



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

TÍTULO DE TESIS:

SÍNTESIS DE INTEGRADORES DE ORDEN FRACCIONARIO
USANDO HARWARE ANALÓGICO REPROGRAMABLE
Y SUS APLICACIONES

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LIC. EN ING. EN MECATRÓNICA

PRESENTA:

CIRO FABIÁN BERMÚDEZ MÁRQUEZ

TUTORES:

DR. JESÚS MANUEL MUÑOZ PACHECO



FCE

Facultad de Ciencias
de la Electrónica

PUEBLA, PUEBLA, 2020

Agradecimientos

Agradezco a mi familia ...

Agradezco al CONACYT ...

Agradezco a la Facultad de Ciencias de la Electrónica de la BUAP ...

Índice general

Agradecimientos	I
Índice general	IV
Índice de figuras	V
Índice de tablas	VII
Resumen	XI
1. Introducción	1
1.1. Justificación	2
1.2. Objetivos	4
1.2.1. Objetivo general	4
1.2.2. Objetivos específicos	4
2. Fundamentos teóricos	5
2.1. Definición de Grünwald-Letnikov	5
2.1.1. Definición de derivada de Grünwald-Letnikov	6
2.1.2. Definición de integral de Grünwald-Letnikov	6
2.1.3. Método numérico para la definición de GL	6
2.2. Definición de Riemann-Liouville	7
2.2.1. Definición de integral de Riemann-Liouville	7
2.2.2. Definición de derivada de Riemann-Liouville	8
2.3. Transformada de Laplace de integrales y derivadas fraccionarias	8
2.4. Expansión de fracciones continuas (CFE)	8
2.4.1. Análisis de error de la CFE	12
2.5. Escalamiento en frecuencia	16
2.6. Teoría de filtros	17
2.6.1. Filtros de primer orden	17
2.6.2. Filtros de segundo orden	18

3. Implementación	21
3.1. ¿Qué es una FPAA?	21
3.2. Características de la tarjeta y requerimientos	21
3.2.1. Alimentación de la tarjeta	21
3.2.2. Instalación de drivers	22
3.2.3. Jumpers por defecto	22
3.2.4. Tamaño variable de cadena de FPAAs	23
3.2.5. DIP Switches	24
3.2.6. Filtros Rauch y buffers de salida	24
3.2.7. Circuito de prueba	25
3.3. AnadigmDesigner2	25
3.3.1. Comunicación con AD2	26
3.3.2. Configuración de relojes	27
3.4. Implementación con aproximación de primer orden	29
3.5. Implementación con aproximación de segundo	34
4. Analisis de no se que ahorita	35
A. Códigos	37
B. Diagramas de flujo	41
C. Gráficas de análisis de integrador fraccionario con CFE	43
D. Esquemático de QuadApex v2.0	53
Bibliografía	55

Índice de figuras

2.1. Diagramas de bode comparativos de integrador fraccionario, funciones de transferencia de primer hasta quinto orden.	12
2.2. Diagrama de bode comparativo de función de transferencia escalada. . .	17
2.3. Definición de los parámetros ω_0 y Q de un par de polos conjugados. . .	19
3.1. Información de la placa desplegada por AD2.	27
3.2. Ventana de configuración de frecuencias de reloj AD2.	28
3.3. Diagrama de bode de ecuación (3.10) para un integrador fraccionario con $k_f = 2\pi 1000$ y $\alpha = 0.5$	30
3.4. Ventana de configuración de parámetros FilterBilinear.	31
3.5. Análisis de orden fraccionario dependiente de n	33
3.6. Implementación de integrador de orden fraccionario utilizando la aproximación de primer orden.	34
C.1. Diagramas de magnitud de aproximaciones de integrador fraccionario general.	44
C.2. Diagramas de fase de aproximaciones de integrador fraccionario general.	45
C.3. Diagramas de error de magnitud de aproximaciones de integrador fraccionario general.	46
C.4. Diagramas de error de fase de aproximaciones de integrador fraccionario general.	47
C.5. Diagramas de magnitud normalizada de aproximaciones de integrador fraccionario general.	48
C.6. Diagramas de fase normalizada de aproximaciones de integrador fraccionario general.	49
C.7. Diagramas de error de magnitud normalizada de aproximaciones de integrador fraccionario general.	50
C.8. Diagramas de error de fase normalizada de aproximaciones de integrador fraccionario general.	51
D.1. Diagrama esquemático de QuadApex v2.0	53

Índice de tablas

2.1.	Aproximaciones racionales de $\frac{1}{s^{0.5}}$	11
2.2.	Máximo error absoluto de magnitud en dB variando α y orden de función de transferencia.	13
2.3.	Máximo error absoluto de fase en grados variando α y orden de función de transferencia.	13
2.4.	Máximo error absoluto de magnitud normalizado en % variando α y orden de función de transferencia.	14
2.5.	Máximo error absoluto de fase normalizado en % variando α y orden de función de transferencia.	14
2.6.	Promedio de error absoluto de magnitud normalizado en % variando α y orden de función de transferencia.	15
2.7.	Promedio error absoluto de fase normalizado en % variando α y orden de función de transferencia.	15
3.1.	Resumen de jumper de la placa.	23
3.2.	DIP Switches	24
3.3.	CAMs básicos de AD2.	26
3.4.	Valores para configurar un filtro bilineal como integrador de orden fraccionario en el rango de ordenes de 0.1 a 0.95.	32
3.5.	Rango de frecuencias absolutas dependiente de F_c y el valor de n	32

Lista de códigos

A.1. Función <code>syms2tf</code>	37
A.2. Función <code>cfetf</code>	37
A.3. Función <code>A10_calculo_valores_FilterBilinear</code>	38
A.4. Función <code>A11_rango_de_frecuencias_absolutas_FilterBilinear</code>	38
A.5. Función <code>A12_grafica_analisis_de_frecuencias_FilterBilinear</code>	38

Resumen

Aquí va un resumen bien chido

Capítulo 1

Introducción

El caos se refiere a un tipo de comportamiento dinámico complejo que posee algunas características muy especiales, tales como extrema sensibilidad a pequeñas variaciones de la condición inicial, trayectorias encerradas en el espacio de fase pero con un exponente de Lyapunov positivo, un espectro de potencia continuo entre muchas otras. En pocas palabras, el caos es simplemente un comportamiento impredecible de un sistema determinista. Es de interés resaltar que los sistemas caóticos ya eran conocidos desde hace mucho tiempo atrás y que no fue hasta hace poco que se logró demostrar que el caos puede ser controlado y debido a esto impactar en muchas áreas, tanto en áreas cercanas a la electrónica como técnicas de modulación, sistemas de comunicación, técnicas de encriptación de datos, como también en áreas relacionadas a los sistemas biológicos, reacciones químicas, toma de decisiones críticas en política, economía, eventos militares, etc [1]. El caos es un fenómeno que ocurre en muchos sistemas no lineales, donde la naturaleza determinista de la estructura se conjuga con la irregularidad del comportamiento, esto significa que, a pesar del hecho de que el sistema se describe mediante un conjunto de ecuaciones diferenciales ordinarias, donde todos los términos son perfectamente conocidos, su comportamiento es irregular y muy sensible a las condiciones iniciales. La primera evidencia de imprevisibilidad en los sistemas deterministas se encuentra en el trabajo del matemático y científico Henri Poincaré sobre el movimiento celestial, mientras que la primera formulación del caos en un modelo matemático expresado por un conjunto de ecuaciones diferenciales ordinarias que exhiben el caos se debe al matemático y meteorólogo Edward Lorenz que estaba estudiando un modelo de movimiento del aire en la atmósfera y descubrió cómo pequeñas variaciones en los valores iniciales de las variables de su modelo dieron como resultado predicciones meteorológicas divergentes [2]. Para el momento de esos estudios faltaba una prueba experimental definitiva del caos y la tecnología y poder de cómputo no eran suficientes para pensar aún en soluciones y aún más distantes las aplicaciones. Debido al constante avance de la electrónica, hoy en día somos capaces de sintetizar mediante dispositivos electrónicos sistemas caóticos, utilizando técnicas de modelado

e implementación es posible crear representaciones de estos, no obstante, todas estas se basan en aproximaciones que aún no han sido exploradas en su totalidad. Por otro lado el cálculo fraccionario es un tema que tiene más de 300 años de antigüedad y que se remonta a cartas enviadas a Leibniz por parte de Bernoulli y de L'Hôpital preguntando acerca de la derivada a la $1/2$ e indagando sobre su significado. Con el paso de los años la teoría de cálculo fraccionario se fue desarrollando pasando por las manos de nombres conocidos como: Euler, Lagrange, Laplace, Fourier hasta llegar a Liouville, Riemann, Grünwald, Letnikov, Caputo entre muchos otros [3]. Pero aún con todo ese desarrollo no fue hasta hace poco que la comunidad científica comenzó a interesarse por esta rama del cálculo y la razón principal de este cambio es que los cálculos necesarios para cualquier posible implementación eran demasiado complejos y lentos, un panorama totalmente diferente se vive en la actualidad, el rápido avance de la tecnología ha logrado realizar avances notables en esta área. El rango de aplicaciones para el cálculo fraccionario es inmenso, por mencionar algunas de las más recientes, la modelación de derivadas fraccionarias para obtener una mejor representación comportamental de un sistema industrial metalúrgico [4], la incorporación de dinámica de orden fraccional para mejorar la robustez de un control PI/PID para motores DC [5, 6], la modelación de señales biológicas como ECG, EMG y EEG debido a su respuesta de magnitud de 20dB, modelos fisiológicos basados en ecuaciones diferenciales lineales que describen fenómenos complejos en el cuerpo humano como la oxigenación de la sangre entre otros [7]. El cálculo fraccionario y los sistemas caóticos se complementan al añadir un nivel de profundidad en la creación de osciladores caóticos modelados como un conjunto de ecuaciones diferenciales no lineales fraccionarias las cuales son materia prima para la creación de nuevas aplicaciones y áreas de desarrollo.

1.1. Justificación

Los osciladores caóticos son una área de oportunidad emergente cuyas aplicaciones han aumentado en los últimos años, a principios del siglo XX se comenzó a explorar el uso de osciladores caóticos para la detección de señales pequeñas basados en la propiedad de extrema sensibilidad a pequeños cambios de la condición inicial que estos presentan [8], y en la seguridad de sistemas de comunicación ya sea en tiempo continuo o discreto [9]. Casi al mismo tiempo los sistemas de orden fraccionario comenzaron a formar parte del interés de la comunidad científica nuevamente y se abrieron las puertas al control de sistemas de orden fraccionario y como implementarlos [10, 11]. Los intentos de implementaciones físicas de sistemas de orden fraccionario han avanzado mucho recientemente y se pueden categorizar en digitales, los cuales utilizan sistemas discretos basados en microprocesadores o FPGA cuya complejidad radica en poder generar un software robusto que sea lo suficiente fiable [12]. Por el otro lado se encuentran las

implementaciones analógicas, las cuales son de gran interés en este trabajo. En la literatura de esta área podemos encontrar implementaciones basadas en la construcción de capacitores electrolíticos especiales que se aproximan al comportamiento fraccionario pero cuya fabricación es compleja [13], métodos electroquímicos que trabajan con compuestos químicos de difícil manipulación y tienen ordenes que no pueden modificarse fácilmente [14], finalmente y en lo que se enfocará este trabajo se encuentran las aproximaciones de funciones racionales en un ancho de banda determinado utilizando los métodos de Newton, Oustaloup, Carlson, Matsuda o expansión de fracciones continuas (CFE) [15, 16, 17]. Una vez que la función de transferencia es obtenida, esta se sintetiza utilizando diferentes metodologías, como la realización pasiva que hace uso de resistencias, capacitores e inductores, con el inconveniente de poder ajustarse a los valores comerciales, o la realización activa, que utiliza las técnicas de diseño basadas en amplificadores operacionales [18, 19]. Una alternativa a estas metodologías es la sintetización utilizando dispositivos analógicos embebidos programables FPAA (Field Programmable Analog Array). Las FPAA han demostrado ser una arquitectura prometedora que facilita mucho el proceso de implementación de circuitos analógicos debido a que su interfaz se basa en Módulos Analógicos Configurables (CAM), los cuales son bloques que pueden interconectarse fácilmente y representan desde circuitos sencillos como integradores, derivadores, sumadores o inversores hasta circuitos más complejos como filtros completos y multiplicadores [20]. Sin embargo en el ámbito tanto de integradores fraccionarios como de osciladores caóticos aun existen metodologías nuevas por explorar, en algunos artículos científicos ya se ha mostrado como implementar un integrador de orden fraccionario utilizando la aproximación de Oustaloup y CAMs de filtros bilineales [21], no obstante la cantidad de recursos para crear un solo integrador es muy grande, lo cual da pie a ser mejorado utilizando aproximaciones diferentes, por ejemplo la expansión de fracciones continuas. De mismo modo implementar osciladores caóticos basados en sistemas de ecuaciones diferenciales de orden entero utilizando una FPAA ya ha sido explorado en años recientes [22], sin embargo los basados en sistemas de ecuaciones diferenciales de orden fraccionario aun se encuentran en una etapa temprana y son ideales para ser estudiados. Las aplicaciones de los osciladores caóticos van en aumento y ser capaces de implementarlos se ha vuelto una tarea relevante, estos ahora se están utilizando en aplicaciones médicas, como en la detección de señales cardíacas [23] y en la creación de radares UWB que tienen futuro prometedor en la medicina [24].

1.2. Objetivos

1.2.1. Objetivo general

Diseño e implementación electrónica de integradores de orden fraccionario mediante una expansión de fracciones continuas para su aplicación en sistemas caóticos.

1.2.2. Objetivos específicos

- Analizar el método de expansión de fracciones continuas para generar una metodología de diseño en MATLAB.
- Caracterizar el error de la expansión de fracciones continuas para generar reglas de diseño.
- Diseñar e implementar en FPAA el integrador de orden fraccionario con aproximaciones de ordenes superiores.
- Diseñar e implementar de FPAA un oscilador caótico de orden fraccionario.

Capítulo 2

Fundamentos teóricos

Al igual que cuando se comienza a estudiar cálculo de orden entero, es necesario familiarizarse con la notación de los operadores matemáticos de la derivada y la integral. En la actualidad la notación más utilizada para el cálculo entero es la dada por Leibniz en (1686), donde el operador diferencial de n -ésimo orden está definido como: $\frac{d^n}{dt^n}$, D_t^n o simplemente D^n con $n \in \mathbb{N}$. Utilizando el mismo razonamiento, puede definirse su operador inverso (antiderivada) de manera que el operador inverso de la derivada de n -ésimo orden está dado por: ${}_a D_t^{-n}$, donde $n \in \mathbb{N}$ y $a \in \mathbb{R}$ representa el límite inferior del dominio de la región donde se aplica dicho operador.

Para generalizar el operador diferencial e integral para orden fraccionario se considera que este puede definirse para parámetros de orden real o incluso complejo. Esto implica que los operadores pueden definirse respectivamente como: D^α y ${}_a D_t^\alpha$ con $\alpha \in \mathbb{R}$.

Es importante resaltar que no una hay una única definición de operadores diferencial fraccional ni integral sino varias expresiones definidas por diferentes autores, entre las mas usadas se encuentran la definición de Grünwald-Letnikov (GL), la de Riemann-Liouville (RL) y la de Caputo (Ca), cada una de estas con sus ventajas y desventajas desde el punto de vista del análisis matemático, complejidad computacional e implementación [3].

2.1. Definición de Grünwald-Letnikov

Comenzamos considerando que para el caso de orden entero la n -ésima derivada para una función f con $n \in \mathbb{N}$ y $j > n$ está dada por:

$$f^{(n)}(t) = \frac{d^n f}{dt^n} = \lim_{h \rightarrow 0} \frac{1}{h^n} \sum_{j=0}^n (-1)^j \binom{n}{j} f(t - jh) \quad (2.1)$$

donde $\binom{n}{j}$ representa el coeficiente binomial dado por la expresión:

$$\binom{n}{j} = \frac{n!}{j!(n-j)!} \quad (2.2)$$

Considerando valores negativos de n tenemos:

$$\binom{-n}{j} = \frac{-n(-n-1)(-n-2)\cdots(-n-j+1)}{j!} = (-1)^j \begin{bmatrix} n \\ j \end{bmatrix} \quad (2.3)$$

donde $\begin{bmatrix} n \\ j \end{bmatrix}$ esta definido como:

$$\begin{bmatrix} n \\ j \end{bmatrix} = \frac{2(n+1)\cdots(n+j-1)}{j!} \quad (2.4)$$

2.1.1. Definición de derivada de Grünwald-Letnikov

Generalizando la ecuación (2.1) podemos escribir la definición de derivada de orden fraccionario de orden α , ($\alpha \in \mathbb{R}$) como:

$$D_t^\alpha f(t) = \lim_{h \rightarrow 0} \frac{1}{h^\alpha} \sum_{j=0}^{\infty} (-1)^j \binom{\alpha}{j} f(t-jh) \quad (2.5)$$

Para calcular el coeficiente binomial podemos utilizar la relación entre la función Gamma de Euler y el factorial definido como:

$$\binom{\alpha}{j} = \frac{\alpha!}{j!(\alpha-j)!} = \frac{\Gamma(\alpha+1)}{\Gamma(j+1)\Gamma(\alpha-j+1)} \quad (2.6)$$

donde la función Gamma de Euler con $r > 0$ esta definida como:

$$\Gamma(r) = \int_0^{\infty} t^{r-1} e^{-t} dt \quad (2.7)$$

2.1.2. Definición de integral de Grünwald-Letnikov

Utilizando la ecuación (2.5) se puede definir un operador de tipo integral para la función f sobre el dominio temporal (a, t) considerando $n = \frac{t-a}{h}$ donde $a \in \mathbb{R}$ como:

$${}_a D_f^\alpha = \lim_{h \rightarrow 0} \frac{1}{h^\alpha} \sum_{j=0}^{\left[\frac{t-a}{h}\right]} (-1)^j \binom{n}{j} f(t-jh) \quad (2.8)$$

2.1.3. Método numérico para la definición de GL

Utilizando como base la ecuación (2.5) esta se puede discretizar para los puntos kh , ($k = 1, 2, \dots$) de la siguiente manera:

$$({\frac{L_m}{h}})D_{t_k}^\alpha f(t) \approx \frac{1}{h^\alpha} \sum_{j=0}^k (-1)^j \binom{\alpha}{j} f(t_{k-j}) \quad (2.9)$$

donde L_m es el tamaño de memoria (memory length), $t_k = kh$, h es el paso de tiempo del cálculo y $(-1)^j \binom{\alpha}{j}$ son coeficientes binomiales $C_j^{(\alpha)}$ ($j = 0, 1, \dots$). Para su calculo utilizamos la siguiente expresión:

$$C_0^{(\alpha)} = 1, \quad C_j^{(\alpha)} = \left(1 - \frac{1+\alpha}{j}\right) C_{j-1}^{(\alpha)} \quad (2.10)$$

Entonces, la solución numérica general de la ecuación diferencial fraccional:

$${}_a D_t^\alpha y(t) = f(y(t), t) \quad (2.11)$$

puede expresarse como:

$$y(t_k) = f(y(t_{k-1}), t_{k-1})h^\alpha - \sum_{j=1}^k C_j^{(\alpha)} y(t_{k-j}) \quad (2.12)$$

Para el termino de la memoria expresada por la sumatoria, el principio de memoria corta puede utilizarse. Entonces el indice superior de la sumatoria en la ecuación (2.12) se cambiará por ν con las siguientes consideraciones: se usa $\nu = k$ para $k < (\frac{L_m}{h})$ y $\nu = (\frac{L_m}{h})$ para $k \geq (\frac{L_m}{h})$, o sin usar el principio de memoria corta se utiliza $\nu = k$ para toda k .

2.2. Definición de Riemann-Liouville

Para esta definición consideramos la fórmula de Cauchy para la integral repetida que esta dada por:

$$f^{(-n)}(t) = \int_a^t \int_a^{\sigma_1} \dots \int_a^{\sigma_{n-1}} f(\sigma_n) d\sigma_n \dots d\sigma_2 d\sigma_1 = \frac{1}{(n-1)!} \int_a^t \frac{f(\tau)}{(t-\tau)^{1-n}} d\tau \quad (2.13)$$

2.2.1. Definición de integral de Riemann-Liouville

Utilizando las propiedades de la función Gamma de Euler con el factorial y la ecuación (2.13) se puede escribir la definición de integral fraccionaria como:

$${}_a D_t^{-\alpha} f(t) = \frac{1}{\Gamma(\alpha)} \int_a^t \frac{f(\tau)}{(t-\tau)^{1-\alpha}} d\tau \quad (2.14)$$

para $\alpha < 0$ y $a \in \mathbb{R}$. No obstante para el caso de $0 < \alpha < 1$ y $f(t)$ siendo una función casual, esto es, $f(t) = 0$ para $t < 0$, la integral fraccionaria esta definida como:

$${}_0D_t^{-\alpha}f(t) = \frac{1}{\Gamma(\alpha)} \int_0^t \frac{f(\tau)}{(t-\tau)^{1-\alpha}} d\tau, \quad \text{para } 0 < \alpha < 1, \quad t > 0 \quad (2.15)$$

2.2.2. Definición de derivada de Riemann-Liouville

De la ecuación (2.14) se puede escribir la definición de derivada fraccionaria de orden α de la siguiente manera:

$${}_aD_t^\alpha f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau \quad (2.16)$$

donde $(n-1 < \alpha < n)$. Pero igual que con la integral si consideramos $0 < \alpha < 1$ y $f(t)$ una función casual, la derivada de orden fraccionaria se puede reescribir como:

$${}_0D_t^\alpha f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_0^t \frac{f(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau \quad (2.17)$$

2.3. Transformada de Laplace de integrales y derivadas fraccionarias

La transformada de Laplace de la integral fraccionaria ya sea para Riemann-Liouville o para Grünwald-Letnikov esta definida como:

$$\mathcal{L}\{{}_0D_t^{-p}f(t)\} = s^{-p}F(s) \quad (2.18)$$

y dadas condiciones iniciales cero la transformada de Laplace de la derivada fraccionaria de orden r para Grünwald-Letnikov, Riemann-Liouville y Caputo se reduce a:

$$\mathcal{L}\{{}_0D_t^r f(t)\} = s^r F(s) \quad (2.19)$$

2.4. Expansión de fracciones continuas (CFE)

A una expresión de la forma:

$$a_1 + \frac{b_1}{a_2 + \frac{b_2}{a_3 + \frac{b_3}{a_4 + \ddots}}} \quad (2.20)$$

se le conoce como una fracción continua. En general $a_1, a_2, a_3, \dots, b_1, b_2, b_3$ pueden ser cualquier número real o complejo, y el número de términos pueden ser finito o infinito.

Una manera más conveniente de escribir la ecuación (2.20) es:

$$a_1 + \frac{b_1}{a_2 + \frac{b_2}{a_3 + \frac{b_3}{a_4 + \dots}}} \quad (2.21)$$

y es la que se encontrará normalmente en libros y artículos. Ambas notaciones son muy similar y se puede pasar de una a otra sin mayor complicación.

De la ecuación (2.21) se pueden formar las siguientes fracciones:

$$c_1 = \frac{a_1}{1}, \quad c_2 = a_1 + \frac{b_1}{a_2}, \quad c_3 = a_1 + \frac{b_1}{a_2 + \frac{b_2}{a_3}}, \quad \dots \quad (2.22)$$

las cuales se obtienen, en sucesión, de cortar el proceso de expansión después del primer, segundo, tercer, \dots término. Estas fracciones son llamadas primer, segundo, tercer, \dots convergente, respectivamente, de la fracción continua. El n -ésimo convergente es:

$$c_n = a_1 + \frac{b_1}{a_2 + \frac{b_2}{a_3 + \dots + \frac{b_{n-1}}{a_n}}} \quad (2.23)$$

En 1776 Lagrange obtuvo la expansión de fracciones continuas (CFE) para la ecuación $(1+x)^\alpha$ como se muestra a continuación [25]:

$$(1+x)^\alpha = \frac{1}{1 - \frac{\alpha x}{1 + \frac{1(1+\alpha)}{1 \cdot 2} x}} \quad (2.24)$$

$$1 + \frac{\frac{1(1-\alpha)}{2 \cdot 3} x}{1 + \frac{2(2+\alpha)}{3 \cdot 4} x}$$

$$1 + \frac{\frac{2(2-k)}{4 \cdot 5} x}{1 + \frac{3(3+\alpha)}{5 \cdot 6} x}$$

$$1 + \dots$$

y escrita de una manera más compacta:

$$(1+x)^\alpha = \frac{1}{1 - \frac{\alpha x}{1 + \frac{1(1+\alpha)}{1 \cdot 2} x} + \frac{1(1-\alpha)}{2 \cdot 3} x} + \frac{2(2+\alpha)}{3 \cdot 4} x + \frac{2(2-\alpha)}{4 \cdot 5} x + \dots \quad (2.25)$$

la ecuación (2.25) puede reescribirse convenientemente multiplicando un m en el numerador y en el denominador como se muestra a continuación:

$$(1+x)^\alpha = \frac{1}{1-} \frac{\alpha x}{1} + \frac{\textcolor{red}{2} \cdot \frac{1(1+\alpha)}{1 \cdot 2} x}{\textcolor{red}{2} \cdot 1} + \frac{\textcolor{blue}{3} \cdot \textcolor{red}{2} \cdot \frac{1(1-\alpha)}{2 \cdot 3} x}{\textcolor{blue}{3} \cdot 1} + \frac{\textcolor{blue}{3} \cdot \frac{2(2+\alpha)}{3 \cdot 4} x}{1} + \dots \quad (2.26)$$

hay que notar que cada denominador esta compuesto por 2 términos, esto se puede ver claramente en la ecuación (2.24), y que contando el término del numerador, m se tiene que agregar en 3 lugares distintos. Si se eligen $m_1 = 2$, $m_2 = 3$, $m_3 = 2$, \dots de manera que se simplifique la ecuación obtenemos:

$$(1+x)^\alpha = \frac{1}{1-} \frac{\alpha x}{1} + \frac{(1+\alpha)x}{2} + \frac{(1-\alpha)x}{3} + \frac{(2+\alpha)x}{2} + \frac{(2-\alpha)x}{5} + \dots \quad (2.27)$$

La ecuación (2.27) se puede encontrar en distintos artículos [17, 16], no obstante para programar un algoritmo que obtenga la aproximación de $(1+x)^\alpha$ hasta el n -ésimo convergente resulta poco intuitiva. Para este fin la ecuación (2.25) resulta más sencilla y contiene un patrón que puede explotarse.

El n -ésimo término de la expansión de fracciones continuas para la ecuación (2.25) se puede calcular utilizando la siguiente ecuación:

$$\frac{\psi(n) [\psi(n) + (-1)^n \alpha]}{(n-1)n} \quad (2.28)$$

donde la función $\psi(x)$ para $x \geq 2$, $x \in \mathbb{Z}^+$ esta definida como¹:

$$\psi(x) = \left\lfloor \frac{x}{2} \right\rfloor \quad (2.29)$$

La ecuación (2.28) se puede utilizar de manera recursiva desde el n -ésimo término hasta el segundo sin olvidar que cada uno de estos siempre debe ir acompañado de la suma de un uno. También vale la pena resaltar que el primer término de la expansión

es $\frac{1}{1-} \frac{\alpha x}{1}$ en conjunto.

Sustituyendo $x = s - 1$ y limitando el número de términos de la ecuación (2.25) obtenemos la aproximación racional para s^α y para obtener la aproximación racional de $\frac{1}{s^\alpha}$ la expresión tiene que ser simplemente invertida. En el apéndice A.2 se muestra un programa en MATLAB que calcula la aproximación para un integrador fraccionario de orden α eligiendo el número de términos n , utilizando el método de CFE descrito previamente.

En general la aproximación utilizando la CFE para un integrador fraccionario $\frac{1}{s^\alpha}$ utilizando los primeros dos términos resulta en una función de transferencia de primer orden como se muestra a continuación:

¹ $\lfloor x \rfloor$ es la función redondeo hacia el entero inferior anterior.

$${}_{(c_2)} \frac{1}{s^\alpha} \approx \frac{(1-\alpha)s + (1+\alpha)}{(1+\alpha)s + (1-\alpha)} \quad (2.30)$$

Al utilizar un número impar de términos el grado del numerador de la función de transferencia siempre será mayor en uno al del denominador, además de que el coeficiente de mayor grado del numerador siempre tendrá signo negativo, esto resulta problemático en la implementación y debido a estas observaciones es recomendable solo trabajar con un número par de términos.

La aproximación de segundo orden tiene la forma:

$${}_{(c_4)} \frac{1}{s^\alpha} \approx \frac{(\alpha^2 - 3\alpha + 2)s^2 + (8 - 2\alpha^2)s + (\alpha^2 + 3\alpha + 2)}{(\alpha^2 + 3\alpha + 2)s^2 + (8 - 2\alpha^2)s + (\alpha^2 - 3\alpha + 2)} \quad (2.31)$$

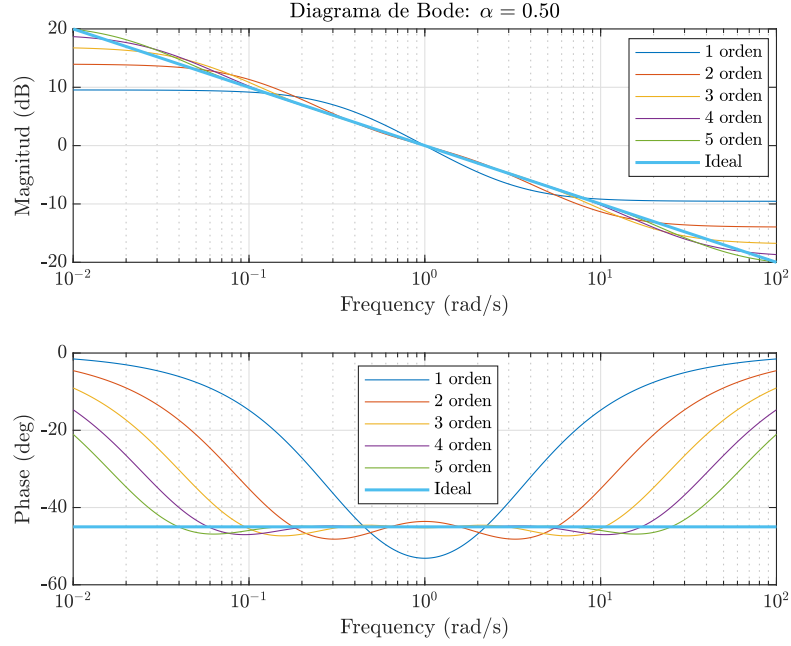
la ventaja de utilizar la aproximación de CFE es que convertimos el problema de orden fraccionario a uno de orden entero de manera sistemática. Por ejemplo para un integrador de orden fraccionario con $\alpha = 0.5$ sus aproximaciones son las mostradas en la Tabla 2.1 y sus correspondientes diagramas de bode se muestran en la Figura 2.1.

Tabla 2.1: Aproximaciones racionales de $\frac{1}{s^{0.5}}$

Orden	No. de términos	Aproximación racional
1	2	$\frac{s+3}{3s+1}$
2	4	$\frac{s^2+10s+5}{5s^2+10s+1}$
3	6	$\frac{s^3+21s^2+35s+7}{7s^3+35s^2+21s+1}$
4	8	$\frac{s^4+36s^3+126s^2+84s+9}{9s^4+84s^3+126s^2+36s+1}$
5	10	$\frac{s^5+55s^4+330s^3+462s^2+165s+11}{11s^5+165s^4+462s^3+330s^2+55s+1}$

Analizando la Figura 2.1 se puede resaltar que en general la aproximación aumenta su ancho de banda conforme el orden de la función de transferencia aumenta y que de mismo modo el error con respecto al integrador ideal disminuye, lo cual era de esperarse, el ancho de banda útil es de 10^{-1} rad/s hasta 10^1 rad/s, es decir aproximadamente dos décadas, no obstante un análisis más profundo es necesario para notar otras peculiaridades que ocurren en este tipo de aproximaciones.

Figura 2.1: Diagramas de bode comparativos de integrador fraccionario, funciones de transferencia de primer hasta quinto orden.



2.4.1. Análisis de error de la CFE

Si variamos el orden α del integrador y el orden de la función de transferencia aproximada podemos notar que entre más pequeño sea α más se alejada la aproximación con respecto a un integrador fraccionario ideal, esto considerando el ancho de banda útil (ver Figuras C.1, C.2). Para cuantificar lo anterior existen dos tipos de errores de interés, el error sin normalizar y el error normalizado.

El error en dB de la magnitud sin normalizar se puede calcular utilizando la siguiente ecuación:

$$\text{error}_{\text{dB}} = 20 \log_{10} \left| \frac{H(j\omega)}{G(j\omega)} \right| \quad (2.32)$$

donde $H(j\omega)$ es el integrador ideal $\frac{1}{s^\alpha}$ y $G(j\omega)$ es la función de transferencia aproximada del integrador utilizando CFE $(c_n) \frac{1}{s^\alpha}$.

El error en grados de la fase sin normalizar se obtiene utilizando la siguiente ecuación:

$$\text{error}_{\text{deg}} = \angle H(j\omega) - \angle G(j\omega) \quad (2.33)$$

Debido a que un integrador fraccionario ideal es en esencia un polo en el origen elevado a una potencia, este tiene un comportamiento de -20α dB/década en magnitud y de -90α grados en fase [26], para normalizar el error este hecho resulta importante ya que la magnitud del integrador ideal en 0.1 rad/seg para cualquier α siempre será 20α dB y para 10 rad/seg de -20α dB, de manera similar la fase permanece constante

en -90α grados para cualquier α .

Entonces la ecuación para el error normalizado de magnitud es:

$$\text{error}_{\text{norm de mag}} = \frac{20 \log_{10} \left| \frac{H(j\omega)}{G(j\omega)} \right|}{20\alpha} = \frac{\log_{10} \left| \frac{H(j\omega)}{G(j\omega)} \right|}{\alpha} \quad (2.34)$$

y la ecuación para el error normalizado de fase es:

$$\text{error}_{\text{norm de fase}} = \frac{\angle H(j\omega) - \angle G(j\omega)}{90\alpha} \quad (2.35)$$

Para poder comparar los errores de manera precisa se debe calcular el valor absoluto de los errores para tener una medida de cuanto se aleja del integrador fraccionario ideal. En las Tablas 2.2 y 2.3 se muestran los errores absolutos máximos sin normalizar para la magnitud y la fase dependiente de α y el orden de la función de transferencia, estos datos se puede corroborar dirigiéndose a las gráficas de las Figuras C.3 y C.4.

Tabla 2.2: Máximo error absoluto de magnitud en dB variando α y orden de función de transferencia.

α/Orden	1 ^{er}	2 ^{do}	3 ^{er}	4 ^{to}	5 ^{to}
0.1	0.4587	0.3333	0.2427	0.0620	0.0220
0.2	0.8931	0.6524	0.4665	0.1172	0.0423
0.3	1.2792	0.9426	0.6527	0.1596	0.0591
0.4	1.5942	1.1889	0.7838	0.1848	0.0710
0.5	1.8155	1.3748	0.8449	0.1904	0.0767
0.6	1.9193	1.4807	0.8259	0.1767	0.0755
0.7	1.8769	1.4676	0.7232	0.1458	0.0669
0.8	1.6464	1.2410	0.5415	0.1021	0.0510
0.9	1.1460	0.7507	0.2939	0.0513	0.0284

Tabla 2.3: Máximo error absoluto de fase en grados variando α y orden de función de transferencia.

α/Orden	1 ^{er}	2 ^{do}	3 ^{er}	4 ^{to}	5 ^{to}
0.1	6.7092	2.8727	0.6548	0.5632	0.2796
0.2	13.2833	5.5250	1.2598	1.0838	0.5338
0.3	19.5614	7.7297	1.7677	1.5208	0.7390
0.4	25.3200	9.2514	2.1362	1.8358	0.8760
0.5	30.2099	9.8632	2.3294	1.9960	0.9311
0.6	33.6307	9.3936	2.3198	1.9760	0.8975
0.7	34.4722	7.8155	2.0880	1.7611	0.7756
0.8	30.6494	5.3543	1.6237	1.3492	0.5739
0.9	19.0601	2.5243	0.9255	0.7528	0.3080

A simple vista se podría creer que el error máximo aumenta conforme aumenta α , no obstante esto es falso debido a que no se esta teniendo en cuenta la escala de la

gráfica y por lo tanto las Tablas 2.2 y 2.3 únicamente nos dan información del total de error y no del porcentaje. Para tomar en cuenta la escala es necesario normalizar el error y de mismo modo calcular el valor absoluto de este. Las gráficas de la respuesta de magnitud y fase normalizadas se pueden encontrar en las Figuras C.5 y C.6, y las gráficas de los errores normalizados en las Figuras C.7 y C.8.

En las Tablas 2.4 y 2.5 se muestran los porcentajes del máximo error absoluto normalizados para la magnitud y la fase, haciendo un análisis de estas se puede llegar a la conclusión de que en efecto **el porcentaje de error es mayor cuanto más pequeño sea α** y esto es de gran interés para poder generar reglas de diseño para la implementación de circuitos integradores fraccionarios.

Tabla 2.4: Máximo error absoluto de magnitud normalizado en % variando α y orden de función de transferencia.

α /Orden	1 ^{er}	2 ^{do}	3 ^{er}	4 ^{to}	5 ^{to}
0.1	22.9364	16.6661	12.1365	3.1006	1.1022
0.2	22.3267	16.3092	11.6624	2.9299	1.0576
0.3	21.3204	15.7107	10.8780	2.6595	0.9850
0.4	19.9281	14.8618	9.7974	2.3094	0.8869
0.5	18.1555	13.7481	8.4493	1.9044	0.7669
0.6	15.9941	12.3390	6.8823	1.4724	0.6290
0.7	13.4063	10.4826	5.1656	1.0415	0.4779
0.8	10.2900	7.7565	3.3847	0.6380	0.3190
0.9	6.3665	4.1706	1.6328	0.2848	0.1578

Tabla 2.5: Máximo error absoluto de fase normalizado en % variando α y orden de función de transferencia.

α /Orden	1 ^{er}	2 ^{do}	3 ^{er}	4 ^{to}	5 ^{to}
0.1	74.5462	31.9184	7.2756	6.2574	3.1070
0.2	73.7962	30.6945	6.9988	6.0212	2.9653
0.3	72.4496	28.6284	6.5472	5.6324	2.7370
0.4	70.3334	25.6983	5.9338	5.0995	2.4334
0.5	67.1331	21.9182	5.1765	4.4355	2.0692
0.6	62.2790	17.3955	4.2958	3.6593	1.6620
0.7	54.7178	12.4056	3.3143	2.7954	1.2312
0.8	42.5686	7.4365	2.2552	1.8739	0.7971
0.9	23.5310	3.1165	1.1426	0.9293	0.3802

Dado que el máximo error ocurre en un rango de frecuencias muy corto es recomendable conocer de igual manera el error promedio, en las Tablas 2.6 y 2.7 se muestra el error promedio normalizado, este tipo de error presenta el mismo efecto de porcentaje de error con respecto a α que el máximo error normalizado. Analizando los datos de las tablas podemos resaltar que en cuanto a magnitud, utilizar una aproximación de segundo orden en lugar de una de primer orden disminuye el error un 5.86 % para $\alpha = 0.1$

y un 2.96 % para $\alpha = 0.9$, en fase el cambio es aún mas notorio, el error disminuye un 26.71 % para $\alpha = 0.1$ y un 5.61 % para $\alpha = 0.9$. No obstante si nos detenemos a pensar detenidamente, con un $\alpha = 0.9$ y una aproximación de primer orden el error promedio es de 4.22 % en magnitud y de 6.55 % en fase, el cual para algunas aplicaciones puede resultar aceptable y cambiar por una aproximación de segundo orden no agregaría gran mejora, lo contrario ocurre con un $\alpha = 0.1$, al cambiar a una aproximación de segundo el error se reduce significativamente, aproximadamente a la mitad en magnitud y a un cuarto en fase, en este caso en definitiva vale la pena cambiar.

Tabla 2.6: Promedio de error absoluto de magnitud normalizado en % variando α y orden de función de transferencia.

α /Orden	1 ^{er}	2 ^{do}	3 ^{er}	4 ^{to}	5 ^{to}
0.1	13.4397	7.5755	2.4058	0.5407	0.2754
0.2	13.1089	7.3398	2.2960	0.5155	0.2639
0.3	12.5614	6.9432	2.1189	0.4751	0.2454
0.4	11.8034	6.3812	1.8827	0.4217	0.2203
0.5	10.8454	5.6502	1.5991	0.3580	0.1897
0.6	9.7075	4.7507	1.2822	0.2875	0.1548
0.7	8.4275	3.6945	0.9478	0.2133	0.1168
0.8	6.8606	2.5119	0.6123	0.1387	0.0772
0.9	4.2249	1.2563	0.2916	0.0667	0.0377

Tabla 2.7: Promedio error absoluto de fase normalizado en % variando α y orden de función de transferencia.

α /Orden	1 ^{er}	2 ^{do}	3 ^{er}	4 ^{to}	5 ^{to}
0.1	34.9335	8.2174	2.6873	1.3313	0.3667
0.2	34.0430	7.8450	2.5975	1.2737	0.3498
0.3	32.5319	7.2389	2.4512	1.1805	0.3226
0.4	30.3547	6.4219	2.2473	1.0556	0.2864
0.5	27.4424	5.4308	1.9840	0.9042	0.2431
0.6	23.6947	4.3175	1.6640	0.7329	0.1948
0.7	18.9857	3.1492	1.2933	0.5488	0.1439
0.8	13.2152	2.0009	0.8821	0.3598	0.0929
0.9	6.5510	0.9388	0.4450	0.1742	0.0441

De ser necesario para convertir el error de magnitud normalizado de % a dB se utiliza la siguiente ecuación:

$$\text{error}_{\text{dB}} = 20\alpha \cdot \left(\frac{\%_{\text{error mag}}}{100} \right) \quad (2.36)$$

y para el error de fase normalizado de % a grados:

$$\text{error}_{\text{deg}} = 90\alpha \cdot \left(\frac{\%_{\text{error fase}}}{100} \right) \quad (2.37)$$

2.5. Escalamiento en frecuencia

El escalamiento en frecuencia es el proceso de correr la respuesta en frecuencia de una red por arriba o por abajo del eje de frecuencia mientras se mantiene igual la impedancia [26].

El escalamiento de frecuencia se consigue multiplicando ésta por un factor de escalamiento k_f mientras se mantiene la impedancia igual. Si consideramos p la variable de frecuencia compleja actual y s la escalada, el proceso de escalamiento se define por la siguiente relación [27]:

$$s = k_f p \quad (2.38)$$

la ecuación 2.12 también se puede ver de la siguiente manera:

$$k_f = \frac{s}{p} = \frac{j\omega'}{j\omega} = \frac{\omega'}{\omega} \quad (2.39)$$

donde ω' es la frecuencia escalada y ω es la frecuencia actual de referencia.

Para entender de mejor manera el escalamiento en frecuencia analicemos el siguiente ejemplo, consideremos la siguiente función de transferencia:

$$N(p) = \frac{20p}{p^2 + 12p + 20} \quad (2.40)$$

la cual corta el eje de la frecuencia en 0.1 rad/s y 200 rad/s, si deseamos que la respuesta en magnitud permanezca idéntica pero desplazada a la derecha en un factor de cien, es decir, que las frecuencias escaladas corten al eje de frecuencia en 10 rad/s y 20k rad/s, entonces $k_f = \frac{10}{0.1} = 100$, sustituyendo la ecuación (2.38) en (2.40):

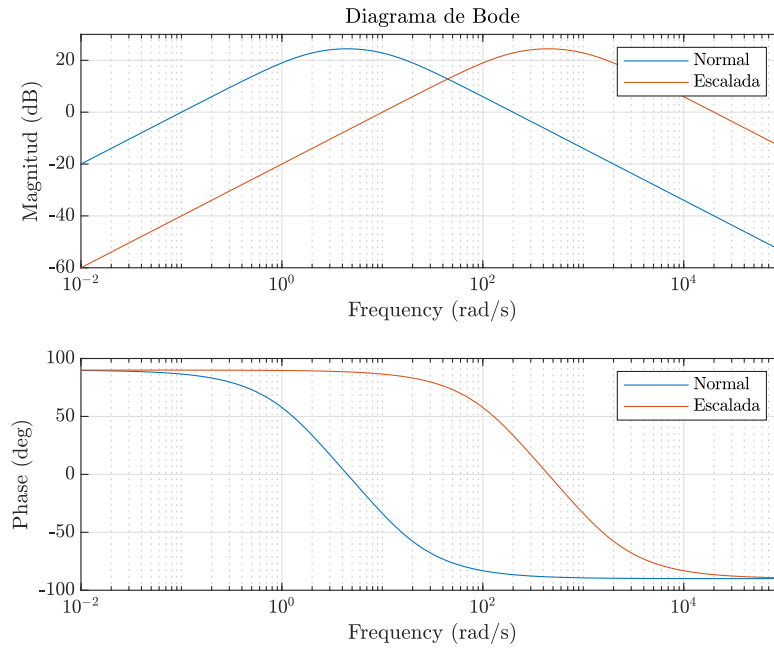
$$N(s) = \frac{200sk_f^{-1}}{s^2k_f^{-2} + 12sk_f^{-1} + 20} = \frac{200k_f s}{s^2 + 12k_f s + 20k_f^2} \quad (2.41)$$

entonces la función de transferencia escalada es:

$$N(s) = \frac{200(100)s}{s^2 + 12(100)s + 20(100)^2} \quad (2.42)$$

En la Figura 2.2 se muestra el resultado de aplicar el escalamiento en frecuencia a la ecuación 2.40. Si se desea aplicar el escalamiento en Hz, simplemente se tiene que tener en consideración el factor de conversión, por ejemplo, si se desea que las frecuencias que corten el eje de la frecuencia sean 0.1 Hz y 200 Hz entonces:

$$k_f = \frac{0.1 \text{ Hz} \cdot \frac{2\pi \text{ rad/s}}{1 \text{ Hz}}}{0.1 \text{ rad/s}} = 2\pi \quad (2.43)$$

Figura 2.2: Diagrama de bode comparativo de función de transferencia escalada.

2.6. Teoría de filtros

2.6.1. Filtros de primer orden

La función de transferencia general de primer orden esta dada por la siguiente ecuación:

$$T(s) = \frac{a_1 s + a_0}{s + \omega_0} \quad (2.44)$$

esta **función de transferencia bilineal** se caracteriza por tener un polo en $s = -\omega_0$, un cero en $s = -\frac{a_0}{a_1}$ y una ganancia de alta frecuencia que tiende a a_1 . Los coeficientes del numerador, a_0 y a_1 , determinan el tipo de filtro, ya sea pasabajas (LP), pasaaltas (HP), pasatodas (AP) o general. La realización activa provee considerablemente más versatilidad que su contra parte pasiva, en muchos casos la ganancia puede ser ajustada a un valor deseado, y algunos parámetros de las funciones de transferencia también sin afectar otros. La impedancia de salida de los circuitos activos es muy baja, haciendo que la colocación en cascada sea muy fácil. Sin embargo, los amplificadores operacionales limitan la operación a altas frecuencias de los circuitos activos [28].

Diferentes tipos de filtros nacen de ubicar el cero de la función de transferencia bilineal general (2.44) en distintos lugares, un cero en el infinito genera un filtro pasabajas de primer orden, y un cero en el origen un filtro pasaaltas. Las ecuaciones básicas de los filtros de primer orden son las siguientes:

Pasabajas (LP)

$$T(s) = \frac{a_0}{s + \omega_0} \quad (2.45)$$

Pasaaltas (HP)

$$T(s) = \frac{a_1 s}{s + \omega_0} \quad (2.46)$$

Pasatodas (AP)

$$T(s) = -a_1 \frac{s - \omega_0}{s + \omega_0} \quad (2.47)$$

2.6.2. Filtros de segundo orden

La función de transferencia general de un filtro de segundo orden (o **bicuadrática**) se expresa en lo general en la forma estándar:

$$T(s) = \frac{a_2 s^2 + a_1 s + a_0}{s^2 + \left(\frac{\omega_0}{Q}\right) s + \omega_0^2} \quad (2.48)$$

donde ω_0 y Q determinan los polos de acuerdo a la siguiente ecuación:

$$p_{1,2} = -\frac{\omega_0}{2Q} \pm j\omega_0 \sqrt{1 - \left(\frac{1}{4Q^2}\right)} \quad (2.49)$$

al parámetro ω_0 se le conoce como la **frecuencia de polo**, y representa la distancia radial entre el origen y el polo en el plano complejo s , a Q se le conoce como el **factor de calidad de polo** y determina la distancia de los polos desde el eje $j\omega$, una representación gráfica de estos parametros se muestra en la Figura 2.3, entre más grande sea el valor de Q , más cerca estarán los polos el eje $j\omega$, y la respuesta del filtro se vuelve más selectiva. Un valor infinito para Q localiza a los polos sobre el eje $j\omega$ y puede producir oscilaciones sostenidas en la realización del circuito. Un valor negativo de Q implica que los polos se encuentran en la mitad derecha del plano s , lo cual ciertamente produce oscilaciones. Los ceros del filtro de segundo orden están determinados por los coeficientes del numerador, a_0 , a_1 y a_2 . Los coeficientes del numerador determinan el tipo de filtro de segundo orden [28].

Para un filtro pasabajas (LP) de segundo orden, los dos ceros están en $s = \infty$ y su respuesta de magnitud muestra un pico para $Q > \frac{1}{\sqrt{2}}$. La respuesta obtenida con $Q = \frac{1}{\sqrt{2}}$ es la respuesta Butterworth o máxima plana. Para un pasaaltas (HP) los ceros están en $s = 0$ y su respuesta de magnitud muestra un pico para $Q > \frac{1}{\sqrt{2}}$. Para un pasabanda (BP) un cero esta en $s = \infty$ y el otro en $s = 0$, la respuesta de magnitud tiene un pico en $\omega = \omega_0$, por lo tanto, la **frecuencia central** del filtro pasabandas es igual a la frecuencia de polo ω_0 . La selectividad de del filtro pasabanda de segundo orden es usualmente medida por su ancho de banda de 3 dB. Este es la diferencia entre

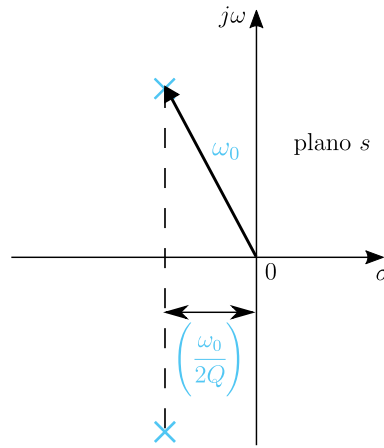
las dos frecuencias ω_1 y ω_2 en las cuales la respuesta de magnitud es 3 dB por debajo de su valor máximo el cual ocurre en ω_0 . La ecuación que describe a las frecuencias ω_1, ω_2 es la siguiente:

$$\omega_1, \omega_2 = \omega_0 \sqrt{1 + \left(\frac{1}{4Q^2}\right)} \pm \frac{\omega_0}{2Q} \quad (2.50)$$

Entonces el ancho de banda BW se define como:

$$BW = \omega_2 - \omega_1 = \frac{\omega_0}{Q} \quad (2.51)$$

Figura 2.3: Definición de los parámetros ω_0 y Q de un par de polos conjugados.



Las ecuaciones básicas de filtros de segundo orden son las siguientes:

Pasabajas (LP)

$$T(s) = \frac{a_0}{s^2 + \frac{\omega_0}{Q}s + \omega_0^2} \quad (2.52)$$

Pasaaltas (HP)

$$T(s) = \frac{a_2 s^2}{s^2 + \frac{\omega_0}{Q}s + \omega_0^2} \quad (2.53)$$

Pasabandas (BP)

$$T(s) = \frac{a_1 s}{s^2 + \frac{\omega_0}{Q}s + \omega_0^2} \quad (2.54)$$

Pasatodas (AP)

$$T(s) = a_2 \frac{s^2 - \frac{\omega_o}{Q}s + \omega_0^2}{s^2 + \frac{\omega_0}{Q}s + \omega_0^2} \quad (2.55)$$

Capítulo 3

Implementación

3.1. ¿Qué es una FPAA?

Una FPAA por sus siglas en inglés (Field Programmable Analog Arrays) es un dispositivo analógico equivalente a las FPGA (Field Programmable Gate Arrays). A diferencia de las FPGA que contienen una gran cantidad de módulos y conexiones que permiten configuraciones arbitrarias de lógica combinatorial y secuencial, los FPAA generalmente contienen una pequeña cantidad de CABs (Configurable Analog Blocks). Los FPAA dirigidos al diseño analógico estándar generalmente presentan un CAB que contiene un amplificador operacional, un arreglo de capacitores programables, y ya sea un arreglo de resistencias programables para circuitos en tiempo continuo o switches configurables para circuitos de capacitores conmutados. Se trabajó con la tarjeta Anadigm QuadApex Development Board v2.0 de la empresa Anadigm, la cual contiene 4 FPAAs AN231E04 que pueden conectarse en cadena y es programada mediante el software AnadigmDesigner2 (AD2). El diagrama esquemático de la tarjeta se puede ver en la Figura D.1 del apéndice.

3.2. Características de la tarjeta y requerimientos

3.2.1. Alimentación de la tarjeta

Para el correcto funcionamiento de la placa esta debe ser alimentada con una fuente de voltaje regulada a 5V de al menos 500mA conectada a la clema de dos terminales. Hay un LED de color verde que indica que la placa se ha encendido correctamente, la placa esta protegida contra la conexión de una fuente de voltaje con la polarización incorrecta.

3.2.2. Instalación de drivers

No conecte la placa QuadApex v2.0 a la PC vía cable USB, ni tampoco inicie AD2 ahora, el driver debe instalarse primero. Se asume que AD2 ya ha sido instalado y registrado en su computadora, de lo contrario entre al link que aparece a continuación.

Para instalar AD2 basta con acceder al siguiente link y seguir los pasos que muestra la pagina:

https://www.anadigm.com/sup_downloadcenter.asp?tab=ad2

es recomendable guardar los datos de registro en un lugar seguro, al iniciar el programa por primera vez es necesario ingresar el **License ID** y la **License key** estos estarán en el correo que le proporcionó a Anadigm.

El driver **CP210x_Drivers.exe** está incluido en el AD2 CD o se puede encontrar en la página de Silicon Labs. Siga los siguientes pasos si es la primera vez que instala este driver, de lo contrario desinstale las versiones anteriores antes de continuar:

1. Ejecute como administrador el ejecutable CP210x_Drivers.exe. El destino por defecto del ejecutable es "C:\Silabs\Mcu\CP210x".
2. Para completar la instalación conecte la QuadApex v2.0 y enciéndala.
3. Acceda al administrador de dispositivos y en **Puertos (COM y LPT)** asegúrese que el driver este bien configurado, si no aparece un signo de admiración y se le asigne un puerto COM la instalación fue exitosa, de lo contrario dar clic derecho sobre el dispositivo y seleccionar **Actualizar controlador**, después buscar el driver en la ruta del paso 1.

El driver en este punto ya está instalado. La instalación del driver y la asignación del puerto es necesaria solo una vez. Si se conecta subsecuentemente a otro puerto de USB de la PC puede ser necesario repetir el paso 3.

3.2.3. Jumpers por defecto

En la Tabla 3.1 se muestra una lista con los jumpers que contiene la placa, una descripción de su funcionamiento y su posición por defecto. Es importante colocar la FPAA con todos sus jumper en el estado por defecto para entender secciones posteriores y evitar errores. La placa no contiene una cantidad muy grande de jumpers y usualmente solo se modificarán los jumpers J8,9,10, los cuales son los que controlan la cantidad de FPPAs activas.

Tabla 3.1: Resumen de jumper de la placa.

Jumper	Función	Estado por defecto	Condición por defecto
J1	Conecta la sección digital a la sección analógica.	Todos los 13 jumpers deben estar conectados	Conecta completamente la alimentación, tierra y todas las señales digitales FPAA a la sección digital.
J2	Selecciona entre 16MHz y 40MHz para la fuente ACLK.	Jumpers off	ACLK=16MHz
J3	Permite la descarga de la configuración de prueba a todos los FPAA desde FLASH (después de reiniciar o apagar y encender). El circuito de prueba despliega ondas sinusoidales a todas las salidas FPAA.	Jumper off	Descarga del circuito de prueba deshabilitado
J4	Permite el almacenamiento de configuraciones primarias en el FLASH del PIC32 si el puente está activado.	Jumper off	No en modo de almacenamiento de configuraciones primarias.
J5	Jumper de repuesto.	Jumper off	-
J6	Un jumper en J5 deshabilitará el módulo oscilador de 16MHz y tristateará su salida. Esto significa que el pin ACLK de los FPAA no se sincronizará, tampoco se sincronizará el microcontrolador PIC32, por lo que toda la placa se desactivará efectivamente.	Jumper off	Oscilador de 16MHz habilitado
J7	Este jumper controla la generación de un suministro negativo (-3.3V) para las etapas del buffer de salida en la placa. El puente a la derecha desactiva el suministro negativo (lo ata al suelo). El puente a la izquierda permite el suministro.	Jumper a la izquierda	Alimentación negativa habilitada
J8,9,10	Estos jumpers en combinación con ACT1, ACT2, ACT3, ACT4 controlan la cadena de FPAAs.	Todos los jumpers especificados deben estar conectados	Todas las FPAAs en cadena
J11	Estos jumpers conectan el transceptor USB a la interfaz UART del microcontrolador PIC32. Si no se requiere control USB, estos 4 puentes se pueden quitar.	Todos los jumpers conectados	USB habilitado

3.2.4. Tamaño variable de cadena de FPAAs

La placa contiene 4 FPAAs en cadena. Esta cadena puede ser cortada a 3, 2 o 1, pero debe ser cortada deshabilitando FPAAs desde el final de la cadena (del lado derecho).

- Para reducir la cadena a 3 FPAAs, remueva los jumpers de **J10** y también el jumper marcado **ACT4** de **J1**. Esto deshabilitará FPPAA #4 (la más cercana a la fuente de alimentación.)
- Para reducir la cadena a 2 FPAAs, remueva los jumpers de **J9** y **J10** y también los jumpers marcados **ACT3** y **ACT4** de **J1**. Esto deshabilitará los FPAAs #3 y #4.
- Para reducir la cadena a 1 FPAA, remueva los jumpers de **J8**, **J9** y **J10** y también los jumpers marcados **ACT2**, **ACT3** y **ACT4** de **J1**. Esto deshabilitará los FPAA #2, #3 y #4.

Nota: si se configura una cadena de FPAA y luego se desconectan 1 o más FPAA de la cadena, los FPAA desconectados aún mantendrán su configuración hasta que se reinicie la placa o se vuelva a configurar la cadena (reducida).

3.2.5. DIP Switches

El usuario puede hacer sus propias conexiones en la placa con cables, pero hay un conjunto de DIP switches que permiten una fácil conexión de ciertas rutas entre los FPAA vecinos y entre los FPAA y los **input/output buffers**. Estos interruptores están abiertos por defecto, lo que significa que todos están hacia la izquierda. El usuario puede cerrar los interruptores empujándolos hacia la derecha. En la Tabla 3.2 se muestra un resumen de los switches. Los DIP switches son pequeños y del tipo deslizantes, por lo que se recomienda una herramienta afilada como un destornillador delgado para abrir y cerrar los switches. Algunos ejemplos se muestran a continuación:

1. Para conectar el filtro Rauch_#1_I01 a I1 de la FPAA#1, cierre los 4 interruptores en S10.
2. Para conectar O4 del FPAA# a I1 del FPAA#2, cierre los 2 interruptores superiores de S2.
3. Para conectar O3 del FPAA#1 al buffer de salida BUF_#1_03, cierre ambos switches de S13.

Función	Tipo	Labels
Conectar filtros Rauch a entradas de FPAA	4way	S8,9,10,11,15,16,17,18
Conectar entre FPAAs	4way	S2,3,4,5,6,7
Conectar FPAA a buffers de salida	2way	S12,13,14,19

Tabla 3.2: DIP Switches

3.2.6. Filtros Rauch y buffers de salida

La tarjeta cuenta con 2 buffers de entrada llamados filtros Rauch listos para usar, **Rauch_#1_I01** y **Rauch_#2_I01**, estos son filtros multipropósito cuya su principal función es convertir una señal single-ended a una diferencial en la FPAA, internamente la FPAA trabaja únicamente con señales diferenciales y debido a esto el uso de estos filtros es imprescindible para introducir señales externas, por ejemplo de generadores de funciones u otros circuitos. Si se usa una señal single-ended, es necesario conectar IN- a GND y conectar la señal en IN+. Para activar el filtro es necesario hacerlo desde el software AD2 haciendo doble clic en la IO cell apropiada (IOCell1-4), seleccionar Input y después Amplifier, esto es debido a que el filtro funciona en combinación con un amplificador integrado en la FPAA y los componentes pasivos de montaje superficial que vienen soldados de fabrica.

Rauch_#1_I01 está conectado a I/O1 de la FPAA #1 y esta configurado con una frecuencia de corte muy alta y ganancia unitaria ($F_o = 490\text{KHz}$). Rauch_#21_I01 está

conectado a I/O1 de la FPAA #2 y esta configurado con una ganancia unitaria y una frecuencia de corte de poco más de 20kHz, que es típica de una aplicación de audio.

La tarjeta también cuenta con 2 buffers de salida listos para usar, **Buf_#1_O3** y **Buf_#2_O3**, cuyo principal propósito es convertir la salida diferencial de la FPAA a una single-end. Buf_#1_O3 está conectado a O3 de la FPAA #1 y está configurado con una frecuencia de corte muy alta y una ganancia unitaria. Si se requiere una frecuencia de corte inferior, es muy fácil modificar este filtro sin quitar los componentes de montaje superficial debajo, simplemente agregando capacitores TH. Buf_#2_O3 está conectado a O3 de FPAA #2 y esta configurado con una ganancia unitaria y una frecuencia de corte de poco más de 20 kHz, que es típica de una aplicación de audio.

Cabe resaltar que para utilizar tanto los filtros Rauch como los buffers de salida sin cables adicionales es necesario cerrar los DIP switches correspondientes, no obstante se pueden cablear externamente a cualquiera de las entradas o salidas de las FPAA.


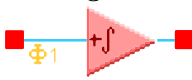

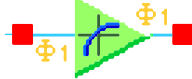
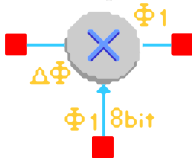

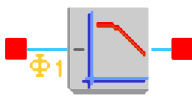

3.2.7. Circuito de prueba

La placa es capaz de autoconfigurar los FPAA con un circuito de prueba para una verificación rápida de la placa. Para hacer esto, coloque un jumper en **J3** y reinicie la placa. Todos los FPAA en la cadena ahora se configurarán con el circuito de prueba (el LED verde indica que la configuración fue exitosa). Este circuito de prueba genera una onda sinusoidal en todas las 7 salidas del FPAA (14 salidas diferenciales). La frecuencia de esta onda sinusoidal es de 1kHz en FPAA #1, 2kHz en FPAA #2, 4kHz en FPAA #3 y 8kHz en FPAA #4. Si se habilitan menos de 4 FPAA en la cadena, solo se configurarán aquellos FPAA que estén habilitados. Si se coloca un jumper en **J3** y **J4**, se cargará un circuito diferente en los FPAA. Este es un circuito de prueba especial utilizado por Anadigm durante la producción. Se aconseja al usuario que no coloque jumpers en J3 y J4 al mismo tiempo.

3.3. AnadigmDesigner2

AD2 trabaja con módulos llamados CAMs (Configurable Analog Modules), estos aportan flexibilidad y sencillez en el proceso diseño debido a que son bloques que hacen desde funciones sencillas como inversores o comparadores hasta diseños completos como filtros y multiplicadores. Los CAMs pueden interconectarse fácilmente unos con otros y únicamente necesitan pequeñas configuraciones para su correcto funcionamiento. Los CAMs más utilizados son los mostrados en la Tabla 3.3. En secciones posteriores se abordará con más detalle como configurarlos y cuales son las ecuaciones necesarias para calcular los parámetros de cada uno.

Tabla 3.3: CAMs básicos de AD2.

Nombre	Función de transferencia	Descripción
GainInv 	$\frac{V_{out}(s)}{V_{in}(s)} = -G$	Ganancia inversora. Gain: 0.01 - 100.0 V/V
Integrator 	$\frac{V_{out}(s)}{V_{in}(s)} = \frac{\pm K}{s}$	Integrador con una constante de integración programable. La salida puede ser inversora o no inversora.
Voltage 	$V_{out} = \pm 2$	Referencia de voltaje de ± 2 V.
TransferFunction 		Lookup Table: función de transferencia especificada por el usuario de 256 de pasos de cuantificación.
Multiplier 	$V_{out} = M \cdot V_x \cdot V_y$	V_x es la entrada de voltaje izquierda. V_y es la entrada de voltaje inferior cuantificado de 8 bits. M factor de multiplicación.
SumDiff 	$V_{out} = \pm G_1 V_{in1} \pm G_2 V_{in2} \pm G_3 V_{in3} \pm G_4 V_{in4}$	Las entradas pueden ser inversoras o no inversoras. Cada entrada tiene una ganancia programable. Configurable desde 2 hasta 4 entradas.
FilterBilinear 	Low Pass Bilinear Filter $\frac{V_{out}(s)}{V_{in}(s)} = \pm \frac{2\pi f_0 G}{s + 2\pi f_0}$ High Pass Bilinear Filter $\frac{V_{out}(s)}{V_{in}(s)} = -\frac{Gs}{s + 2\pi f_0}$ ⋮	Puede ser configurado como pasabajas, pasaaltas, pasatodas o general (Polo y cero).
FilterBiquad 	Low Pass Biquadratic Filter $\frac{V_{out}(s)}{V_{in}(s)} = \frac{\pm 4\pi^2 f_0^2 G}{s^2 + \frac{2\pi f_0}{Q}s + 4\pi^2 f_0^2}$ High Pass Biquadratic Filter $\frac{V_{out}(s)}{V_{in}(s)} = \frac{-Gs^2}{s^2 + \frac{2\pi f_0}{Q}s + 4\pi^2 f_0^2}$ ⋮	Puede ser configurado como pasabajas, pasaaltas, pasabanda, rechazabanda o general (Polos y ceros).

3.3.1. Comunicación con AD2

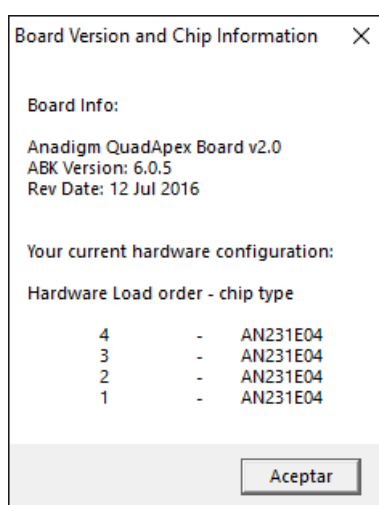
Ahora la placa puede ser programada lo que en este caso significa configurar los FPAA. Esto puede realizarse desde una PC o una laptop usando un cable estándar USB tipo A-B. La placa usa una emulación de comunicación serial de tal manera que

desde la computadora la placa aparecerá como un puerto COM.

Para programar la placa conecte el cable USB entre la placa y la computadora y encienda la placa. Abra AD2 y haga clic en **Settings/Preferences**, después haga clic en la pestaña **Port**. En el menú desplegable **Select Port** debe estar el puerto COM correspondiente a la placa Anadigm. Este es en realidad un puerto COM virtual porque la placa tiene un **USB/UART bridge** llamado CP2102 el cual emula un puerto serial.

Seleccione el puerto COM correspondiente. Haga clic en **Apply**, después en **OK**. Para comprobar que AD2 ahora puede comunicarse con la placa, haga clic en **Target/Display Board Information**. El LED verde D4 se apagará.

Figura 3.1: Información de la placa desplegada por AD2.



La Figura 3.1 muestra la información que despliega AD2 acerca de la placa donde todas las 4 FPAA están configuradas en cadena. Si menos FPAA están configuradas entonces la información de la placa mostrará cuantas FPAA están disponibles.

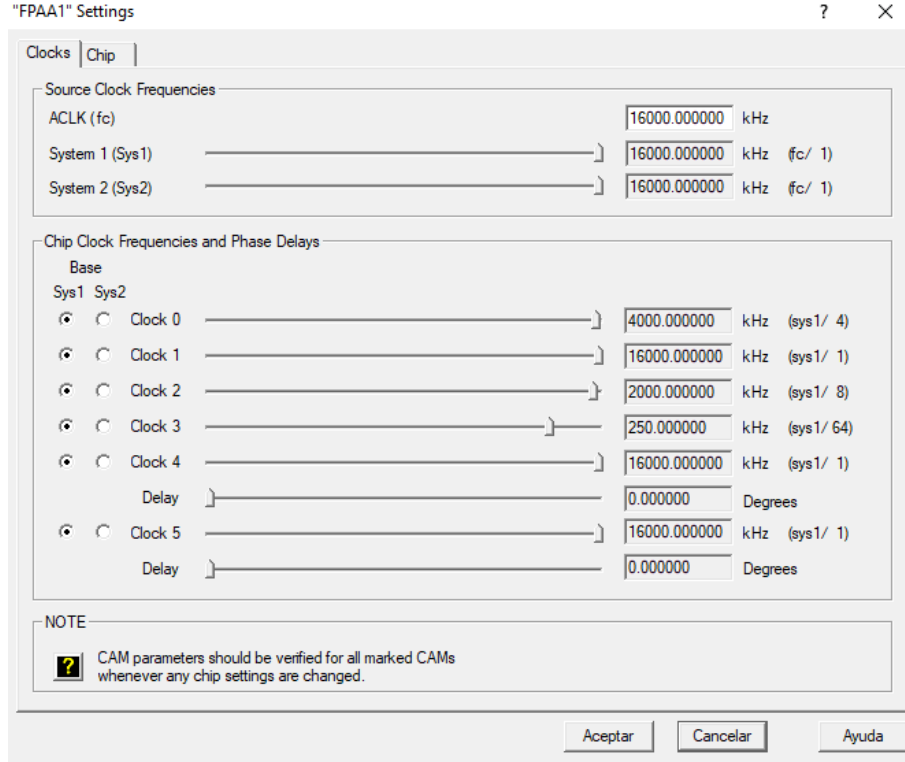
Ahora debería ser posible descargar un circuito desde AD2. Para hacer esto simplemente se da clic sobre el icono de descarga (a la izquierda del signo de interrogación, debajo de Dynamic Config). Como alternativa presiona **Ctrl + W**. Un LED amarillo comenzará a parpadear mientras cada FPAA es configurada y finalmente un LED verde se encenderá indicando que todas las FPAA han sido configuradas correctamente. Un LED rojo indicará que la configuración falló. Es importante que el número de FPAA en AD2 coincida con el número de FPAA habilitadas en la placa. Por ejemplo, si la tarjeta fue configurada con una cadena de 3 FPAA, entonces estará esperando 3 configuraciones de FPAA para ser descargadas desde AD2.

3.3.2. Configuración de relojes

Las configuraciones de los CAMs dependen de las frecuencias de reloj que se seleccionen para cada FPAA. Cada FPAA tiene dos fuentes de frecuencias de reloj de

sistema **Sys1** y **Sys2**, y seis frecuencias de reloj de chip, desde **Clock 0** hasta **Clock 5** que son subdivisiones de cualquiera de las fuentes de reloj sistema. Seleccionar y configurar correctamente los relojes es importante ya que el rango de operación de los CAMs depende de esto.

Figura 3.2: Ventana de configuración de frecuencias de reloj AD2.



La tarjeta tiene una fuente de frecuencia de reloj principal llamada **ACLK** o f_c cuya frecuencia es igual a 16 MHz y no puede ser cambiada. Sys1 y Sys2 dependen de esta frecuencia y pueden ser modificadas utilizando las siguientes ecuaciones:

$$\text{Sys1} = \frac{f_c}{m} \quad \text{Sys2} = \frac{f_c}{m} \quad (3.1)$$

donde $m \in [1, 510]$ y es par. Usualmente Sys1 y Sys2 no se modifican pero es necesario entender que estas dos fuentes de frecuencias de reloj a su vez controlan a las fuentes de reloj de chip desde **Clock 0** hasta **Clock 5** dada la siguiente ecuación:

$$\text{Clock } h = \frac{\text{Sys1}}{n} \quad \text{Clock } h = \frac{\text{Sys2}}{n} \quad (3.2)$$

donde $n \in [1, 510]$ y es par y $h \in [0, 5]$. Configurar las fuentes de reloj de chip cobrará importancia en secciones posteriores y las ecuaciones anteriores no deben olvidarse.

3.4. Implementación con aproximación de primer orden

Para realizar la implementación física de la aproximación de primer orden del integrador fraccionario se utilizó el CAM **FilterBilinear** en su modo **Pole and Zero**, esto debido a su flexibilidad y semejanza con la función de transferencia de la aproximación de la CFE de primer orden. La función de transferencia del CAM en este modo es la siguiente:

$$\frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = -\frac{G_H(s + 2\pi f_z)}{s + 2\pi f_p} \quad (3.3)$$

donde G_L esta definida como:

$$G_L = \frac{f_z}{f_p} G_H \quad (3.4)$$

y donde G_L es la ganancia en DC, G_H es la ganancia de alta frecuencia, f_p es la frecuencia del polo y f_z es la frecuencia del cero. Si se utiliza un CAM de ganancia inversora (**GainInv**) se puede prescindir del signo negativo de la ecuación (3.3).

La función de transferencia de la aproximación con la CFE de primer orden, como se mencionó anteriormente es la siguiente:

$$\frac{1}{(c_2) s^\alpha} \approx \frac{(1 - \alpha)s + (1 + \alpha)}{(1 + \alpha)s + (1 - \alpha)} \quad (3.5)$$

si consideramos la siguiente sustitución:

$$A = \frac{1 - \alpha}{1 + \alpha} \quad (3.6)$$

podemos reescribir la ecuación (3.5) de la siguiente manera:

$$\frac{1}{(c_2) s^\alpha} \approx \frac{As + 1}{s + A} \quad (3.7)$$

no obstante el ancho de bando de la ecuación (3.7) es de 10^{-1} rad/s hasta 10^1 rad/s, ver Figura 2.2, lo cual es un rango de frecuencias muy bajo y presenta dificultades en su implementación. Por esta razón se opta por aplicar un escalamiento en frecuencia a la ecuación (3.7) utilizando los siguiente pasos:

1. Cambiamos la variable compleja por p la cual representa la frecuencia actual:

$$N(p) = \frac{Ap + 1}{p + A} \quad (3.8)$$

2. Utilizamos la sustitución $p = \frac{s}{k_f}$:

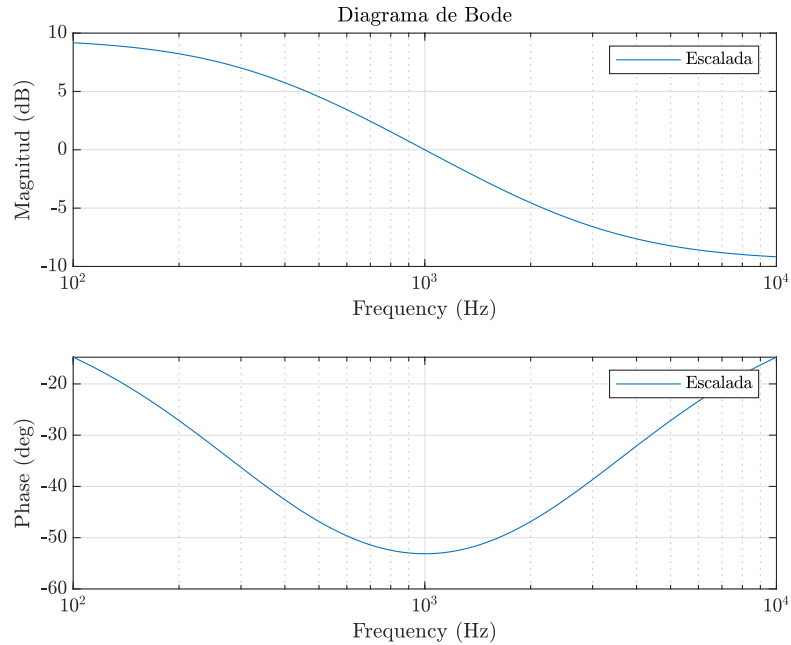
$$N(s) = \frac{Ask_f^{-1} + 1}{sk_f^{-1} + A} \quad (3.9)$$

3. Reescribimos de una manera más conveniente:

$$_{(c_2)} \frac{1}{s^\alpha} \approx \frac{As + k_f}{s + Ak_f} \quad (3.10)$$

El factor de escalamiento k_f se puede elegir a conveniencia dependiendo del ancho de banda necesario, no obstante la frecuencia de entrada de la mayoría de los CAM esta limitada aproximadamente a valores menores de 2 MHz, esto debido a que el tipo de procesamiento de señal del CAM se basa en arquitecturas de datos muestreados. Por esta razón se opta por un escalamiento con un factor $k_f = 2\pi 1000$. En la Figura 3.3 se muestra el resultado de aplicar el escalamiento. El nuevo ancho de banda abarca desde 100 Hz hasta 10 kHz el cual se encuentra dentro del rango de operación de los CAMs.

Figura 3.3: Diagrama de bode de ecuación (3.10) para un integrador fraccionario con $k_f = 2\pi 1000$ y $\alpha = 0.5$.



Para poder configurar el CAM FilterBilinear es necesario encontrar los valores correctos para los parámetros G_H , G_L , f_z y f_p dado un orden α , y esto se logra igualando las ecuaciones (3.10) y (3.3) con el fin de encontrar expresiones para calcularlos.

$$\frac{G_H(s + 2\pi f_z)}{s + 2\pi f_p} = \frac{As + k_f}{s + Ak_f} \quad (3.11)$$

de la ecuación (3.11) se deducen las siguientes ecuaciones:

$$G_H = A \quad (3.12)$$

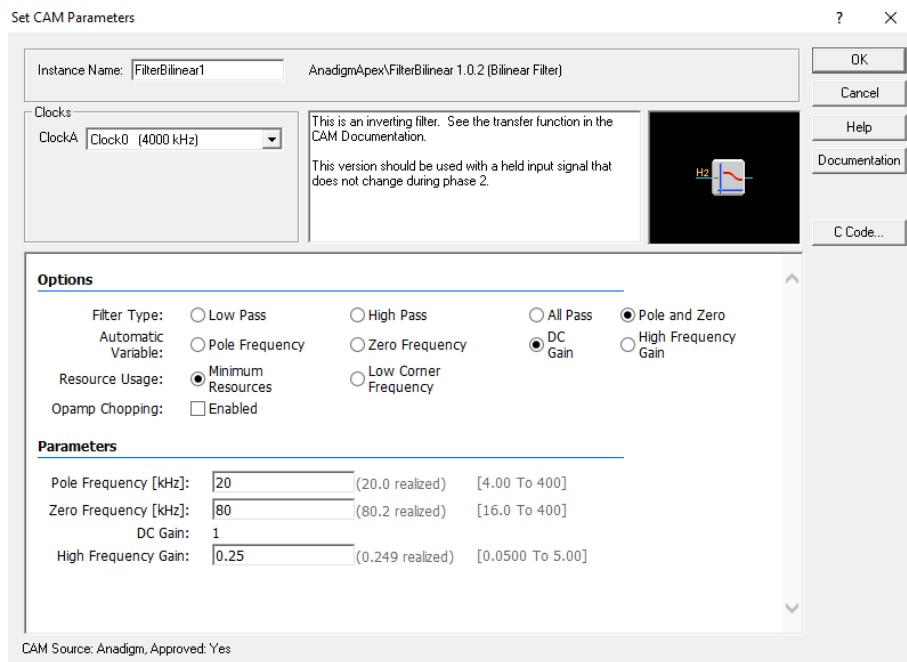
$$f_p = \frac{Ak_f}{2\pi} \quad (3.13)$$

$$f_z = \frac{k_f}{2A\pi} \quad (3.14)$$

$$G_L = \frac{1}{A} \quad (3.15)$$

utilizando las ecuaciones anteriores y considerando $k_f = 2\pi 1000$ obtenemos los datos de la Tabla 3.4, esta tabla se puede generar utilizando el código del apéndice A.3, sin embargo debido a que el CAM FilterBilinear depende directamente de la frecuencia de reloj que se seleccione en el parámetro **ClockA** no todos los valores de la tabla pueden ser ingresados, en la ventana de configuración aparecerán marcados en color rojo, por lo tanto es necesario hacer un análisis de frecuencias para este parámetro. En la ventana de configuración del CAM, ver Figura 3.4, se pueden ingresar 3 de los 4 parámetros requeridos, el parámetro restante es calculado automáticamente. Por simplicidad se eligió que el parámetro que se calcule automáticamente sea **DC Gain**.

Figura 3.4: Ventana de configuración de parámetros FilterBilinear.



Los límites absolutos de los valores que se pueden ingresar para las frecuencias f_z y f_p son $[\frac{F_c}{1000}, \frac{F_c}{10}]$ donde F_c es la frecuencia de reloj que se selecciona para el parámetro **ClockA** y los límites de ganancia están interrelacionados con los otros parámetros que pueden restringir el rango a menos que el límite absoluto el cual es $[0.01\frac{V}{V}, 100.0\frac{V}{V}]$.

Tabla 3.4: Valores para configurar un filtro bilineal como integrador de orden fraccionario en el rango de ordenes de 0.1 a 0.95.

α	f_p [kHz]	f_z [kHz]	G_L	G_H
0.10	0.818182	1.222222	1.222222	0.818182
0.15	0.739130	1.352941	1.352941	0.739130
0.20	0.666667	1.500000	1.500000	0.666667
0.25	0.600000	1.666667	1.666667	0.600000
0.30	0.538462	1.857143	1.857143	0.538462
0.35	0.481481	2.076923	2.076923	0.481481
0.40	0.428571	2.333333	2.333333	0.428571
0.45	0.379310	2.636364	2.636364	0.379310
0.50	0.333333	3.000000	3.000000	0.333333
0.55	0.290323	3.444444	3.444444	0.290323
0.60	0.250000	4.000000	4.000000	0.250000
0.65	0.212121	4.714286	4.714286	0.212121
0.70	0.176471	5.666667	5.666667	0.176471
0.75	0.142857	7.000000	7.000000	0.142857
0.80	0.111111	9.000000	9.000000	0.111111
0.85	0.081081	12.333333	12.333333	0.081081
0.90	0.052632	19.000000	19.000000	0.052632
0.95	0.025641	39.000000	39.000000	0.025641

Hay que tener presente que F_c puede ser cualquier Clock, desde el 0 hasta el 5, en otras palabras, F_c es igual a la ecuación (3.2). Un análisis de todas las posibles frecuencias para F_c dependiente de las subdivisiones dadas por n se muestran en la Tabla 3.5, si se desea ver la tabla completa se puede usar el código del apéndice A.4.

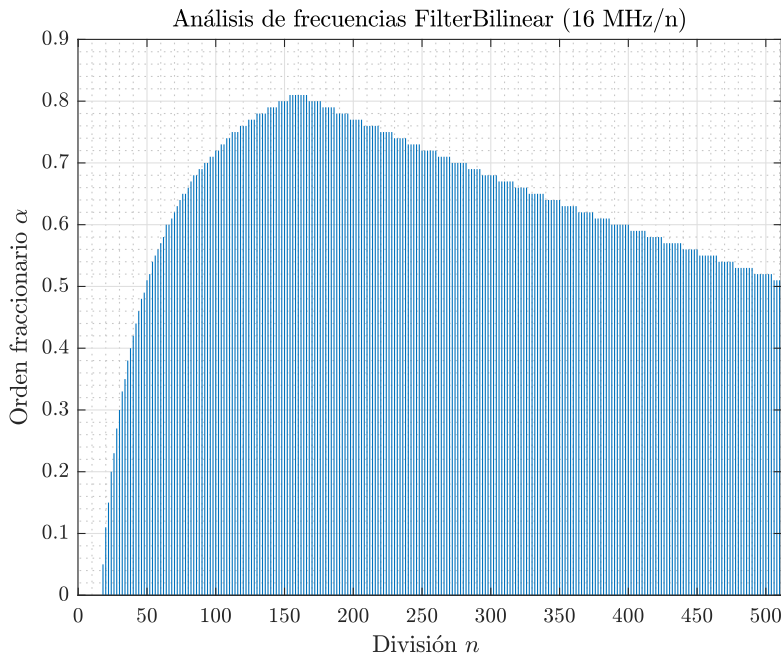
Tabla 3.5: Rango de frecuencias absolutas dependiente de F_c y el valor de n .

F_c [kHz]	n	min= $F_c/1000$ [kHz]	max= $F_c/10$ [kHz]
16000.0000	1.0000	16.0000	1600.0000
8000.0000	2.0000	8.0000	800.0000
4000.0000	4.0000	4.0000	400.0000
\vdots	\vdots	\vdots	\vdots
31.6206	506.0000	0.0316	3.1621
31.4961	508.0000	0.0315	3.1496
31.3725	510.0000	0.0314	3.1373

Para demostrar que no todos los valores de la Tabla 3.4 pueden ser implementados debido al rango de frecuencias absolutas mostradas en la Tabla 3.5 consideremos los siguientes ejemplos:

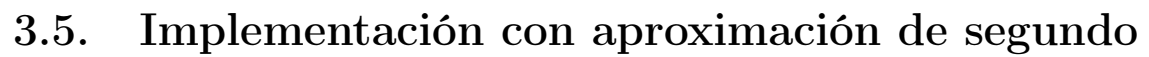
Imaginemos que queremos implemetar un integrador de orden fraccionario con la aproximación de primer orden, un $\alpha = 0.9$ y escalado en frecuencia para que su ancho de banda se encuentre entre 100 Hz y 10 kHz. Dados los requerimientos anteriores $k_f = 2\pi 1000$ y utilizando la Tabla 3.4 sabemos que $f_p = 0.052632$, $f_z = 19.0$, $G_L = 19.0$ y $G_H = 0.052632$. Como se mencionó anteriormente la fuente de frecuencia de reloj principal $f_c = 16$ MHz no puede modificarse y de mismo modo se recomienda que Sys1 y Sys2 tampoco se modifiquen, es decir $m = 1$. Se opta por utilizar el Clock 0 para configurar el CAM FilterBilinear y que este dependa de Sys1 = 16 MHz. Para que los datos que se ingresen en la ventana de configuración del CAM sean validos estos deben encontrarse dentro del rango de frecuencias absolutas de la Tabla 3.5, por lo tanto se tiene que elegir un n adecuado de tal manera que $F_c = \text{Clock 0} = \frac{\text{Sys1}}{n}$ cumpla con los limites absolutos de frecuencia, en otras palabras que los parámetros f_z y f_p se encuentren dentro del rango $[\frac{F_c}{1000}, \frac{F_c}{10}]$. El rango de frecuencias en el que nos tenemos que encontrar para poder implemetentar el integrador es $[f_p, f_z] = [0.052632, 19.0]$, no obstante el proceso de encontrar n es muy laborioso debido a que podría haber más de un valor que haga posible que se encuentren dentro del rango, de mismo modo podría ocurrir el caso contrario y ninguna n cumplir. Por esta razón en el Apéndice A.5 se muestra un código para resolver este problema, este genera una gráfica, ver Figura 3.5, la cual muestra la relación entre n y el orden α , en otras palabra hasta que orden fraccionario podemos implementar dependiendo del valor de n .

Figura 3.5: Análisis de orden fraccionario dependiente de n .



Analizando la Figura 3.5 podemos notar que $n = 160$ es el valor con el cual podemos implementar la mayor cantidad de integradores fraccionarios. Haciendo los cálculos

Figura 3.6: Implementación de integrador de orden fraccionario utilizando la aproximación de primer orden.



Capítulo 4

Analisis de no se que ahorita

Apéndice A

Códigos

```
1 % Convertir sym a funcion de transferencia
2 function R = syms2tf(G)
3     [symNum,symDen] = numden(G); % Obtener num y den funcion simbolica
4     TFnum = sym2poly(symNum);    % Convertir num sym a polinomio
5     TFden = sym2poly(symDen);    % Convertir den sym a polinomio
6     R = tf(TFnum,TFden);         % Generar funcion de transferencia
7 end
```

Código A.1: Función syms2tf

```
1 function R = cfetf(alfa,n)
2 % Calcula la aproximacion utilizando CFE de un integrador fraccional
3 % 1/s^(alfa)
4 %     R = cfetf(alfa,n)
5 %     alfa: es el orden del integrador
6 %     n    : es el numero de terminos de la aproximacion
7     syms s x;
8     eqns = sym(zeros(n,1));
9     for i=n:-1:2
10         if i == n
11             eqns(i) = 1 + n_term_cfe(i,alfa)*x;
12         else
13             eqns(i) = 1 + (n_term_cfe(i,alfa)*x)/eqns(i+1);
14         end
15     end
16     eqns(1) = 1/(1 - (alfa*x/eqns(2)));
17     derivate = simplify(subs(eqns(1),x,s-1));
18     integrator = collect(1/derivate);
19     sys = syms2tf(integrator);
20 %     pretty(integrator);
21     R = sys;
22 end
23 %% Funciones
24 function R = psi_cfe(x)
25     R = floor(x/2);
26 end
27
28 function R = n_term_cfe(n,k)
29     R = ( psi_cfe(n) * ( psi_cfe(n) + k*(-1)^n ) )/( (n-1)*(n));
30 end
```

Código A.2: Función cfetf

```

1 %% Clear subroutine
2 clear all;
3 close all;
4 clc;
5 %%
6 paso = 0.05;
7 alpha = (0.1:paso:0.95)';           % Orden del integrador
8 kf = 2*pi*1000;                     % Factor de escalamiento
9 A = (1 - alpha)./(1 + alpha);
10 Gh = A;                            % Ganancia de alta frecuencia
11 Gl = 1./A;                         % Ganancia de baja frecuencia
12 fp = (A*kf)./(2*pi*1000);          % Frecuencia de polo en kHz
13 fz = (kf)./(A*2*pi*1000);          % Frecuencia de zero en kHz
14 T = table(alpha,fp,fz,Gl,Gh);
15 disp(T);

```

Código A.3: Función A10_calculo_valores_FilterBilinear

```

1 %% Clean subroutine
2 clear all;
3 close all;
4 clc;
5 format short;
6 %%
7 sys1 = 16e6;                        % 16 MHz
8 n = [1, ( 2:2:510)];
9 freq = ((sys1./n)/1000)';           % en kHz
10 freq_min = freq/1000;
11 freq_max = freq/10;
12 T = table(freq, n', freq_min, freq_max);
13 disp(T);

```

Código A.4: Función A11_rango_de_frecuencias_absolutas_FilterBilinear

```

1 %% Clean subroutine
2 clear all;
3 close all;
4 clc;
5 format short;
6 %% Calcular fp y fz
7 paso = 0.01;
8 alpha = (paso:paso:0.99)';         % Orden fraccionario
9 kf = 2*pi*1000;                     % Factor de escalamiento
10 A = (1 - alpha)./(1 + alpha);
11 fp = (A*kf)./(2*pi*1000);          % Frecuencia de polo en kHz
12 fz = (kf)./(A*2*pi*1000);          % Frecuencia de cero en kHz
13
14 %% Rango de frecuencias absolutas
15 sys1 = 16e6;                        % 16 MHz
16 n = [1, ( 2:2:510)];                % n
17 freqs = ((sys1./n)/1000)';          % in kHz
18 freq_min = freqs/1000;
19 freq_max = freqs/10;
20 %%
21 vv = zeros(size(freqs,1),9);        % Matriz binaria de verdad
22 for i=1:size(freqs,1)               % Frecuencias
23     for j=1:size(alpha,1)           % Ordenes
24         cond1 = fp(j) >= freq_min(i) && fp(j) <= freq_max(i);

```

```

25     cond2 = fz(j) >= freq_min(i)  && fz(j) <= freq_max(i);
26     if cond1 && cond2
27         vv(i,j) = 1;
28     else
29         vv(i,j) = 0;
30     end
31 end
32 end
33 R = [freqs freq_min freq_max vv];
34 % csvwrite('Analisis.csv',R);
35 %%
36 T = sum(vv,2);      % Suma horizontal
37 Tp = T*paso;       % Conversion
38 %% Graficar
39 bar(n,Tp,'DisplayName','Orden');
40 title('An\'\'alisis de frecuencias FilterBilinear (16 MHz/n)','interpreter','latex');
41 xlabel('Divisi\'\'on $n$ ','interpreter','latex');
42 ylabel('Orden fraccionario $\alpha$','interpreter','latex');
43 % legend('interpreter','latex','FontSize',10);
44 grid on;
45 grid minor;
46 set(gca,'TickLabelInterpreter','latex');

```

Código A.5: Función A12_grafica_analisis_de_frecuencias_FilterBilinear

Apéndice B

Diagramas de flujo

Apéndice C

Gráficas de análisis de integrador fraccionario con CFE

Figura C.1: Diagramas de magnitud de aproximaciones de integrador fraccionario general.

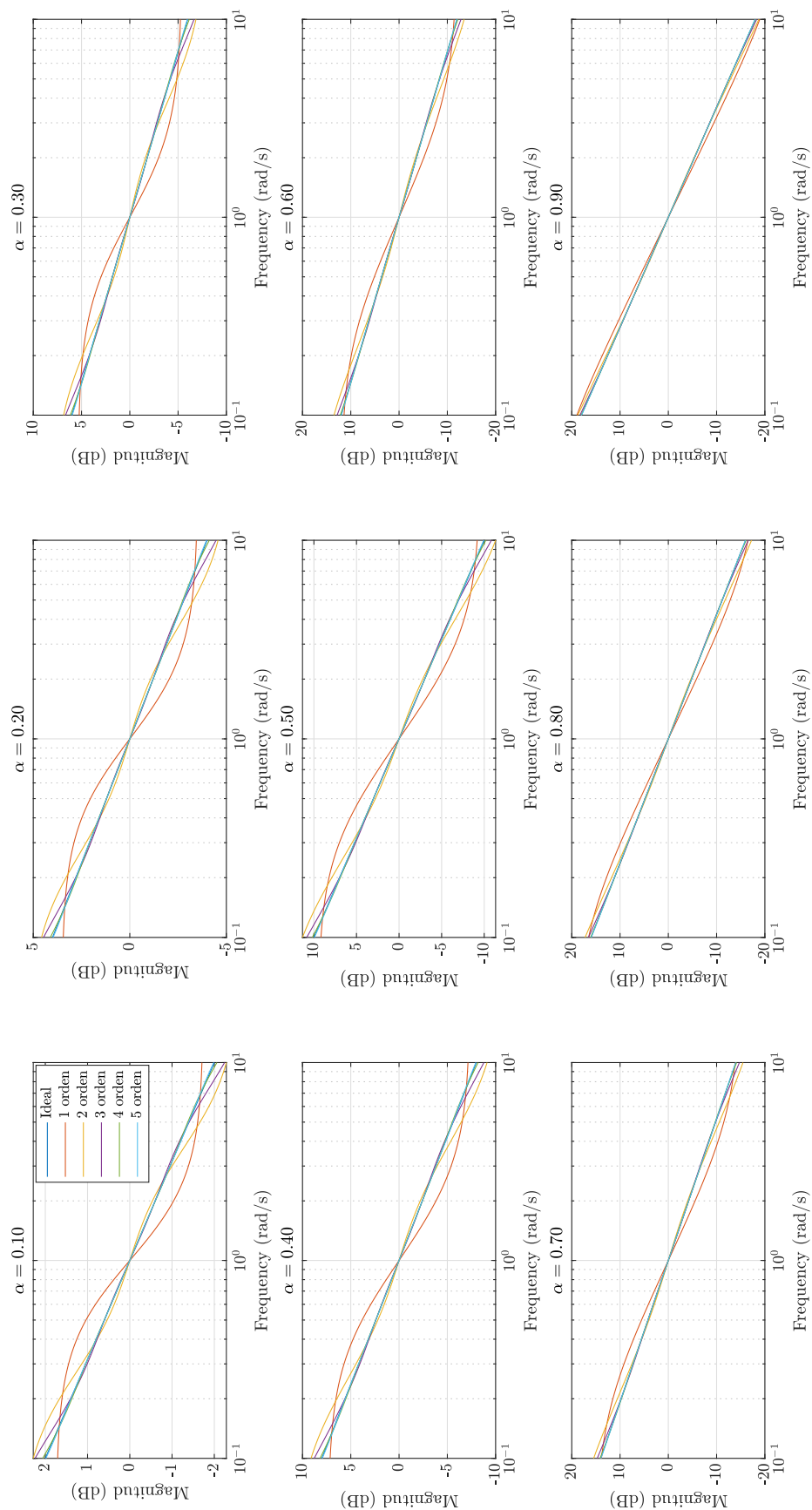


Figura C.2: Diagramas de fase de aproximaciones de integrador fraccionario general.

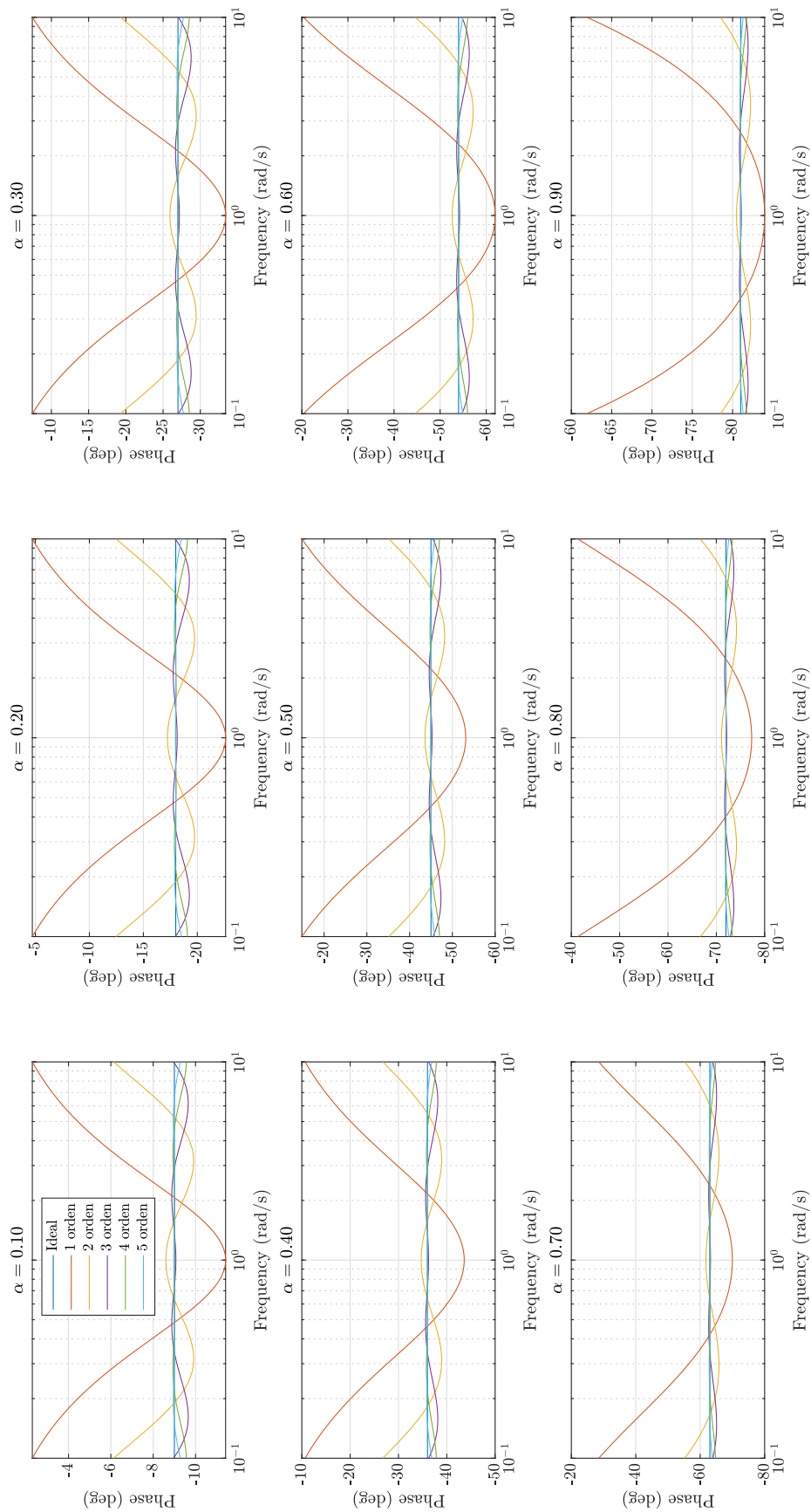


Figura C.3: Diagramas de error de magnitud de aproximaciones de integrador fraccionario general.

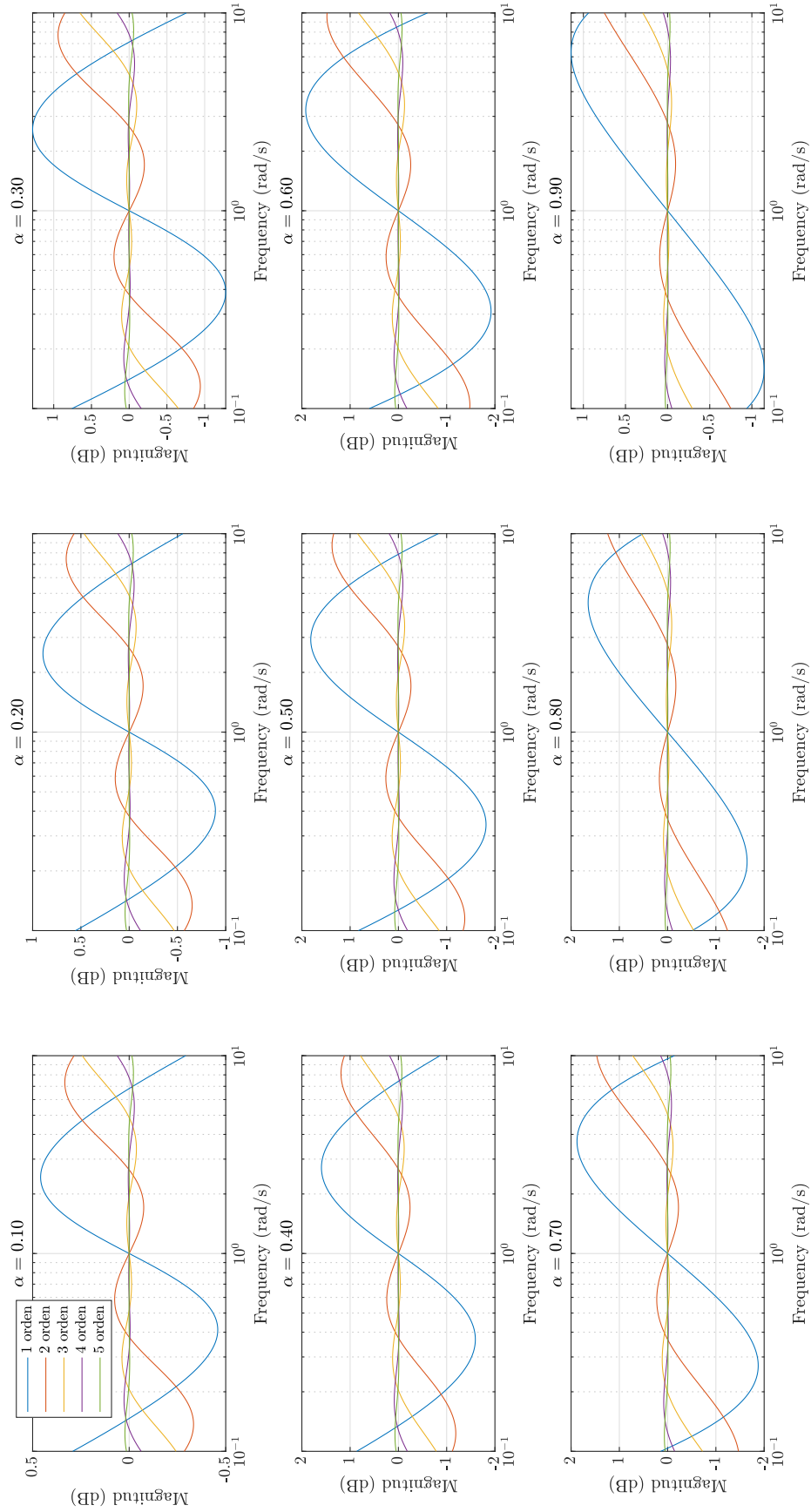


Figura C.4: Diagramas de error de fase de aproximaciones de integrador fraccionario general.

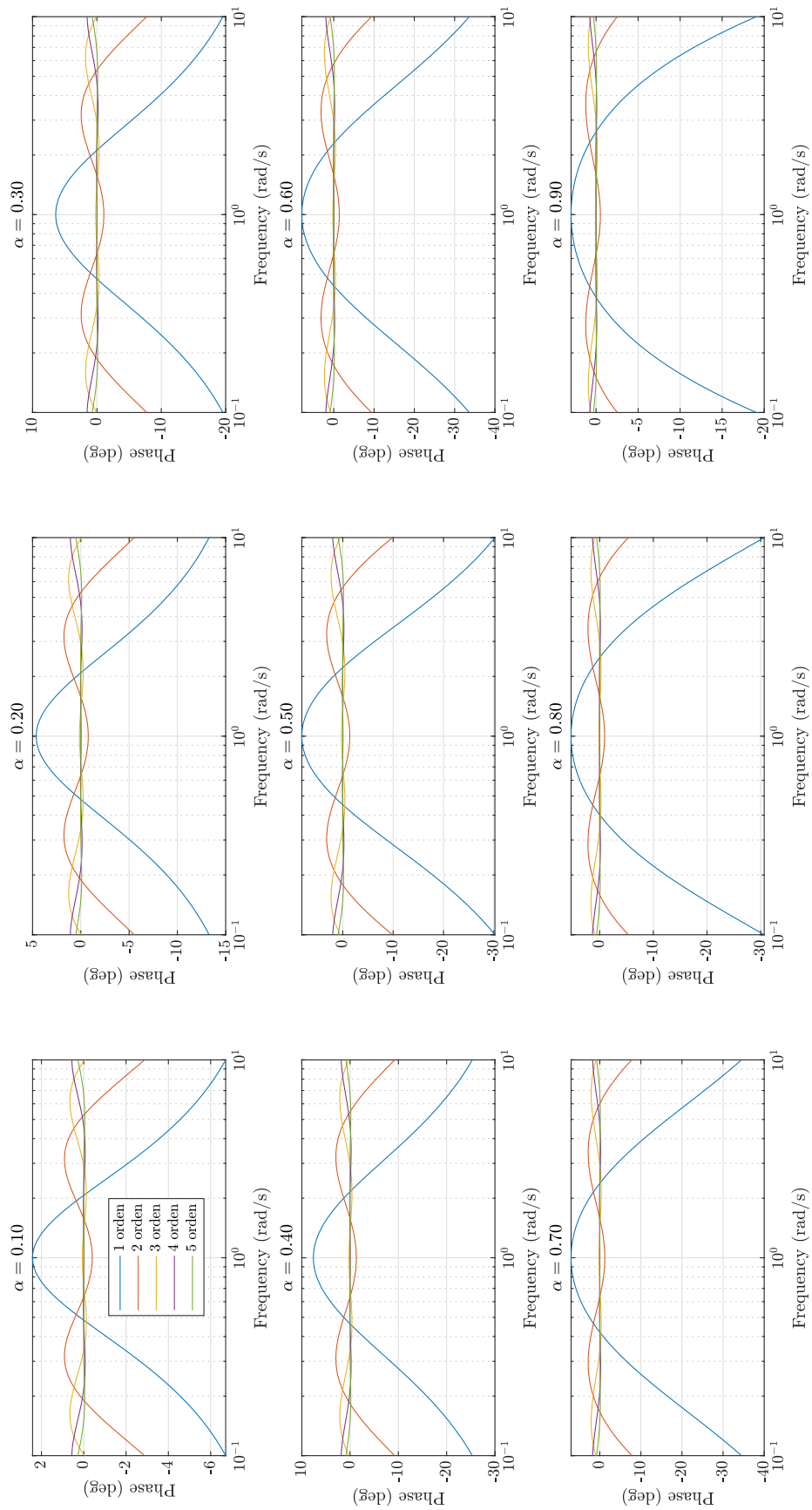


Figura C.5: Diagramas de magnitud normalizada de aproximaciones de integrador fraccionario general.

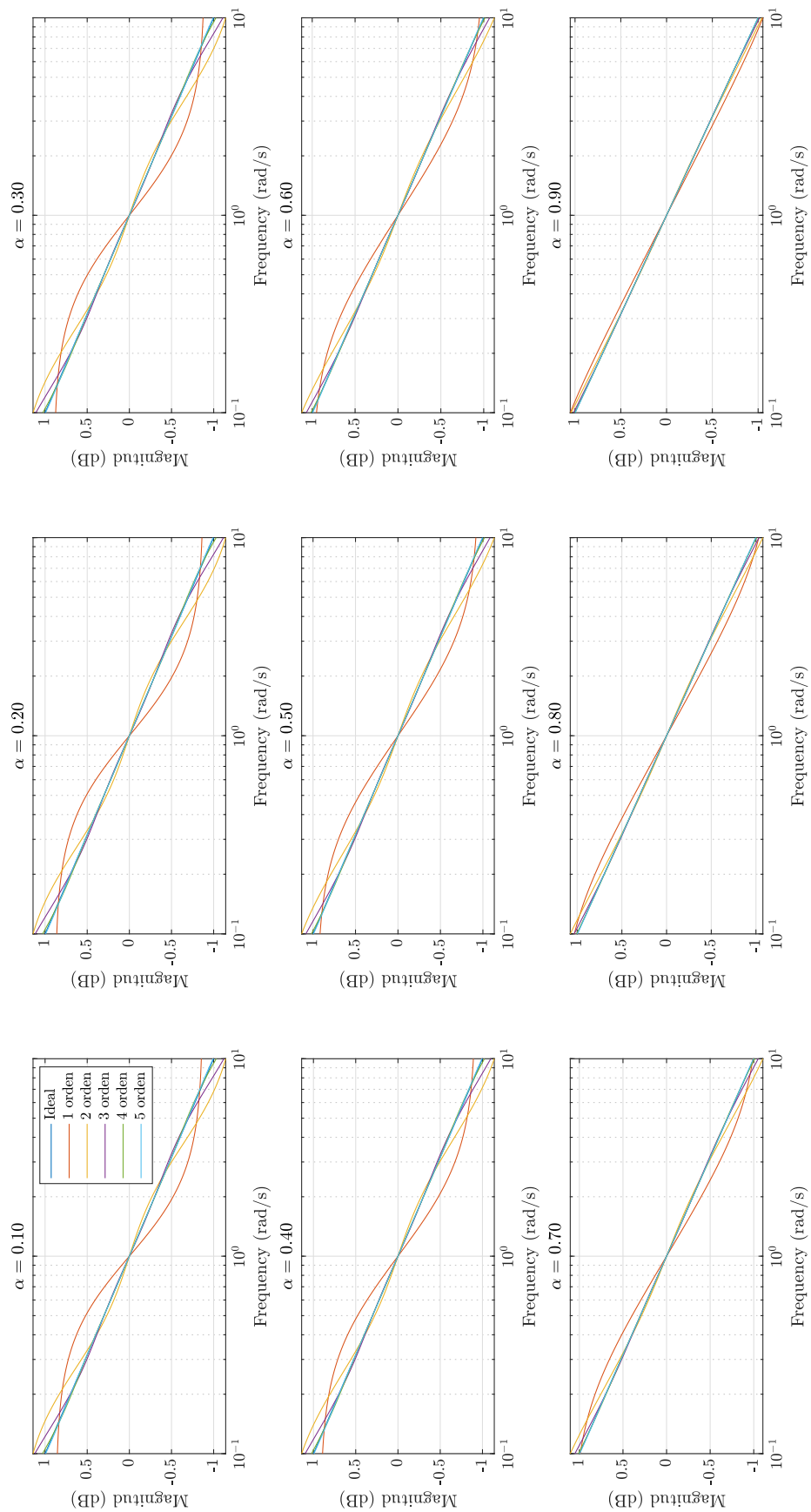


Figura C.6: Diagramas de fase normalizada de aproximaciones de integrador fraccional general.

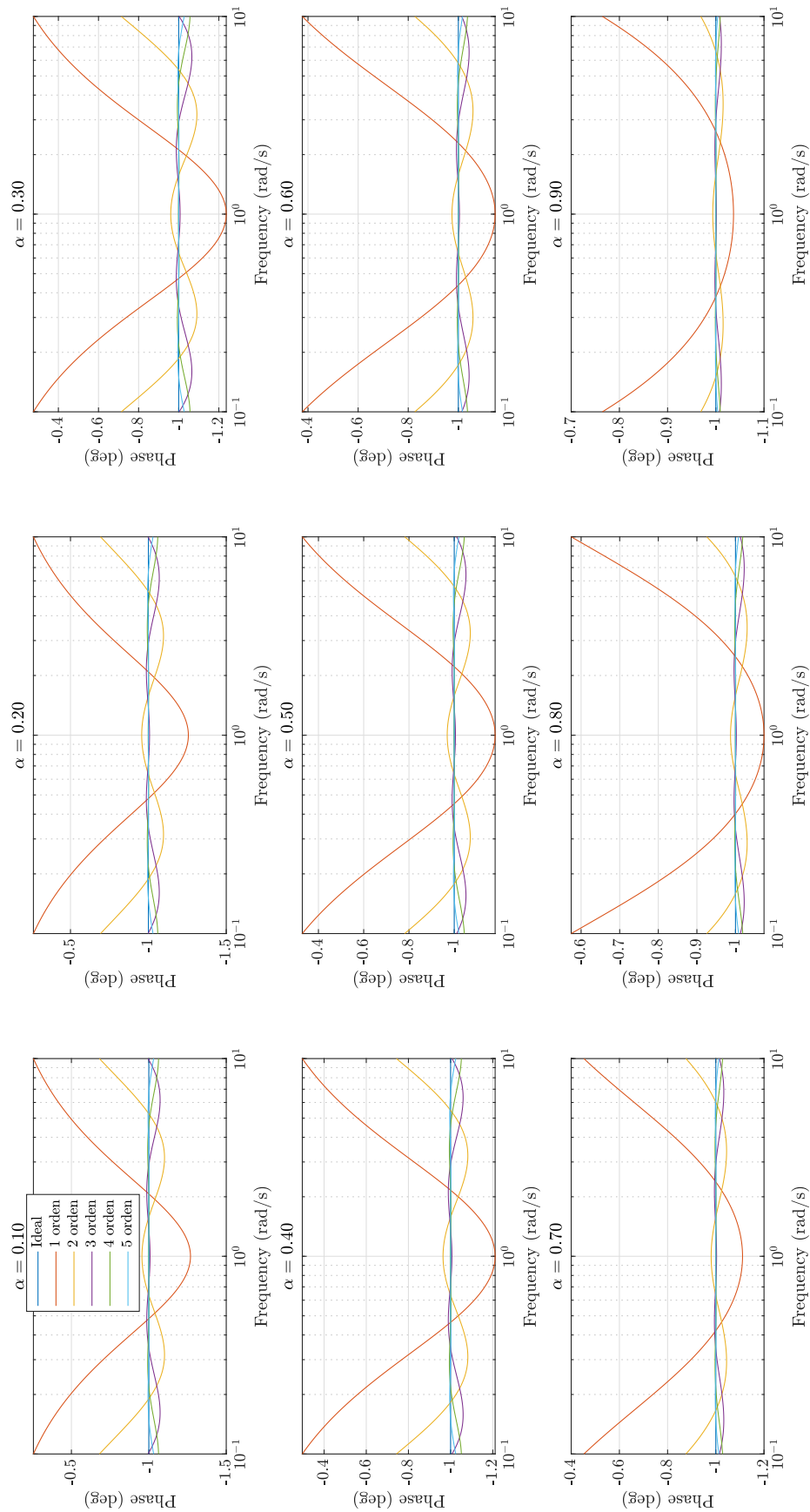


Figura C.7: Diagramas de error de magnitud normalizada de aproximaciones de integrador fraccionario general.

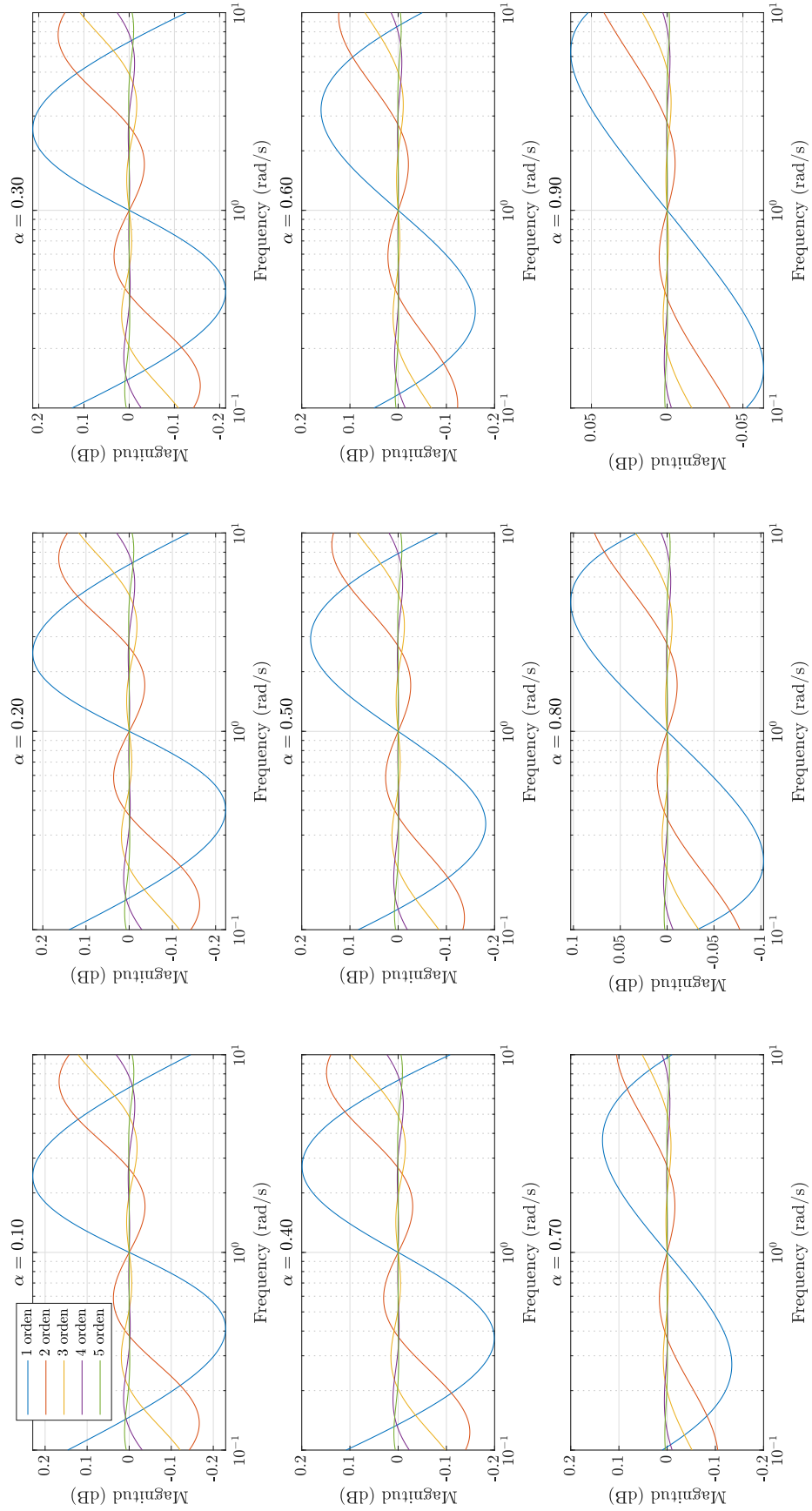
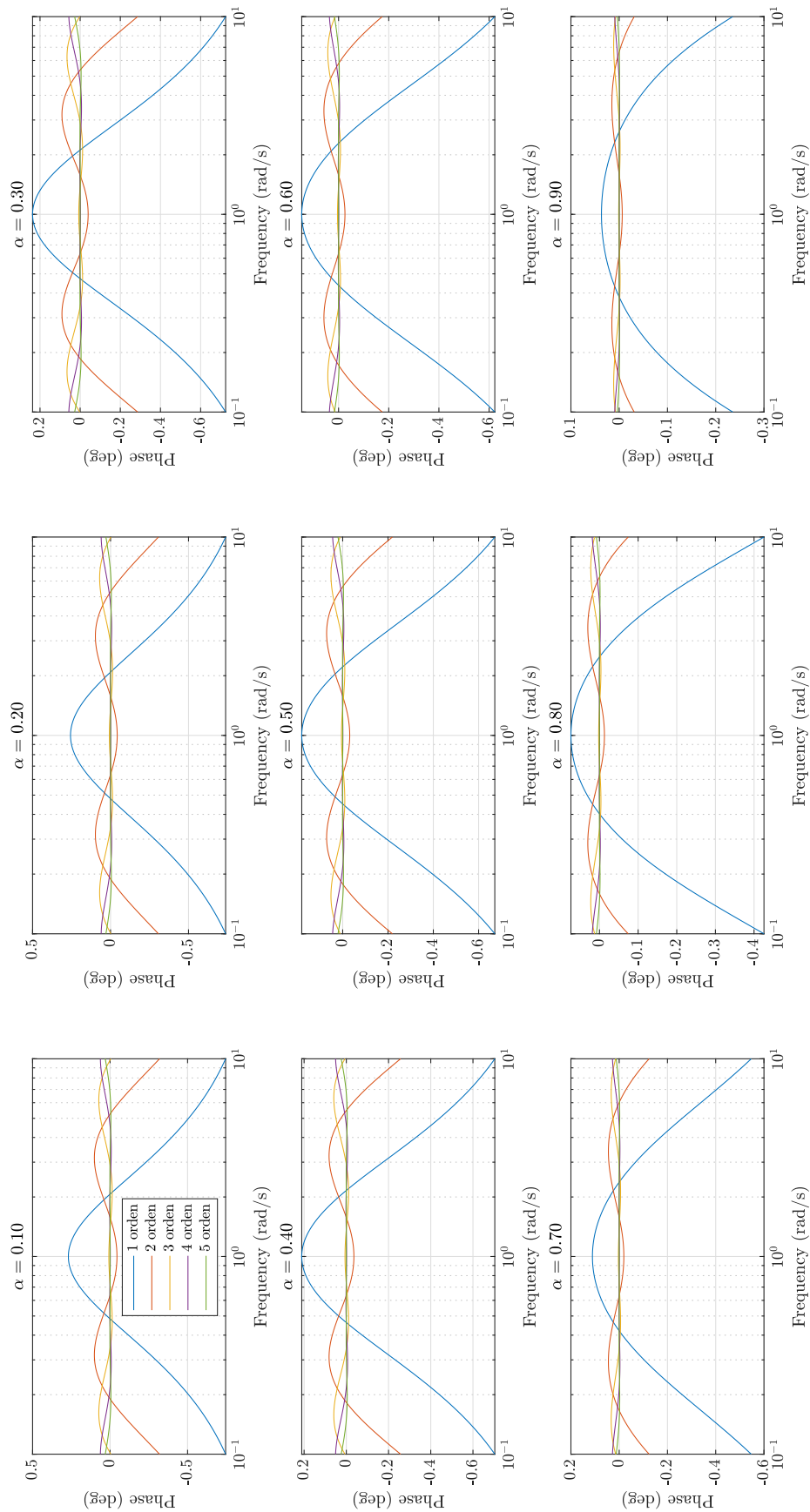


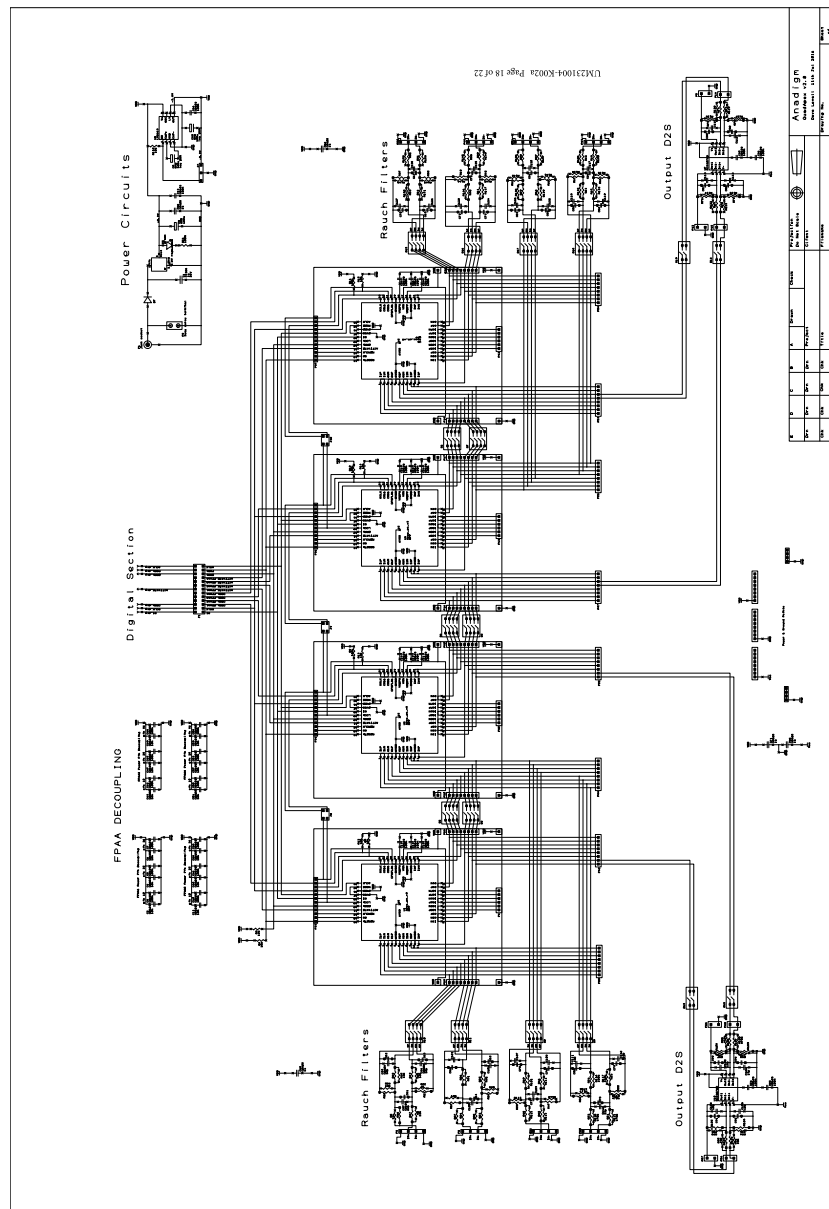
Figura C.8: Diagramas de error de fase normalizada de aproximaciones de integrador fraccionario general.



Apêndice D

Esquemático de QuadApex v2.0

Figura D.1: Diagrama esquemático de QuadApex v2.0



Bibliografía

- [1] J. M. M. Pacheco and E. T. Cuautle, *Electronic Design Automation of Multi-Scroll Chaos Generators*. BENTHAM SCIENCE PUB, 2010.
- [2] A. Buscarino, L. Fortuna, M. Frasca, and G. Sciuto, *A Concise Guide to Chaotic Electronic Circuits*. Springer-Verlag GmbH, 2014.
- [3] I. Petráš, *Fractional-Order Nonlinear Systems*. Springer Berlin Heidelberg, 2011.
- [4] I. Petráš and J. Terpak, “Fractional calculus as a simple tool for modeling and analysis of long memory process in industry,” *Mathematics*, vol. 7, p. 511, jun 2019.
- [5] A. Tepljakov, E. A. Gonzalez, E. Petlenkov, J. Belikov, C. A. Monje, and I. Petráš, “Incorporation of fractional-order dynamics into an existing PI/PID DC motor control loop,” *ISA Transactions*, vol. 60, pp. 262–273, jan 2016.
- [6] S. W. Khubalkar, A. S. Junghare, M. V. Aware, A. S. Chopade, and S. Das, “Demonstrative fractional order – PID controller based DC motor drive on digital platform,” *ISA Transactions*, vol. 82, pp. 79–93, nov 2018.
- [7] M. D. Ortigueira, *Fractional Calculus for Scientists and Engineers*. Springer-Verlag GmbH, 2011.
- [8] G. Wang, D. Chen, J. Lin, and X. Chen, “The application of chaotic oscillators to weak signal detection,” *IEEE Transactions on Industrial Electronics*, vol. 46, pp. 440–444, apr 1999.
- [9] V. Tepin, *Self-parametric chaotic oscillators for secure communication systems*. St. Petersburg State Polytech. Univ, 2002.
- [10] Y. Chen, I. Petras, and D. Xue, “Fractional order control - a tutorial,” *American Control Conference*, 2009.
- [11] S. Das, *Functional Fractional Calculus for System Identification and Controls*. Springer-Verlag GmbH, 2007.

- [12] E. Gunay and K. Altun, "A performance comparison study of programmable platforms: FPAA and FPGA implementation of COOK communication system," *European Conference on Circuit Theory and Design (ECCTD)*, sep 2017.
- [13] I. S. Jesus and J. A. T. Machado, "Development of fractional order capacitors based on electrolyte processes," *Nonlinear Dynamics*, vol. 56, pp. 45–55, jun 2008.
- [14] K. Biswas, S. Sen, and P. Dutta, "Realization of a constant phase element and its performance study in a differentiator circuit," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, pp. 802–806, sep 2006.
- [15] A. Charef, "Analogue realisation of fractional-order integrator, differentiator and fractional $PI^\lambda D^\mu$ controller," *IEE Proceedings - Control Theory and Applications*, vol. 153, pp. 714–720, nov 2006.
- [16] B. Krishna, "Studies on fractional order differentiators and integrators: A survey," *Signal Processing*, vol. 91, pp. 386–426, Mar. 2011.
- [17] B. T. Krishna and K. V. V. S. Reddy, "Active and passive realization of fractance device of order $1/2$," *Active and Passive Electronic Components*, vol. 2008, pp. 1–5, 2008.
- [18] A. Tepljakov, E. Petlenkov, and J. Belikov, "Efficient analog implementations of fractional-order controllers," *Proceedings of the 14th International Carpathian Control Conference (ICCC)*, may 2013.
- [19] L. Dorcak, J. Terpak, I. Petras, J. Valsa, and E. Gonzalez, "Comparison of the electronic realization of the fractional-order system and its model," *Proceedings of the 13th International Carpathian Control Conference (ICCC)*, may 2012.
- [20] N. Fragoulis, G. Souliotis, D. Besiris, and K. Giannakopoulos, "Field programmable analogue array design based on the wave active filter design method," *AEU - International Journal of Electronics and Communications*, vol. 63, pp. 889–895, oct 2009.
- [21] R. Caponetto and D. Porto, "Analog implementation of non integer order integrator via field programmable analog array," *IFAC Proceedings Volumes*, vol. 39, pp. 107–111, jan 2006.
- [22] C. Li, W. J.-C. Thio, J. C. Sprott, H. H.-C. Iu, and Y. Xu, "Constructing infinitely many attractors in a programmable chaotic circuit," *IEEE Access*, vol. 6, pp. 29003–29012, 2018.

- [23] F. Jiang, X. Wang, J. Jin, and D. Yang, *The application of chaotic duffing oscillators to ballistocardiograph signal detection*. IEEE, jul 2010.
- [24] B. Kumari and N. Gupta, “Experimental investigation on chaotic oscillator coupled dielectric resonator antenna for medical applications,” *IEEE International Conference on Antenna Innovations & Modern Technologies for Ground, Aircraft and Satellite Applications (iAIM)*, nov 2017.
- [25] C. D. Olds, *Continued Fractions*. The Mathematical Association of America, 2009.
- [26] M. S. Charles Alexander, *Fundamentals of Electric Circuits*. McGraw-Hill Education, 2016.
- [27] L. P. Huelsman and P. E. Allen, *Introduction to the Theory and Design of Active Filters (Electrical Engineering Series)*. McGraw-Hill Book Company, 1980.
- [28] K. S. Adel S. Sedra, *Microelectronic Circuits*. Oxford University Press Inc, 2015.