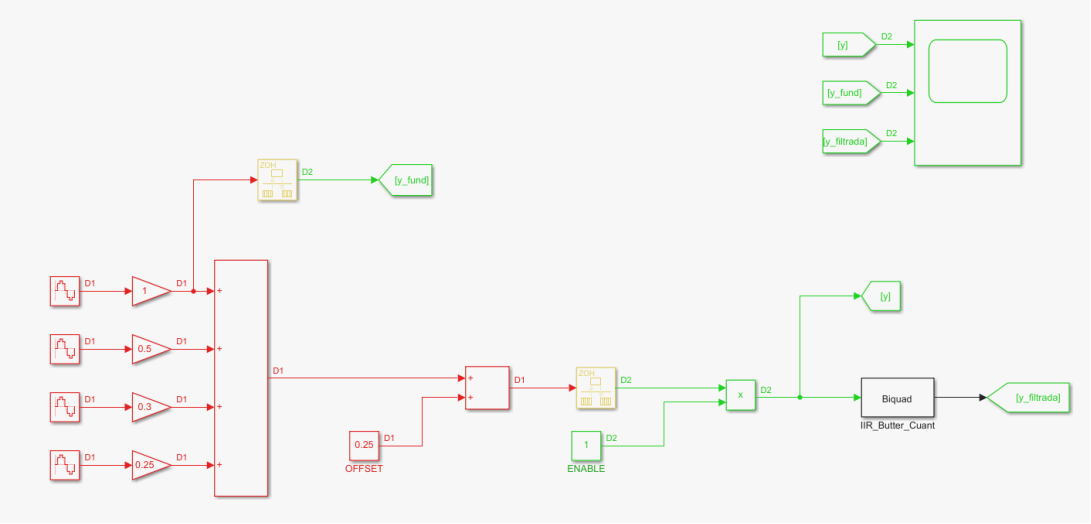


Figura 5: Implementación del bloque del FilterDesigner en Simulink

En la Figura 5 se observa el diagrama implementado en Simulink, y en la Figura 6 el resultado de la

señal. Se puede comprobar que la señal converge y además filtra tanto las componentes a bajas frecuencias y altas frecuencias, tal y como se ha diseñado en el primer apartado.

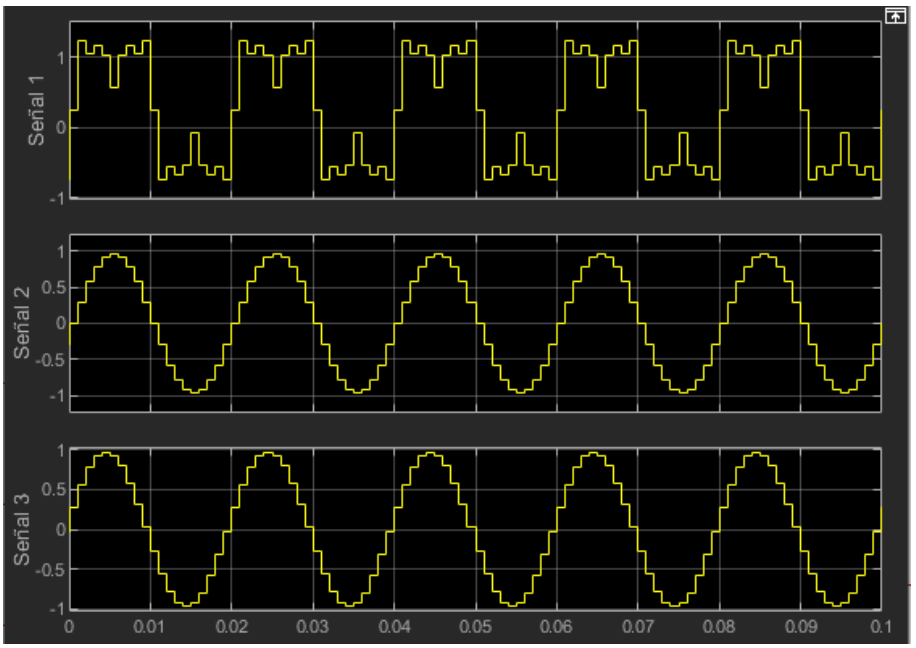


Figura 6: Señal obtenida del DSC

Con todo este proceso, se ha observado cómo implementar un filtro de respuesta impulsional infinita en un dispositivo real. También se ha analizado los errores de cuantización y cómo solucionarlos de forma sencilla.