ANÁLISE E IMPLEMENTAÇÃO DE FILTROS DIGITAIS FRACCIONÁRIOS

Ricardo Jorge Costa Barbosa

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
2012

Este relatório satisfaz, parcialmente, os requisitos que constam da Unidade Curricular de de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Eletrotécnica e de Computadores, ramo de Automação e Sistemas

Candidato: Ricardo Jorge Costa Barbosa, Nº 1050400, 1050400@isep.ipp.pt
Orientação Científica: Ramiro de Sousa Barbosa, rsb@isep.ipp.pt
Co-Orientação Científica: Maria Isabel de Castro Lopes Martins Pinto Ferreira, mis@isep.ipp.pt

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
28 de novembro de 2012

Aos meus pais, avós e namorada.

Agradecimentos

Gostaria de agradecer todas as pessoas que, de alguma forma, contribuíram para a realização deste trabalho final de curso e me ajudaram a ultrapassar as dificuldades sentidas.

É com orgulho que digo que me encontro a finalizar o meu percurso académico na melhor escola de Engenharia de Portugal e uma das melhores escolas do mundo de Engenharia, o Instituto Superior de Engenharia do Porto.

Ao meu orientador, Sr. Eng.º Ramiro Barbosa, por toda a ajuda que me prestou ao longo deste trabalho, pela motivação que me transmitiu, pela cedência de equipamento e instalações, por todos os seus conselhos e incentivos, que me serviram de referência e permitiram enriquecer os meus conhecimentos a todos os níveis.

À minha coorientadora, Sra. Eng.ª Isabel Martins, agradeço toda a disponibilidade e ajuda demonstrada no âmbito da DSP.

Agradeço efusivamente por terem aceite o desafio de me acompanhar neste projeto aos orientadores anteriormente referidos.

Por fim, agradeço especialmente a minha família e namorada Marina Junqueira e também aos amigos pelo apoio incondicional a cada momento e pela confiança que sempre demonstraram em mim.

Resumo

O projeto realizado teve como tema a aplicação das derivadas e integrais fraccionários para a implementação de filtros digitais numa perspetiva de processamento digital de sinais.

Numa primeira fase do trabalho, é efetuado uma abordagem teórica sobre os filtros digitais e o cálculo fraccionário. Estes conceitos teóricos são utilizados posteriormente para o desenvolvimento do presente projeto.

Numa segunda fase, é desenvolvida uma interface gráfica em ambiente MatLab, utilizando a ferramenta GUIDE. Esta interface gráfica tem como objetivo a implementação de filtros digitais fraccionários.

Na terceira fase deste projeto são implementados os filtros desenvolvidos experimentalmente através do ADSP-2181, onde será possível analisar e comparar os resultados experimentais com os resultados obtidos por simulação no MatLab.

Como quarta e última fase deste projeto é efetuado uma reflexão sobre todo o desenvolvimento da Tese e o que esta me proporcionou.

Com este relatório pretendo apresentar todo o esforço aplicado na realização deste trabalho, bem como alguns dos conhecimentos adquiridos ao longo do curso.

Palavras-Chave

Filtro digital, Processamento digital de sinal, Cálculo fraccionário, MatLab, GUIDE, ADSP-2181.

Abstract

The thesis is concerned with the application of fractional derivatives and integrals for the implementation of digital filters in a perspective of digital signal processing.

In the first phase of the work, it is given an introduction to digital filters and fractional calculus. These theoretical concepts will be used in the development of the present project.

In a second phase, it is developed a graphical user interface (GUI) in MATLAB using the GUIDE tool. This graphical interface aims to implement fractional digital filters.

In the third phase of this project are implemented the filters developed experimentally through the ADSP-2181, where it will be possible to analyse and compare the experimental results with the simulation results obtained from MatLab.

As a fourth and final phase of this project is made a reflection over the entire development of the thesis.

With this report I intend to present all the effort applied to the realization of this work, as well as some of the knowledge acquired along the course.

Keywords

Digital Filter, Digital Signal Processing, Fractional Calculus, MatLab, GUIDE, ADSP-2181.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

DSP – Processamento Digital de Sinal

FIR – Resposta Impulsional Finita

GUIDE – *GUI Design Environment*

IRR – Resposta Impulsional Infinita

1. INTRODUÇÃO

Nesta secção é apresentado o tema principal de estudo nesta tese, nomeadamente a análise e implementação de filtros digitais fraccionários.

1.1. CONTEXTUALIZAÇÃO

No âmbito da Unidade Curricular de Tese/Dissertação inserida no curso de Engenharia Eletrotécnica e de Computadores, foi dada a hipótese de abraçar entre a realização de um estágio ou de um projeto, tendo escolhido a segunda opção.

Esta Tese tem como objetivo o estudo e aplicação das derivadas e integrais de ordem não inteira na área do processamento digital de sinal, nomeadamente no projeto de filtros digitais.

1.2. OBJETIVOS

O principal objetivo deste projeto é o estudo e implementação de filtros digitais de ordem fracionária considerando os diversos tipos de estruturas existentes.

A realização desta dissertação foi dividida em várias fases:

- Análise e estudo de filtros digitais (FIR e IIR) e cálculo fraccionário;

- Implementação de filtros digitais fraccionários;
- Análise e estudo das diversas topologias de estruturas existentes para filtros digitais;
- Implementação de estruturas de filtros digitais fraccionários;
- Desenvolvimento de uma interface gráfica para auxílio na implementação de filtros digitais fraccionários e estruturas de filtros digitais fraccionários;
- Análise e teste em *hardware* DSP, analisando os resultados obtidos através da interface gráfica.

1.3. ORGANIZAÇÃO DO RELATÓRIO

O relatório encontra-se dividido em seis capítulos.

No capítulo 1, "Introdução", é dado a conhecer o contexto do projeto desenvolvido, assim como os objetivos estabelecidos para a sua execução e ainda a motivação que levou à sua realização.

No capítulo 2,"Fundamentos de Filtros Digitais e Cálculo Fraccionário", são apresentados os fundamentos de filtros digitais e do cálculo fraccionário. Este capítulo pretende estabelecer um enquadramento teórico e matemático que servirá de base para os restantes capítulos.

No capítulo 3, "Implementação de Filtros Digitais e Interface Gráfica", é descrito a implementação de filtros digitais FIR e IIR e o desenvolvimento de uma interface gráfica para o auxílio na implementação de filtros digitais fraccionários.

No capítulo 4, "Implementação em DSP", vão ser tecidos algumas considerações teóricas e importantes sobre a DSP utilizada e feita a comparação de algumas experiências teóricas e práticas dos filtros fraccionários.

No capítulo 5, "Conclusões", é apresentada uma reflexão sobre toda a Tese e o que esta me proporcionou ao longo de todo o tempo do seu desenvolvimento.

2. FUNDAMENTOS DE FILTROS DIGITAIS E CÁLCULO FRACCIONÁRIO

Neste capítulo é apresentado todo o estudo teórico necessário para o desenvolvimento e implementação de filtros digitais fraccionários. Estes conceitos teóricos serão a base para os restantes capítulos.

2.1. SINAIS

A vida baseia-se em sinais, que são medidos, processados, analisados, e fundamentam frequentemente as decisões que se formam. O som, a temperatura e a luz são exemplos de sinais que são utilizados no dia-a-dia.

Os ouvidos convertem o som em sinais elétricos, que chegam ao cérebro, e este é capaz de analisar algumas das suas propriedades, tais como amplitude, frequência e fase, determinar a direção em que se encontra a fonte de som, e reconhecê-lo, como música, fala, o ruído de um automóvel, etc.

Os nervos colocados nas partes expostas da pele sentem a temperatura e enviam para o cérebro sinais elétricos, que podem motivar decisões, tais como ligar um aquecedor, abrir uma janela, entre outros.

Os olhos focam as imagens na retina, que converte essas imagens em sinais elétricos e os enviam para o cérebro, que, pela análise da cor, da forma, da intensidade, e da luz é capaz de reconhecer objetos, medir distâncias ou detetar o movimento.

Um dos objetivos dos sinais é o transporte de informação. Como referido anteriormente, o sinal mais utilizado pelo ser humano é a voz humana. A voz humana é produzida através das cordas vocais para falar, cantar, gargalhar, chorar, gritar, entre outros, e a sua frequência ronda os 60 e 7000 Hz. O sinal é recebido através do sistema auditivo, sendo transmitido de igual modo para todos os seres humanos; porém o processamento do sinal recebido é inerente ao recetor.

Um sinal é uma função que representa uma quantidade ou variável física e contém informações sobre o comportamento ou a natureza do fenómeno.

Como exemplo de um sinal, pode considerar-se um circuito RC, em que o sinal pode representar a tensão no condensador ou a corrente que percorre uma resistência. No âmbito da matemática, o sinal, é representado por uma função de uma variável independente *t* (*t* representa o tempo), podendo ser indicado por $x(t)$. O processamento de sinal é responsável pela representação, transformação e manipulação dos sinais e da informação neles contidos.

2.2. SINAIS E SUA CLASSIFICAÇÃO

O conceito e a teoria de sinais e sistemas são necessários em quase todos os campos de engenharia e também em outras áreas científicas. Nesta secção, será apresentado uma breve descrição de sinais [1].

2.2.1. SINAIS DE TEMPO CONTÍNUO E DE TEMPO DISCRETO

Um sinal $x(t)$ é um sinal de tempo contínuo se *t* for uma variável contínua. No caso de o *t* se assumir como variável discreta o $x(t)$ seja definido em tempos discretos, então o $x(t)$ é um sinal de tempo discreto. Devido a ser definido em tempos discretos, um sinal discreto frequentemente é identificado por uma sequência de números, denotada como $\{X_n\}$ ou

$x[n]$ onde *n* é um valor inteiro. Na Figura 1 é possível visualizar um sinal contínuo $x(t)$ e um sinal discreto $x[n]$.

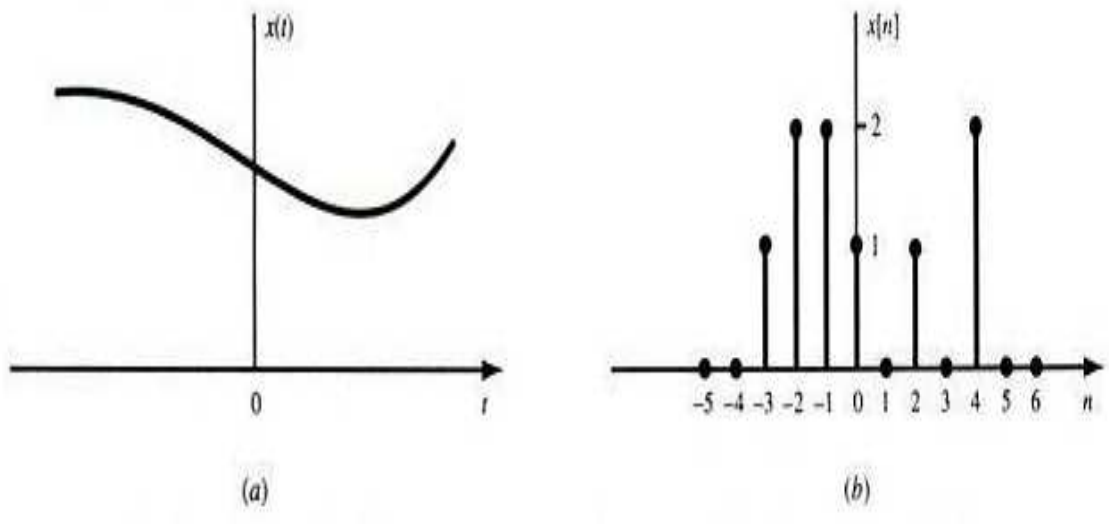

Figura 1:Representação gráfica de sinais: (a) tempo contínuo e (b) tempo discreto [1]

Um sinal discreto $x[n]$ pode representar um fenómeno no qual a variável independente é, inerentemente, discreta.

2.2.2. SINAIS ANALÓGICOS E DIGITAIS

Se um sinal contínuo $x(t)$ assumir qualquer valor no intervalo (*a*,*b*), onde *a* pode ser -∞ e *b* +∞, então o sinal de tempo contínuo $x(t)$ é denominado como sinal analógico. Se um sinal de tempo discreto $s[n]$ puder assumir apenas um número finito de valores distintos, então ele é chamado de sinal digital.

2.2.3. SINAIS REAIS E COMPLEXOS

Um sinal $x(t)$ é um sinal real se o seu valor for um número real e é um sinal complexo se o seu valor for um número complexo. Um sinal complexo $x(t)$, em geral, é uma função com a forma:

$$x(t) = x_1(t) + jx_2(t) \tag{2.1}$$

Neste caso $x_1(t)$ e $x_2(t)$ são sinais reais e $j = \sqrt{-1}$.

2.2.4. SINAIS PARES E ÍMPARES

Um sinal $x(t)$ ou $x[n]$é denominado par se:

$$x(-t) = x(t)$$
$$x[-n] = x[n] \tag{2.2}$$

Um sinal $x(t)$ ou $x[n]$é denominado ímpar se:

$$x(-t) = -x(t)$$
$$x[-n] = -x[n] \tag{2.3}$$

Na Figura 2 é possível visualizar exemplos de sinais pares e ímpares.

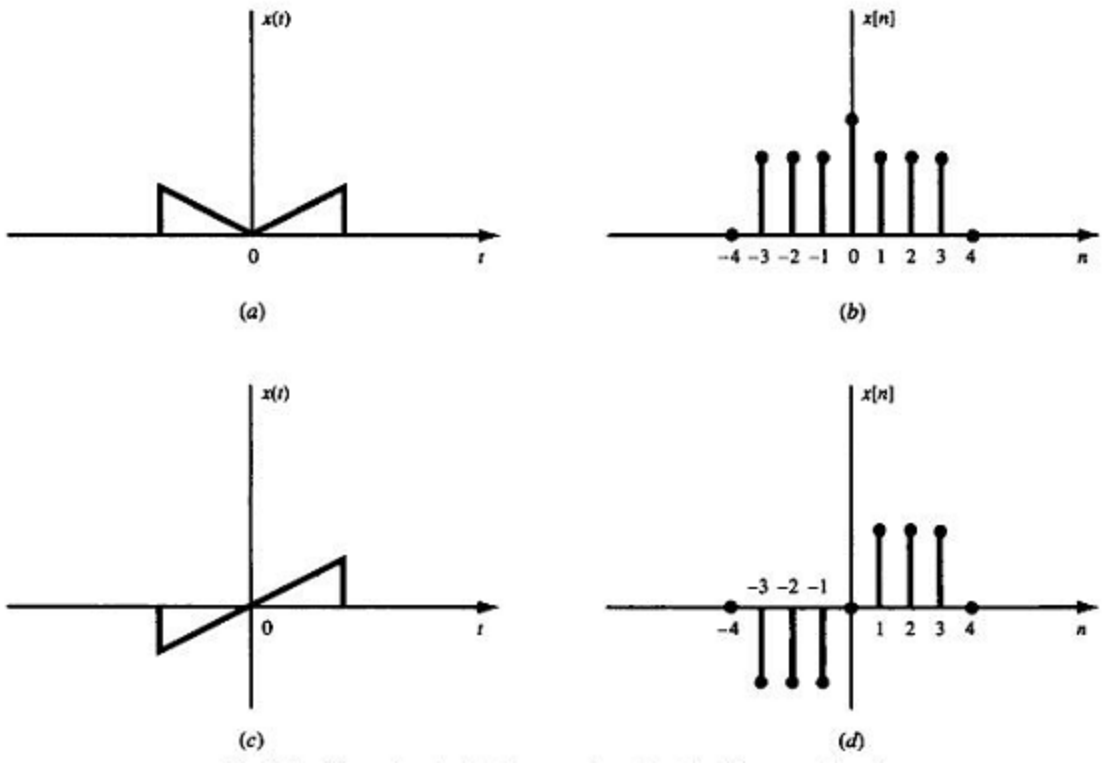

Figura 2: Exemplos de sinais pares (a e b) e sinais ímpares (c e d) [1]

2.3. SINAIS BÁSICOS DE TEMPO CONTÍNUO E DISCRETO

2.3.1. DEGRAU UNITÁRIO

A função degrau unitário $u(t)$, também conhecida como função de Heaviside unitária, é definida da seguinte forma:

$$u(t) = \begin{cases} 1, & t > 0 \\ 0, & t < 0 \end{cases} \tag{2.4}$$

A Figura 3 mostra a função Heaviside.

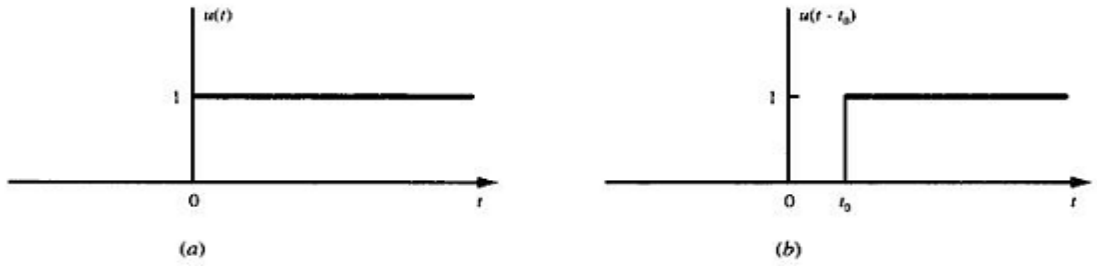

Figura 3: (a) Função degrau unitária, (b) função degrau unitária deslocada [1]

De acordo com o que foi referido anteriormente, é possível obter um sinal discreto. A sequência que se segue mostra a função em degrau unitário, $u[n]$, enquanto que na Figura 4 é possível visualizar a sequência em degrau unitário discreto.

$$u[n] = \begin{cases} 1\,, n > 0 \\ 0\,, n < 0 \end{cases} \tag{2.5}$$

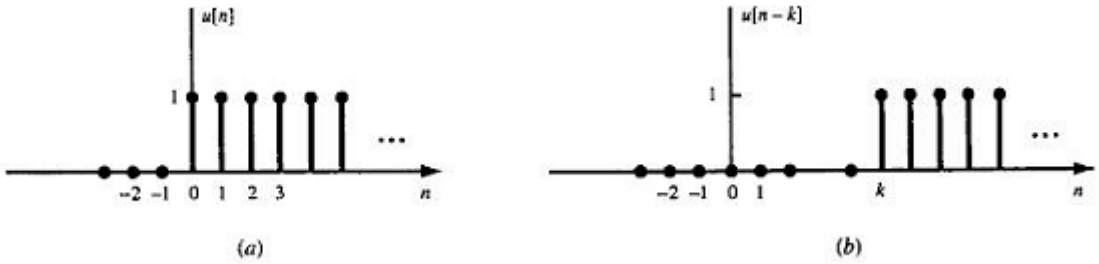

Figura 4: (a) Sequência degrau unitário, (b) Sequência degrau unitário deslocado [1]

2.3.2. FUNÇÃO IMPULSO UNITÁRIO

A função impulso unitário *δ(t)*, também conhecida como função delta de Dirac, tem um papel importante na análise de sistemas. Normalmente, *δ(t)* é definida como o limite de uma função convencional adequadamente escolhida, que tem uma área unitária dentro de um intervalo de tempo infinitesimal, como é possível visualizar na Figura 5, sendo definida como:

$$\delta(\mathrm{t}) = \begin{cases} 0, & t \neq 0 \\ \infty, & t = 0 \end{cases} \tag{2.6}$$

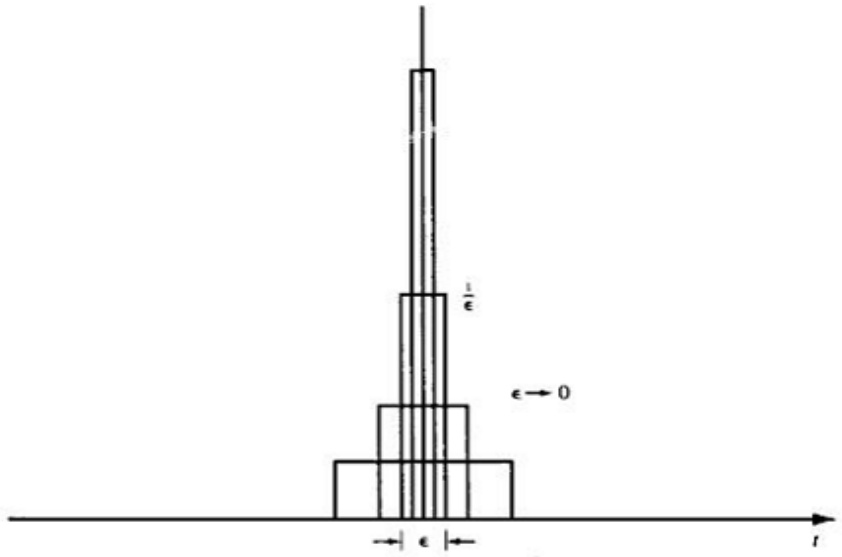

Figura 5: Impulso unitário [1]

Tal como o degrau unitário, o impulso unitário pode assumir um valor discreto. Na sequência o impulso unitário discreto, δ[n], é definido como:

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} \tag{2.7}$$

Na Figura 6 é possível visualizar um impulso unitário discreto.

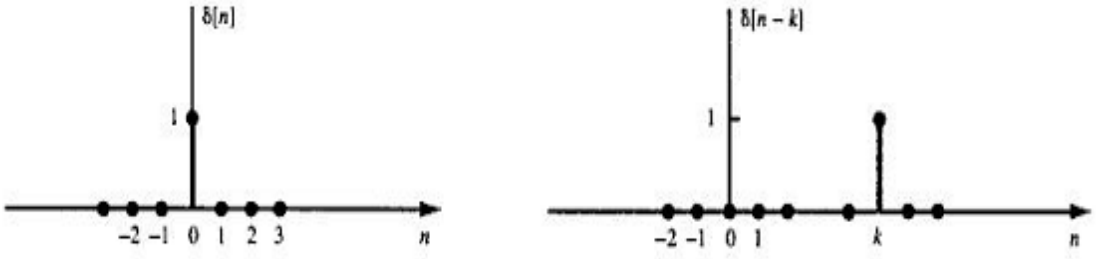

Figura 6: (a) Sequência impulso (amostra) unitário, (b) Sequência impulso unitário deslocado [1]

Nesta secção foi apresentada uma breve descrição dos sinais e suas classificações. Na secção seguinte são abordados os filtros digitais, assunto primordial no desenvolvimento do presente projeto.

2.4. FILTROS DIGITAIS

Na esfera tecnológica da atualidade, existe a necessidade de medir grandezas que variam com o tempo. A pressão arterial, a luminosidade de uma estrela, a amplitude de vibração de um terramoto, o fluxo de um líquido num tubo e a pressão acústica de sinais sonoros são alguns exemplos de grandezas que variam com o tempo. Para que estes dados possam ser tratados por computador existe a necessidade de converter o sinal num sinal digital, isto é, a função que varia no tempo que descreve uma grandeza deve ser amostrada. Assim, os dados tratados serão finitos e não infinitos, como seria caso o sinal não fosse amostrado. Para que os dados amostrados representem fielmente o sinal é preciso tratar os dados de forma adequada. Assim sendo, é fundamental respeitar as condições do teorema da amostragem. Para além de amostrar o sinal é necessário quantificá-lo. Os valores que as amostras podem ter estão relacionados com a resolução que é dada, assim se a amostra for representada por 8 bits estas podem ter 256 possíveis níveis (2^8=256) [2][3][4].

Os filtros digitais são filtros que atuam sobre sinais representados na forma digital, ou seja, são elementos que, exercendo um certo processamento, vão criar um outro sinal, no qual ocorreram modificações nas informações contidas no sinal inicial.

O uso de filtros digitais pode ter como fim várias aplicações sendo que algumas delas são a remoção de ruído, o destaque de determinadas informações ou a separação de informações de faixas de frequência diferentes.

Os filtros digitais têm várias vantagens comparativamente aos filtros analógicos, pois são programáveis e o seu funcionamento pode ser armazenado numa memória. Este facto faz com que não seja necessário reprojetar o circuito do filtro, o que não acontece quando se tratam de filtros analógicos, nos quais se pretende fazer alguma alteração. Uma outra vantagem prende-se com o facto de não ser necessária a construção de circuitos especiais para cada implementação. Ao contrário do que acontece com os filtros analógicos, os filtros digitais não sofrem qualquer alteração com as variações de temperatura. Este tipo de filtros é muito versátil, uma vez que é possível processar sinais das mais variadas formas, sendo igualmente possível a adaptação do filtro de acordo com o sinal de entrada.

2.4.1. CLASSIFICAÇÃO DOS FILTROS

Os filtros podem ser classificados em quatros categorias, nomeadamente passa-alto, passa-baixo, passa-banda e rejeição de banda. Cada filtro tem uma aplicação específica no processamento digital de sinal. Um dos objetivos dessas aplicações pode envolver a conceção de filtros digitais. De um modo geral, um filtro é projetado tendo como base as especificações das faixas passa banda, rejeição de banda e banda de transmissão da resposta de frequência. A faixa passa-banda é o intervalo de frequência em que ganho em amplitude do filtro atinge aproximadamente o valor de uma unidade. A faixa de rejeição de banda é definida como o intervalo de frequência sobre a qual a resposta de frequência do filtro é atenuada de forma a eliminar o sinal de entrada. A faixa banda de transmissão indica as frequências entre as faixas passa-banda e rejeição de banda.

As especificações do filtro passa-baixo, ilustrado na Figura 7, permite a passagem de baixas frequências e atenua as altas frequências. Como é apresentado na Figura 7, Ω_p e Ω_s, são as frequências de corte da faixa passa-banda e a frequência de corte da faixa de rejeição de banda, respectivamente; δ_p é o parâmetro para especificar o *ripple* da resposta em frequência da faixa passa-banda, enquanto o δ_s especifica o *ripple* da resposta em frequência da faixa rejeição de banda.

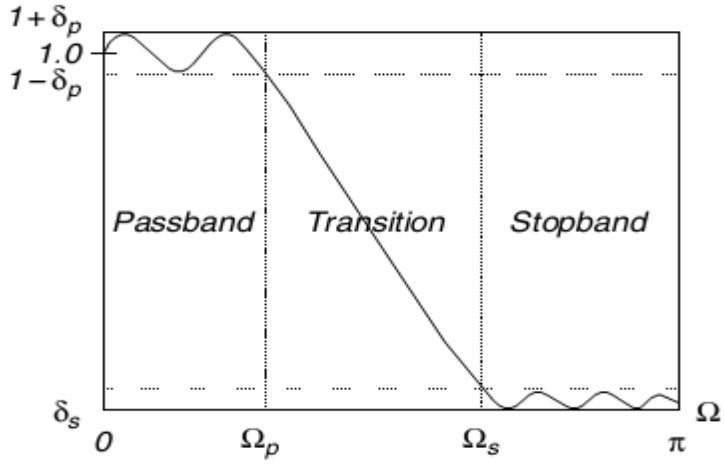

Figura 7: Resposta em frequência de um filtro passa-baixo [2]

O filtro passa-alto funciona de maneira inversa ao filtro passa-baixo, isto é, permite a passagem de altas frequências e atenua a amplitude das frequências menores que a frequência de corte. A resposta em frequência para o filtro passa-alto pode ser visualizada na Figura 8.

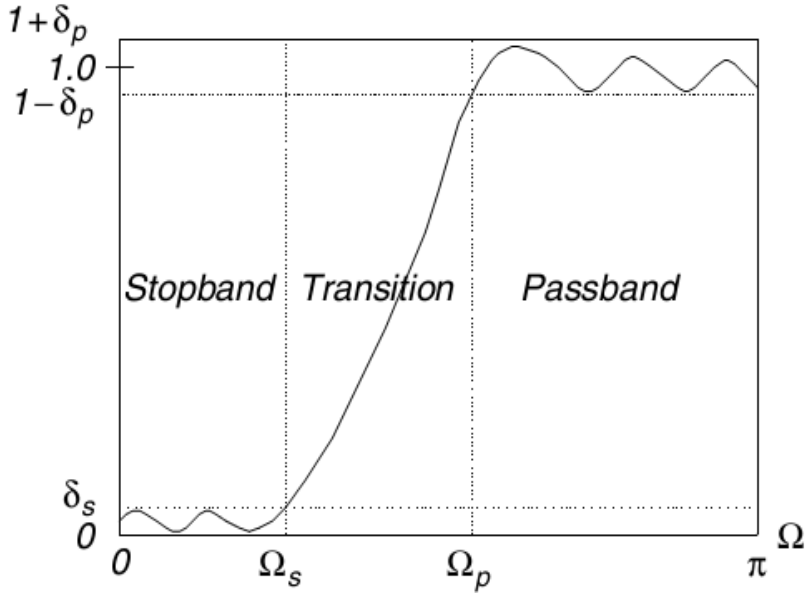

Figura 8: Resposta em frequência de um filtro passa-alto [2]

O filtro passa-banda, ilustrado na Figura 9, permite a passagem de frequências de uma determinada faixa, atenuando as baixas e altas frequências. Como é apresentado na Figura 9, Ω_{pL} e Ω_{sL} são os limites inferiores e superiores da passa-banda; Ω_{pH} e Ω_{sH} são a frequência limite superior da faixa passa-banda e o limite inferior da frequência da faixa de rejeição de banda, respectivamente; δ_p é o parâmetro para especificar o *ripple* da resposta em frequência da faixa passa-banda, e o δ_s especifica o *ripple* da resposta em frequência da faixa rejeição de banda.

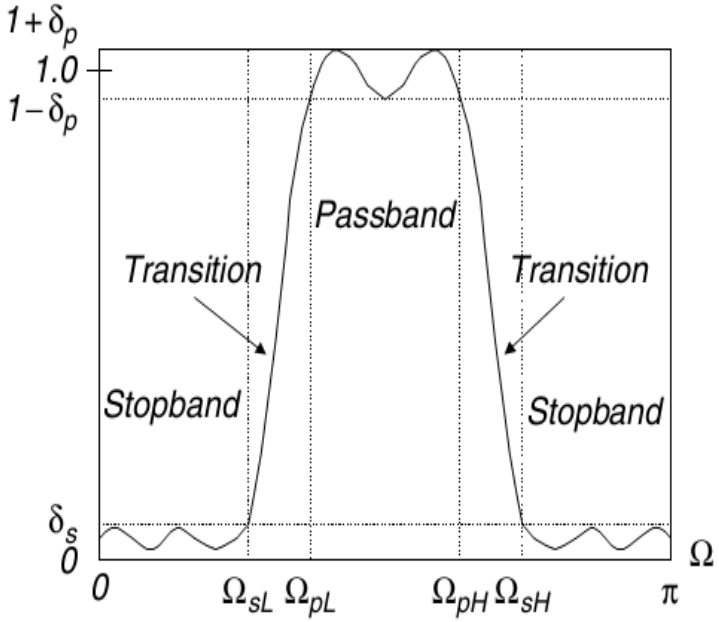

Figura 9: Resposta em frequência de um filtro passa-banda [2]

Por último, como pode ser visualizado na Figura 10, o filtro rejeição de banda atenua as frequências de uma determinada faixa, permitindo as baixas e altas frequências.

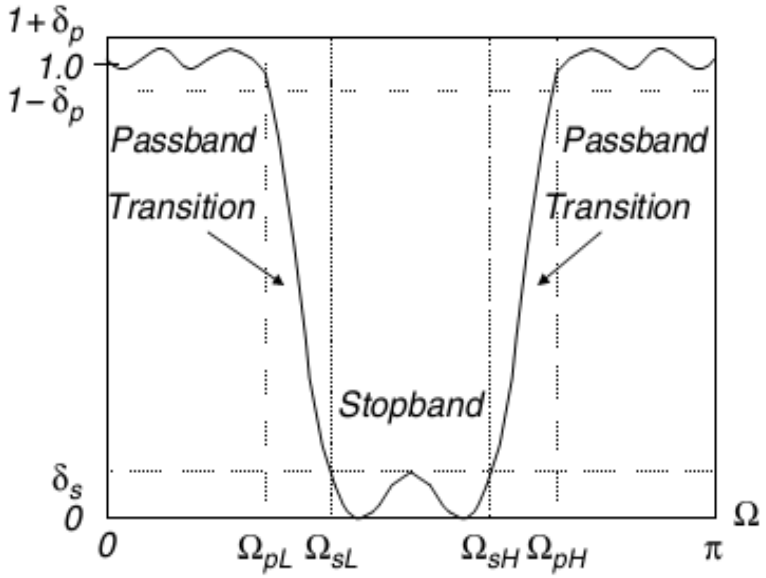

Figura 10: Resposta em frequência de um filtro rejeição de banda [2]

Todos os tipos de filtros digitais são implementados a partir de um sistema FIR ou IIR. Os sistemas FIR e IIR podem assumir diferentes topologias, tais como a direta, tipologia em cascata ou em paralelo [5][6].

2.4.2. TIPOS DE FILTROS

Seguidamente, serão abordados dois tipos de filtros, os de resposta impulsional infinita (IIR) e os de resposta impulsional finita (FIR). Uma das diferenças entre estes dois tipos de filtros prende-se com o facto de os filtros IIR, serem mais eficientes computacionalmente para uma determinada resposta em frequência, uma vez que têm um número menor de coeficientes [10][11].

Uma das características dos filtros FIR diz respeito à estabilidade existente, quando os filtros são implementados através de estruturas não recursivas. Por outro lado, se forem implementados recursivamente têm uma menor propagação de erros.

Nas implementações com aritmética finita é possível reduzir o ruído de quantificação inerente através de implementações não recursivas. Os filtros IIR são utilizados em aplicações onde as características lineares não constituem um motivo de preocupação, enquanto os filtros FIR são utilizados quando são necessárias as características lineares.

2.4.3. FILTROS FIR

A estrutura de um filtro FIR é bastante regular e, uma vez definidos os coeficientes, o filtro pode ser completamente especificado. Na Figura 11, pode observar-se uma estrutura simples de um filtro FIR, em que a passagem pelos componentes do filtro dá-se sempre da esquerda para a direita. Por isso, este filtro é também denominado por *feed-foward*.

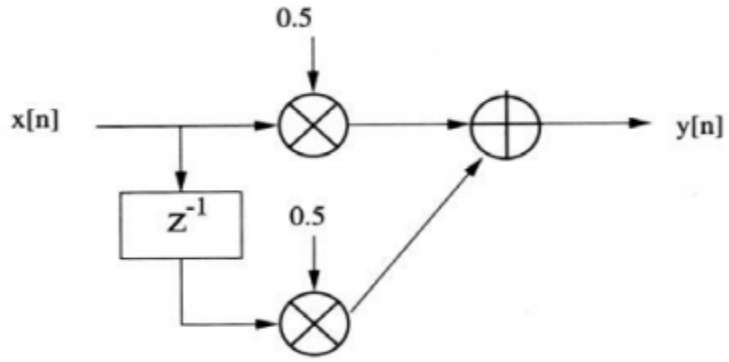

Figura 11: Estrutura simples de um filtro FIR

Os filtros FIR são expressos através da seguinte equação:

$$y[n] = \sum_{k=0}^{N} b_k \, x[n-k] \tag{2.8}$$

Nos filtros FIR a função de transferência $H(z)$ é:

$$H(z) = \sum_{k=0}^{N} b_k z^{-k} \tag{2.9}$$

A resposta deste tipo de filtro ao impulso $h[n]$ é dada por:

$$h[n] = f(x) = \begin{cases} b_n, & 0 \leq n \leq N \\ 0, & senão \end{cases} \tag{2.10}$$

A implementação do filtro FIR assenta na correspondente equação às diferenças:

$$y[n] = h[0]x[n] + h[1]x[n-1] + \cdots + h[N]x[n-N] \tag{2.11}$$

O filtro pode ser representado da forma indicada na Figura 12.

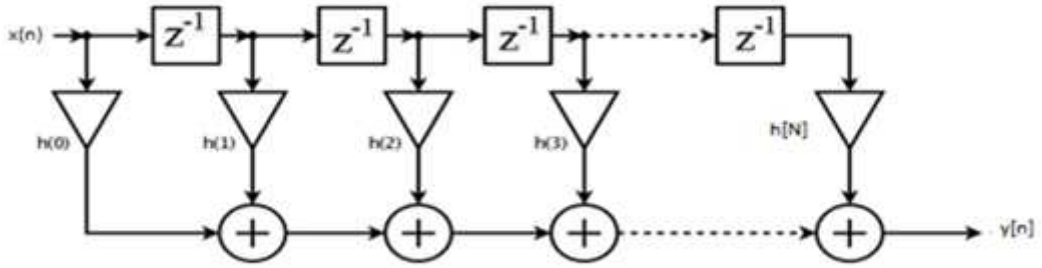

Figura 12: Estrutura básica de um filtro FIR

A representação de um filtro FIR pode ser feita apenas com os seus coeficientes. Por exemplo, se for considerado um filtro com coeficientes [1,-1], este pode ser representado da forma indicada na Figura 13.

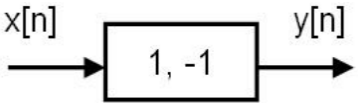

Figura 13: Exemplo de representação de um filtro FIR

Estes filtros têm três propriedades importantes, sendo elas:

- Causalidade;
- Linearidade;
- Invariância no tempo.

Um sistema é considerado causal quando não necessita de informações futuras para calcular a saída atual.

Para o sistema ser linear tem que obedecer ao princípio da sobreposição:

$$T\{ax_1[n] + bx_2[n]\} = aT\{x_1[n]\} + bT\{x_2[n]\} \quad (2.12)$$

Por último, para que o sistema seja invariante no tempo a sua saída terá de refletir qualquer deslocamento da entrada.

Apenas com a mudança dos seus coeficientes, os filtros FIR podem implementar diferentes funções. A função do filtro depende do seu comportamento no domínio das frequências.

2.4.4. FILTROS IIR

Nos filtros IIR a função de transferência $H(z)$ é definida por:

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{p} b_k z^{-k}}{1 + \sum_{k=1}^{q} a_k z^{-k}} \tag{2.13}$$

Onde se convencionou que a_0=1, o que é sempre possível fazer. A implementação do filtro IIR assenta na correspondente equação às diferenças:

$$y[n] = \sum_{k=0}^{p} b_i x[n-k] - \sum_{k=1}^{q} a_k y[n-k] \tag{2.14}$$

2.5. PROJETO DE FILTROS

O projeto de filtros digitais começa com as especificações do filtro, as quais podem incluir restrições de magnitude e/ou fase para a resposta em frequência, restrições de resposta do filtro à amostra unitária ou ao degrau, e especificações do tipo de filtro.

Depois de definir as especificações, o passo seguinte é encontrar um conjunto de filtros que produza um filtro aceitável.

2.5.1. PROJETO DE FILTROS FIR

A resposta em frequência de um filtro FIR causal de *N*-ésima ordem é dada por:

$$H\left(e^{j\omega}\right) = \sum_{k=0}^{N} h(n) e^{j\omega} \tag{2.15}$$

O projeto de um filtro FIR consiste em determinar os coeficientes de $h(n)$ que produzam uma resposta em frequência capaz de atender a um dado conjunto de especificações para o filtro. Os filtros FIR têm duas vantagens importantes em relação dos filtros IIR. Primeiro como já foi mencionado, é garantido que os filtros FIR são estáveis, mesmo depois da quantização dos coeficientes dos filtros. Segundo, podem ser facilmente condicionados a

terem fase linear. Como, em geral, os filtros FIR são projetados para terem fase linear, de seguida será apresentado o projeto de filtros digitais FIR utilizando janelas.

2.5.1.1. JANELA RECTÂNGULAR

A janela rectângular é a mais simples; no entanto, é a que oferece um pior desempenho no que diz respeito à atenuação da banda de corte. A seguinte expressão define esta janela:

$$w[n] = \begin{cases} 1, & 0 \leq n \leq M \\ 0, & senão \end{cases} \tag{2.16}$$

A resposta em frequência é obtida através da seguinte expressão:

$$w(e^{jw}) = \sum_{n=0}^{N} e^{-jwn} = \frac{1 - e^{-jw(N+1)}}{1 - e^{-jw}} = e^{-jw\frac{N}{2}} \frac{\sin\left(\frac{w(N+1)}{2}\right)}{\sin\left(\frac{w}{2}\right)} \tag{2.17}$$

A magnitude da função $\sin\left[\frac{w(N+1)/2}{\sin\left(\frac{w}{2}\right)}\right]$ é mostrada na Figura 14 para o caso de N=7. À medida que *N* aumenta, a largura do lóbulo principal diminui.

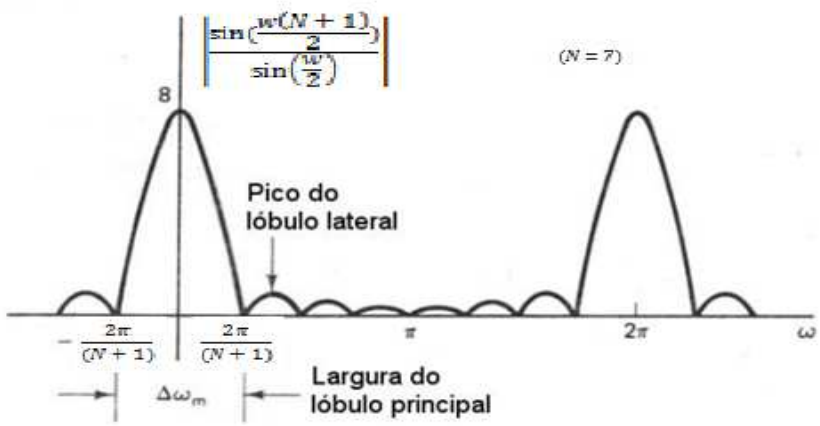

Figura 14: Magnitude da transformada de Fourier de uma janela rectângular

A largura do lóbulo central para uma janela rectângular é:

$$\Delta w_n = \frac{4\pi}{(N+1)} \tag{2.18}$$

Sendo que o primeiro zero ocorre quando:

$$\sin\left[\frac{w(N+1)}{2}\right] = 0 \Rightarrow \frac{w(N+1)}{2} = \pi \tag{2.19}$$

Assim, a largura do lóbulo central é o dobro desse valor:

$$2w = \frac{4\pi}{(N+1)} \tag{2.20}$$

É, também, possível observar-se que a magnitude do primeiro lóbulo lateral é aproximadamente:

$$w = \frac{3\pi}{(N+1)} \tag{2.21}$$

e é dada por:

$$\frac{\sin\left[\frac{w(N+1)}{2}\right]}{\sin\left(\frac{w}{2}\right)} = \frac{\sin\left(\frac{3\pi}{2}\right)}{\sin\left[\frac{3\pi}{2(N+1)}\right]} \cong \frac{2(N+1)}{3\pi} \tag{2.22}$$

À medida que *N* aumenta, a largura de cada lóbulo lateral diminui, mas a área sobre cada um permanece constante. Assim, as amplitudes relativas dos picos laterais vão permanecer constantes e a atenuação da banda de passagem permanece em cerca de 21 dB, o que significa que as ondulações vão sofrer um pico perto das bordas das bandas. Este acontecimento é conhecido como o fenómeno de Gibbs. Este fenómeno pode ocorrer por causa da transição de 0 para 1 e de 1 para 0 da janela rectângular.

2.5.1.2. JANELA HAMMING

Para o cálculo da janela de Hamming utiliza-se a seguinte expressão:

$$w[n] = \begin{cases} 0{,}54 - 0{,}46\cos\left(\frac{2\pi n}{N}\right), & 0 \le n \le N \\ 0, & sen\tilde{a}o \end{cases} \tag{2.23}$$

2.5.1.3. JANELA BLACKMAN

Esta janela é similar à anterior, mas tem um segundo harmónico, o que faz com que esta se aproxime de zero com mais suavidade. A expressão a calcular para se obter esta janela é a seguinte:

$$w[n] = \begin{cases} 0{,}42 - 0{,}5\cos\left(\frac{2\pi n}{N}\right) + 0.08\cos\left(\frac{4\pi n}{N}\right), & 0 \le n \le N \\ 0, & senão \end{cases} \quad (2.24)$$

2.5.1.4. JANELA KAISER

Esta é considerada a melhor janela, pois oferece um lóbulo central largo para a dada atenuação de corte, o que implica uma mais brusca banda de transição. A função foi definida por Kaiser e é dada por:

$$w[n] = \begin{cases} \frac{I_0\left[\beta\sqrt{1-\left(1-\frac{2n}{N}\right)^2}\right]}{I_0[\beta]}, & 0 \le n \le N \\ 0, & senão \end{cases} \quad (2.25)$$

I_0 é a função de Bessel modificada de ordem zero:

$$I_0(x) = 1 + \sum_{n=1}^{\infty}\left[\frac{\left(\frac{x}{2}\right)^n}{n!}\right]^2 \quad (2.26)$$

Na expressão de $w[n]$ existem dois parâmetros:

1. O comprimento de *N*;
2. O parâmetro β.

Variando β e *N* é possível ajustar a amplitude dos lóbulos laterais. Kaiser encontrou duas fórmulas que permitem calcular N e β, de modo a atender às especificações do filtro. Assim, dado que δ_1 é fixo, a frequência de corte *w* da banda de passagem do filtro passa-baixo é maior que a frequência, tal que:

$$\left|H\left(e^{jw}\right)\right| \ge 1 - \delta_1 \quad (2.27)$$

A frequência da banda de corte tem tolerância de δ_2, satisfazendo:

$$\left|H\left(e^{jw}\right)\right| \ge \delta_2 \quad (2.28)$$

A largura da banda de transição é:

$$\Delta w = w_s - w_p \quad (2.29)$$

Sabendo que:

$$A = -20 \log \delta_2 \tag{2.30}$$

Considerando que $\delta_1 = \delta_2$, Kaiser mostrou que:

$$\beta = \begin{cases} 0{,}112(A - 8{,}7) & , A > 50 \\ 0{,}5842(A - 21)^{0{,}4} + 0{,}07886(A - 21) & , 2 \leq A \leq 50 \\ 0 & , senão \end{cases} \tag{2.31}$$

Além disso, dados Δw e *A*, *N* é aproximadamente:

$$N = \frac{A - 8}{2{,}285 \Delta w} \tag{2.32}$$

Para se projetar um filtro usando a janela de Kaiser é necessário realizar as seguintes etapas:

1. Estabelecer as especificações w_p, w_s e δ;
2. Estabelecer a frequência de corte w_c do filtro passa-baixo ideal ao qual se aplicará a janela;
3. Calcular $A = 20 \log \delta$ e $\Delta w = w_s - w_p$ e usar as fórmulas de Kaiser para encontrar os valores de *N* e β;
4. Encontrar a resposta ao impulso do filtro.

2.5.2. PROJETO DE FILTROS IIR

Existem duas abordagens gerais utilizadas para conceber filtros digitais IIR. A abordagem mais comum consiste em projetar um filtro IIR analógico e mapeá-lo em um filtro digital equivalente. Neste sentido, é prudente considerar técnicas ideais para mapear esses filtros no domínio do tempo discreto. Além disso, porque há procedimentos que facilitam o projeto de filtros analógicos, esta abordagem torna o projeto do filtro IIR relativamente simples [16][17][23].

A segunda abordagem para conceber filtros digitais IIR traduz-se na utilização de um procedimento de conceção algorítmica, que geralmente exige a utilização de um computador para resolver um conjunto de equações lineares ou não lineares. Estes métodos

podem ser utilizados para conceber filtros digitais com características de resposta em frequência arbitrária para quais não existe protótipo filtro analógico ou a conceção de filtros quanto a outros tipos de restrições impostas sobre a conceção.

Neste ponto, serão apresentadas as diferentes aproximações que auxiliam o projeto de filtros digitais IIR.

2.5.2.1. FILTRO DE BUTTERWORTH

A principal característica de um filtro IIR implementado através da aproximação Butterworth é que a resposta em magnitude é plana na banda passante e na banda de corte.

O filtro de Butterworth tem como resposta em frequência a seguinte equação:

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \left(\frac{\Omega}{\Omega_c}\right)^{2N}} \tag{2.33}$$

Onde os parâmetros *N* é a ordem do filtro e Ω_c é a frequência de corte. Na Figura 15 é possível visualizar a resposta em frequência de um filtro Butterworth.

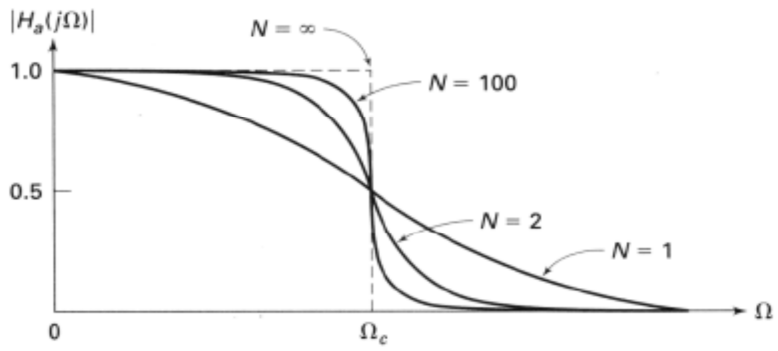

Figura 15: Resposta em frequência de um filtro Butterworth

Através da analise da Figura 15 anterior é possível retirar que:

- Para Ω=0, $|H_a(j0)|^2 = 1$ para todo a ordem do filtro;
- Em $\Omega = \Omega_c, |H_a(j\Omega_c)|^2 = 0.5,$ para todo o valor *N*, o que vai significar uma atenuação de 3 dB na frequência de corte;
- $|H_a(j\Omega)|^2$ é uma função decrescente ao longo da frequência;

- $|H_a(j\Omega)|^2$ aproxima-se de um filtro passa-baixo quando a ordem tende para o infinito;

- $|H_a(j\Omega)|^2$ assume o valor máximo para $\Omega = 0$.

2.5.2.2. FILTRO DE CHEBYCHEV TIPO I

Como foi anteriormente visto, a aproximação de Butterworth de ordem *N* conduz ao melhor polinómio em Ω, que maximiza o aplanamento da característica de atenuação da frequência $\Omega = 0$. O erro de atenuação relativamente ao filtro passa-baixo é nulo em $\Omega = 0$, sendo progressivamente crescente na banda de passagem. A aproximação de Chebychev leva a um polinómio em Ω que minimiza o erro da banda de passagem segundo um critério de erro oscilante entre um certo número de valores máximos e mínimos. A sua resposta em frequência é dada por :

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \varepsilon^2 T_N^2\left(\frac{\Omega}{\Omega_c}\right)} \tag{2.34}$$

Onde a ordem do filtro é *N* e ε é o fator de ondulação da banda passante.

2.5.2.3. FILTRO DE CHEBYCHEV TIPO II

Os filtros de Butterworth e de Chebychev são filtros polinomiais, isto é, só têm polos na sua função transferência. Os zeros de transmissão só acontecem para Ω infinito. A introdução de zeros de transmissão numa frequência finita conduz a uma variação da atenuação muito mais abrupta do que aquilo que se consegue só utilizando pólos. A inversão da característica de atenuação de um filtro de Chebychev, dado que este tem igual ondulação da atenuação na banda de passagem, conduzirá a uma atenuação com igual ondulação na banda de atenuação e originará atenuações infinitas nalgumas frequências, uma vez que o filtro de Chebychev tem atenuação nula em algumas frequências.

À semelhança dos filtros Butterworth, os filtros de Chebychev inversos têm a propriedade de aplanamento máximo na banda de passagem; contudo os últimos são mais seletivos na banda de transição.

A função $|H_a(j\Omega)|^2$do filtro de Chebychev inverso está relacionada com a função do filtro de Chebychev por:

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \left[\varepsilon^2 T_N^2 \left(\left[\frac{\Omega}{\Omega_c}^{-1}\right]\right)^{-1}\right]} \tag{2.35}$$

Esta transformação garante que os filtros inversos de Chebychev têm aplanamento máximo na origem, tal como os filtros de Butterworth, e igual ondulação na banda de atenuação, conseguida através da introdução dos zeros de transmissão provocada pela inversão da característica. Assim, para frequências pouco maiores do que $\Omega = 1$, a selectividade dos filtros inversos de Chebychev é maior do que a dos filtros de Chebychev e Butterworth devido ao efeito dos zeros de transmissão.

2.5.2.4. FILTRO ELÍPTICO

Os filtros elípticos têm a propriedade de ter igual ondulação na banda de passagem e na banda de atenuação. De certo modo, os filtros elípticos são semelhantes aos filtros de Chebychev na banda de passagem e aos filtros inversos de Chebychev na banda de atenuação. A determinação destes filtros é muito complexa e baseia-se na utilização de integrais elípticos.

A sua função de transferência é dada pela seguinte equação, onde U_N é a função jacobiana elíptica:

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \varepsilon^2 U_N^2(\Omega)} \tag{2.36}$$

2.6. FUNDAMENTOS DO CÁLCULO FRACIONÁRIO

O cálculo fracionário é uma ferramenta precisa para modelar, por exemplo, a descrição de fenómenos naturais. Um método bastante comum para se utilizar esta ferramenta é substituindo uma derivada de ordem inteira de uma equação diferencial parcial, que descreve um determinado fenómeno, por uma derivada de ordem não-inteira. Várias descobertas importantes e generalizações incidiram sobre esta técnica, em diversas áreas de conhecimento, tais como: mecânica dos fluidos, redes elétricas, entre outras [20][26][27].

Sendo assim, pode dizer-se que o cálculo fracionário representa a teoria das derivadas e integrais com uma ordem arbitrária, a qual unifica e generaliza as noções de integração e

diferenciação de ordem inteira. De facto, a designação correta desta teoria seria a integração e diferenciação a uma ordem arbitrária.

2.6.1. DEFINIÇÕES BÁSICAS DE DERIVADAS FRACCIONÁRIAS

Nesta subsecção são apresentados várias definições de derivadas e integrais fracionárias, assim como algumas das suas principais propriedades. Estas definições originam de uma generalização das noções de derivadas e integrais de ordem inteira provenientes do cálculo diferencial clássico [20][27].

$$\dots \int_a^t d\tau_2 \int_a^{\tau_2} f(\tau_1) d\tau_1 \,, \int_a^t f(\tau_1) d\tau_1 \,, f(t), \frac{df(t)}{dt}, \frac{d^2 f(t)}{dt^2} \dots \tag{2.37}$$

O operador fundamental, ${}_aD_t^\alpha$, que generaliza as derivadas e integrais a uma ordem α fracionária, é dado por:

$${}_aD_t^\alpha = \begin{cases} d^\alpha / dt^\alpha, & Re(\alpha) > 0 \\ 1, & Re(\alpha) = 0 \\ \int_a^t (f\tau)^{-\alpha}, & Re(\alpha) < 0 \end{cases} \tag{2.38}$$

Onde α pode assumir um valor arbitrário: real, racional, irracional ou mesmo complexo. Neste projeto apenas são considerados os valores de α reais.

De seguida, são apresentadas as definições mais frequentemente utilizadas e consideradas como mais importantes, quer sob o ponto de vista de desenvolvimento matemático quer da sua aplicação na resolução de problemas práticos e, em particular na área de controlo automático de sistemas.

2.6.1.1. DEFINIÇÃO DE GRÜNWALD –LETNIKOV

A definição de Grünwald – Letnikov é considerada como sendo a mais elementar, dado que impõe menos restrições nas funções em que é aplicada e unifica num único operador as noções de derivada e integral.

Na equação seguinte é possível visualizar a definição de Grünwald – Letnikov para a derivada e integral fracionário de ordem α (com $\alpha \in \Re$).

$$ {}_aD_t^{\alpha} f(t) = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{k=0}^{\left[\frac{t-a}{h}\right]} (-1)^k \binom{\alpha}{k} f(t - kh) \tag{2.39} $$

em que:

$$ \binom{\alpha}{k} = \frac{\alpha(\alpha-1)(\alpha-2)\ldots(\alpha-\mathrm{k}-1)}{k!} = \frac{\Gamma(\alpha+1)}{\Gamma(k+1)\Gamma(\alpha-\mathrm{k}+1)} \tag{2.40} $$

Onde $\Gamma(\mathrm{x})$ é a função Gama, que generaliza a função fatorial para valores não inteiros do argumento.

A definição Grünwald – Letnikov é geralmente utilizada para obter a solução numérica de sistemas definidos por derivadas fracionárias [27].

2.6.1.2. DEFINIÇÃO DE RIEMANN-LIOUVILLE

A definição Riemann-Liouville resulta da generalização da fórmula do integral de Cauchy, que reduz o cálculo da primitiva, correspondente à integração de multiplicidade *n* de uma função *f*(*t*), a uma integração simples do tipo convolução. A definição de Riemann-Liouville para a derivada fracionária da função *f*(t) é ($\alpha > 0$):

$$ {}_aD_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha+1-\mathrm{n}}} \mathrm{d}\tau, \quad \mathrm{n\text{-}1} < \alpha < \mathrm{n} \tag{2.41} $$

Esta definição desempenha um papel muito importante no desenvolvimento da teoria das derivadas e integrais fracionárias e na sua aplicação na área da matemática pura [27].

2.6.2. TRANSFORMADAS DE LAPLACE DE DERIVADAS FRACCIONÁRIAS

A transformada de Laplace (*L)*, é um método adequado e muito utilizado para a análise e projeto de sistemas de ordem inteira, assim como para os sistemas fracionários. A sua capacidade de transformar operações complexas como a integração e diferenciação em operações algébricas na variável complexa *s* revela-se extremamente útil no projeto de sistemas de controlo. Do mesmo modo, que tem a capacidade para reduzir a complexidade associada à análise no domínio dos tempos das derivadas e integrais faccionárias é também de enorme importância para o estudo de sistemas fracionários [20][26][27].

A transformada de Laplace da derivada de ordem inteira *n* de uma função *f*(*t*) é dada por:

$$L\{f^{(n)}(t)\} = s^n F(s) - \sum_{k=0}^{n-1} s^{n-k-1} f^{(k)}(0) = s^n F(s) - \sum_{k=0}^{n-1} s^k f^{(n-k-1)}(0) \quad (2.42)$$

2.6.2.1. INTEGRAL FRACIONÁRIO

O integral fracionário de Riemann-Liouville e de Grünwald–Letnikov de ordem $\alpha>0$ pode ser expresso como a convolução das funções $\boldsymbol{g(t) = t^{\alpha-1}}$ e $\boldsymbol{f(t)}$:

$${}_0D_t^{\alpha} f(t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(t) d\tau = \frac{t^{\alpha-1}}{\Gamma(\alpha)} * f(t) \quad (2.43)$$

Sabendo que:

$$L\{t^{\alpha-1}\} = \Gamma(\alpha) s^{-\alpha} \quad (2.44)$$

Utilizando a propriedade da transformada de Laplace na equação (2.43), obtém-se a transformada de Laplace do integral de Riemann-Liouville e de Grünwald – Letnikov:

$$L\{\ {}_0D_t^{\alpha} f(t)\} = F(s) s^{-\alpha} \quad (2.45)$$

2.6.2.2. DERIVADA FRACIONÁRIA

A expressão da derivada fracionária de Riemann-Liouville pode ser dada na seguinte forma, considerando que $\alpha>0$:

$${}_0D_t^{\alpha} f(t) = {}_0D_t^{n}({}_0D_t^{-(n-\alpha)} f(t)) \quad (2.46)$$

Através da expressão anterior é possível obter a expressão final para a transformada de Laplace da derivada fracionária de Riemann-Liouville de ordem $\alpha>0$, dada por:

$$L\{{}_0^R D_t^{\alpha} f(t)\} = s^{\alpha} F(s) - \sum_{k=0}^{n-1} s^k [({}_0D_t^{\alpha-k-1}]_{k=0} \quad (2.47)$$

Enquanto que a transformada de Laplace da derivada fracionária de Grünwald – Letnikov é dada por:

$$L\{{}_0^G D_t^{\alpha} f(t)\} = s^{\alpha} F(s), \ \ 0 < \alpha < 1 \quad (2.48)$$

Caso as condições sejam nulas, então a transformada de Laplace das derivadas e integrais fracionários é dada unicamente pela seguinte expressão:

$$L\{{}_0^G D_t^\alpha f(t)\} = s^\alpha F(s), \quad \alpha \in \mathcal{R} \tag{2.49}$$

Importa salientar que a expressão anterior (2.49) é uma generalização direta do resultado do caso de ordem inteira com a multiplicação da transformada do sinal pelo operador $s^\alpha (\alpha \in \mathcal{R})$. Neste sentido, os métodos clássicos de análise e projeto de sistemas lineares baseados no plano complexo *s* poderão ser facilmente adaptáveis para o caso de sistemas lineares fracionários [20][27].

2.6.3. TRANSFORMADAS DE FOURIER DE DERIVADAS FRACCIONÁRIAS

A transformada de Fourier, normalmente representada pelo símbolo *F*, é uma ferramenta muito importante e eficaz na análise da resposta em frequência de sistemas.

$$H(w) = F\{h(t)\} = \int_{-\infty}^{\infty} h(t) e^{-jwt} dt \tag{2.50}$$

Na equação anterior, é possível visualizar a transformada de Fourier de uma função *h(t)* integrável no intervalo $(-\infty, \infty)$.

A transformada de Fourier da derivada a uma ordem inteira *n* de uma função *h*(*t*), considerando que tanto *h*(*t*) como as suas derivadas $h'(t), h''(t), \dots, h^{(n-1)}(t)$ se anulam quando $t \to \pm\infty$, é dada pela expressão seguinte [27]:

$$F\{h^n(t)\} = (jw)^n F(w) \tag{2.51}$$

2.6.3.1. INTEGRAL FRACIONÁRIO

Do mesmo modo que a transformada de Laplace, o integral fracionário de Riemann-Liouville é dado pela seguinte equação:

$$F\{{}_{-\infty}^R D_t^{-\alpha} g(t)\} = (jw)^{-\alpha} G(w) \tag{2.52}$$

Onde $G(w) = F\{g(t)\}$. A expressão (2.52) fornece também as transformadas de Fourier dos integrais fracionários de Grünwald – Letnikov ${}_{-\infty}^G D_t^\alpha g(t)$, dado que neste caso coincidem com o integral fracionário de Riemann-Liouville ${}_{-\infty}^R D_t^{-\alpha} g(t)$ [27].

2.6.3.1. DERIVADA FRACIONÁRIA

As definições de Riemann-Liouville e de Grünwald – Letnikov são dadas pela expressão seguinte:

$$F\{D^{\alpha}g(t)\} = (jw)^{\alpha}G(w) \tag{2.53}$$

A resposta em frequência de um sistema fracionário pode ser obtida substituindo o operador $s^{\alpha}(\alpha \in \mathcal{R})$ por $(jw)^{\alpha}(\alpha \in \mathcal{R})$ na sua função de transferência. Desta maneira, os métodos clássicos de análise e projeto de sistemas dinâmicos no domínio das frequências continuam válidos para a análise e projeto de sistemas de controlo fracionário.

2.7. DISCRETIZAÇÃO DE OPERADORES FRACCIONÁRIOS

Antes de uma análise dos filtros de ordem fracionária, é necessário efetuar a discretização de integradores e diferenciadores de ordem fracionária [28][29][30].

O método mais frequentemente utilizado para a obtenção de equivalentes discretos fracionários $s^{\alpha}(\alpha \in \mathcal{R})$ consiste na utilização de uma função geradora do tipo $s = w(z^{-1})$ [31][32][33]. Logo, o equivalente discreto fracionário do operador s^{α} é obtido elevando a correspondente função geradora a uma ordem fracionária α, ou seja, realizando a operação:

$$s^{\alpha} = [w(z^{-1})]^{\alpha} \tag{2.54}$$

Se for considerada uma função contínua $G(s)$, a função de transferência discreta $G(z)$ é obtida através da substituição $s^{\alpha} = H^{\alpha}(z)$, como é possível visualizar na seguinte equação:

$$G(z) = G(s)|_{s^{\alpha}=H^{\alpha}(z)} \tag{2.55}$$

Onde $H^{\alpha}(z)$ representa o equivalente fracionário de ordem α do operador s^{α} expresso em função variável complexa z ou do operador em atraso z^{-1}, dado pela expressão seguinte:

$$H^{\alpha}(z) = [w(z^{-1})]^{\alpha} \tag{2.56}$$

A expressão anterior (2.56), é conhecida como o método em malha aberta de conversão de analógico para digital ou, mais simplesmente, método de conversão de $s \rightarrow z$ [34][35]. Os métodos de conversão de analógico para digital mais utilizados são os seguintes:

- Operador de Euler;
- Operador de Tustin;
- Operador de Al-Alaoui.

Na Tabela 1 é possível visualizar a aproximação de $H^{\alpha}(z)$ para cada um dos métodos referidos.

Tabela 1: Métodos de discretização de $S \rightarrow Z$

Método	Aproximação, $H^{\alpha}\left(z^{-1}\right)$
Euler Grünwald-Letnikov	$s^{\alpha} \approx \left(\frac{1-z^{-1}}{T}\right)^{\alpha}$
Tustin	$s^{\alpha} \approx \left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right)^{\alpha}$
Al-Alaoui	$s^{\alpha} \approx \left(\frac{8}{7T}\frac{1-z^{-1}}{1+z^{-1}/7}\right)^{\alpha}$

2.8. FILTRO FIR FRACIONÁRIO

Após esta breve introdução aos operadores utilizados para a discretização de integradores e diferenciadores de ordem fracionária, pretende-se agora determinar a resposta impulsional $h^{\alpha}(k)$ de cada um deles. Para a sua obtenção considera-se que $h^{\alpha}(k) = 0$ para $k < 0$, isto é, que o sistema é causal. Considerando que as respostas impulsionais $h^{\alpha}(k)$ são obtidas expandindo em série de potências (*i.e.* efetuando a série de Taylor em torno de $x=z^{-1}=0$) para cada uma das funções geradas $H_E^{\alpha}(z^{-1})$, $H_T^{\alpha}(z^{-1})$ e $H_A^{\alpha}(z^{-1})$ em que os índices *E,T* e *A* representam, respectivamente, os operadores Euler, Tustin e Al-Alaoui [30][36].

Assim, efetuando, uma expansão de uma série de potências (PSE) sobre o operador de Euler $H_E^{\alpha}(z^{-1})$, obtém-se:

$$H_E^{\alpha}(z^{-1}) = \left[\frac{1}{T}(1-z^{-1})\right]^{\alpha} = \left(\frac{1}{T}\right)^{\alpha} \sum_{k=0}^{\infty} (-1)^k \binom{\alpha}{k} z^{-k} = \sum_{k=0}^{\infty} h_E^{\alpha}(k) z^{-k} \qquad (2.57)$$

Em que a sua resposta impulsional, $h_E^{\alpha}(k)$, é obtida a partir da equação anterior, sendo dada através da expressão seguinte:

$$h_E^{\alpha}(k) = \begin{cases} \left(\frac{1}{T}\right)^{\alpha} (-1)^k \binom{\alpha}{k}, & k \geq 0 \\ 0, & k < 0 \end{cases} \tag{2.58}$$

Procedendo da mesma forma para os operadores Tustin e de Al-Alaoui, respectivamente $H_T^{\alpha}(z^{-1})$ e $H_A^{\alpha}(z^{-1})$, vem que:

$$\begin{aligned} H_T^{\alpha}(z^{-1}) &= \left[\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right]^{\alpha} = \left(\frac{2}{T}\right)^{\alpha} \sum_{k=0}^{\infty} \left[\sum_{j=0}^{k} (-1)^j \binom{\alpha}{j}\binom{-\alpha}{k-j}\right] (z)^{-k} \\ &= \sum_{k=0}^{\infty} h_T^{\alpha}(k) z^{-k} \end{aligned} \tag{2.59}$$

$$\begin{aligned} H_A^{\alpha}(z^{-1}) &= \left[\frac{8}{7T}\left(\frac{1-z^{-1}}{1+z^{-1}/7}\right)\right]^{\alpha} \\ &= \left(\frac{8}{7T}\right)^{\alpha} \sum_{k=0}^{\infty} \left[\sum_{j=0}^{k} (-1)^j \left(\frac{1}{7}\right)^{k-j} \binom{\alpha}{j}\binom{-\alpha}{k-j}\right] (z)^{-k} \\ &= \sum_{k=0}^{\infty} h_A^{\alpha}(k) z^{-k} \end{aligned} \tag{2.60}$$

Do mesmo modo, será apresentado de seguida as suas respostas impulsionais, $h_t^{\alpha}(k)$ e $h_A^{\alpha}(k)$, dadas por:

$$h_T^{\alpha}(k) = \begin{cases} \left(\frac{2}{T}\right)^{\alpha} \sum_{j=0}^{k} (-1)^j \binom{\alpha}{j}\binom{-\alpha}{k-j}, & k \geq 0 \\ 0, & k < 0 \end{cases} \tag{2.61}$$

$$h_A^{\alpha}(k) = \begin{cases} \left(\frac{8}{7T}\right)^{\alpha} \sum_{j=0}^{k} (-1)^j \left(\frac{1}{7}\right)^{k-j} \binom{\alpha}{j}\binom{-\alpha}{k-j}, & k \geq 0 \\ 0, & k < 0 \end{cases} \tag{2.62}$$

É de destacar que a aplicação do método de expansão de em série de potências leva a respostas impulsionais dos integradores e diferenciadores fracionários com uma duração infinita, ou seja, dadas na forma de um filtro IIR. Em termos práticos, estas sequências devem ser truncadas gerando aproximações na forma de um filtro de resposta impulsional finita (filtro FIR) [34][36].

2.9. FILTRO IIR FRACIONÁRIO

Nesta subsecção serão abordados os filtros IIR fracionários [5][20][36]. Para tal considera-se que a resposta impulsional $h^{\alpha}(k)$ do operador fracionário está definida para $k \geq 0$. A função racional $H(z^{-1})$ que aproxima a função irracional $H^{\alpha}(z^{-1})$ possui a forma de:

$$H(z^{-1}) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} \tag{2.63}$$

A expressão anterior pode ser reescrita na forma $H(z)A(z) = B(z)$, e considerando que $h^{\alpha}(k)$ é dado aproximadamente pela resposta impulsional de $H(z^{-1})$ então esta função possui a seguinte equação das diferenças:

$$h^{\alpha}(k) + \sum_{i=1}^{n} a_i \, h^{\alpha}(k-i) = \begin{cases} b_k, & k = 0,1,\ldots,m \\ 0, & k > m \end{cases} \tag{2.64}$$

Como se pode constatar pela análise da expressão anterior, está-se na presença de um sistema de equações lineares, que podem ser manipuladas de diferentes formas para determinação dos coeficientes a_k e b_k. Neste sentido, serão seguidamente apresentadas três soluções lineares subóptimas, nomeadamente os métodos de Padé, Prony e Shanks [5][36].

2.9.1. MÉTODO DE PADÉ

O método de Padé fornece uma aproximação $H(z^{-1})$ que produz um encaixe perfeito da reposta impulsional $h(k)$ na resposta impulsional $h^{\alpha}(k)$ para $0 \leq k \leq m+n$, isto é, para os primeiros $m+n+1$ valores de k. Assim, através da equação (2.65) obtém-se:

$$h^{\alpha}(k) + \sum_{i=1}^{n} a_i \, h^{\alpha}(k-i) = \begin{cases} b_k, & k = 0,1,\ldots,m \\ 0, & k = m+1,\ldots,m+n \end{cases} \tag{2.65}$$

Em que $h^{\alpha}(k) = 0$ para $h < 0$. Na forma matricial, o sistema de equações (2.65) pode ser descrito da seguinte forma:

$$\left[\begin{array}{cccc} h^{\alpha}(0) & 0 & \cdots & 0 \\ h^{\alpha}(1) & h^{\alpha}(0) & \cdots & 0 \\ h^{\alpha}(2) & h^{\alpha}(1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h^{\alpha}(m) & h^{\alpha}(m-1) & \cdots & h^{\alpha}(m-n) \\ \hline h^{\alpha}(m+1) & h^{\alpha}(m) & \cdots & h^{\alpha}(m-n+1) \\ \vdots & \vdots & \ddots & \vdots \\ h^{\alpha}(m+n) & h^{\alpha}(m+n-1) & \cdots & h^{\alpha}(m) \end{array}\right] \left[\begin{array}{c} 1 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{array}\right] = \left[\begin{array}{c} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_m \\ \hline 0 \\ \vdots \\ 0 \end{array}\right] \tag{2.66}$$

No método de aproximação de Padé, a obtenção dos coeficientes a_k e b_k a partir do sistema apresentado na equação (2.66) é realizada em dois passos, sendo que inicialmente resolve-se para os coeficientes do denominador a_k (usando a segunda parte da equação (2.66)) e de seguida resolve-se para os coeficientes do numerador b_k (usando a primeira parte da equação (2.66)) [35][36].

2.9.2. MÉTODO DE PRONY

O método de Prony difere do método de aproximação de Padé na forma de determinar os coeficientes do denominador $a_k (k = 1,2, \dots, n)$. Assim, o erro de modelação é dado por:

$$E_p(z) = A(z)H^{\alpha}(z) - B(z) \tag{2.67}$$

Ao qual corresponde a seguinte equação nos tempos (onde o símbolo * denota a operação de convolução):

$$e_p(k) = \hat{b_k} - b_k = a_k * h^{\alpha}(k) - b_k \tag{2.68}$$

Dado que $b_k = 0$ para $k > m$, o erro $e_p(k)$ pode ser expresso da seguinte forma:

$$e_p(k) = \begin{cases} h^{\alpha}(k) + \sum_{i=1}^{n} a_i \, h^{\alpha}(k-i) - b_i, & k = 0,1, \dots, m \\ h^{\alpha}(k) + \sum_{i=1}^{n} a_i \, h^{\alpha}(k-i), & k > m \end{cases} \tag{2.69}$$

O qual é uma função linear dos coeficientes a_k e b_k.

Assumindo que o erro $e_p(k) = 0$ na segunda parte da equação (2.69), para $K = m + 1, m + 2, \dots, N - 1$, vem:

$$h^{\alpha}(k) + \sum_{i=1}^{n} a_i\, h^{\alpha}(k-i) = 0, \quad k = m+1, m+2, \dots, N-1 \tag{2.70}$$

Na equação anterior transcrita, *N* representa o número de amostras utilizadas da sequência impulsional $h^{\alpha}(k)$ desejada. Importa realçar que a equação (2.70) só depende dos coeficientes a_k.

Na forma matricial, o sistema de equação (2.70) é descrito da seguinte forma:

$$\begin{bmatrix} h^{\alpha}(m) & h^{\alpha}(m-1) & \cdots & h^{\alpha}(m-n+1) \\ h^{\alpha}(m+1) & h^{\alpha}(m) & \cdots & h^{\alpha}(m-n+2) \\ \vdots & \vdots & \ddots & \vdots \\ h^{\alpha}(N-2) & h^{\alpha}(N-3) & \cdots & h^{\alpha}(N-n-1) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = - \begin{bmatrix} h^{\alpha}(m+1) \\ h^{\alpha}(m+2) \\ \vdots \\ h^{\alpha}(N-1) \end{bmatrix} \tag{2.71}$$

Desta forma os coeficientes a_k são determinados minimizando o erro quadrático. Os coeficientes b_k do numerador são calculados da mesma forma que o método de Padé [36].

2.9.3. MÉTODO DE SHANKS

O método de Shanks aqui descrito fornece uma alternativa ao de Prony na forma de determinar os coeficientes do numerador $b_k(k = 1,2, \dots, n)$. Assim, em vez de forçar um ajuste perfeito para os primeiros $m + 1$ valores da sequência impulsional $h^{\alpha}(k)$, o método de Shanks minimiza o erro $e_s(k) = h^{\alpha}(k) - \hat{h}(z)$ através dos mínimos quadrados, para as *N* amostras consideradas no intervalo $[0, N - 1]$.

Como se pode observar, pela seguinte equação (2.72) a aproximação $H(z)$ é subdividida em duas funções colocadas em série, $B(z)$ e $A(z)$:

$$H(z) = B(z)\left\{\frac{1}{A(z)}\right\} \tag{2.72}$$

O processo de determinação dos coeficientes de a_k e b_k é estabelecido em dois passos. Primeiro resolve-se para os coeficientes do denominador a_k e, de seguida, resolve-se para os coeficientes do numerador b_k.

Assim, no primeiro passo, calcula-se o denominador $A(z)$ usando o método de Prony, ou seja, obtendo a sua solução pelos mínimos quadrados sobre o intervalo $[m+1, N-1]$. Após a determinação de $A(z)$, a resposta impulsional $g(k)$ referente à função $1/A(z)$ é obtida, por exemplo, de modo recursivo através da seguinte expressão com $g(k) = 0$ para $k < 0$:

$$g(k) = \delta(k) - \sum_{i=1}^{n} a_i \, g(k-i) \tag{2.73}$$

Contrariamente ao método de Prony, para determinar os coeficientes do numerador b_k da função $B(z)$ o método de Shanks em vez de forçar o erro $e_p(k) = a_k * h^{\alpha}(k) - b_k = 0$ para $k = 0, \dots, m$ minimiza o erro $e_s(k) = h^{\alpha}(k) - \hat{h}(k)$, logo temos:

$$e_s(k) = h^{\alpha}(k) - \sum_{i=1}^{m} b_i \, g(k-i), \mathrm{k} = 0,1, \dots, \mathrm{N}-1 \tag{2.74}$$

Na realidade, o método de Shanks pode ser formulado em termos de obtenção da solução de um sistema de equações sobredeterminado através do método dos mínimos quadrados. Esta é a forma considerada neste estudo para a determinação dos coeficientes b_k [36]. Esta formulação equivalente tem por base a equação (2.74), em que se coloca o erro $e_s(k) = 0$ para $k = 0,1, \dots, N-1$, ou seja:

$$h^{\alpha}(k) - \textstyle\sum_{i=1}^{m} b_i \, g(k-i), k = 0,1, \dots, N-1 \tag{2.75}$$

Em notação matricial a equação anterior toma a seguinte forma:

$$\begin{bmatrix} g(0) & 0 & 0 & 0 & 0 \\ g(1) & g(0) & 0 & 0 & 0 \\ g(2) & g(1) & g(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g(N-1) & g(N-2) & g(N-3) & \cdots & g(N-m-1) \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} h^{\alpha}(0) \\ h^{\alpha}(1) \\ h^{\alpha}(2) \\ \vdots \\ h^{\alpha}(N-1) \end{bmatrix} \tag{2.76}$$

O método de Shanks é computacionalmente mais pesado que o método Prony; no entanto, a redução obtida no erro pode ser desejável em algumas aplicações [5].

2.10. INTERPRETAÇÃO DE OPERADORES FRACCIONÁRIOS NO DOMÍNIO DAS FREQUÊNCIAS

Nesta secção é apresentado a interpretação dos operadores fraccionários no domínio das frequências, comparando os ideais com os reais.

2.10.1. FILTROS FRACCIONÁRIOS IDEAIS

As Figura 16 e 17 mostram a resposta em frequência de filtros fraccionários ideais para o caso do integrador e do diferenciador, respectivamente.

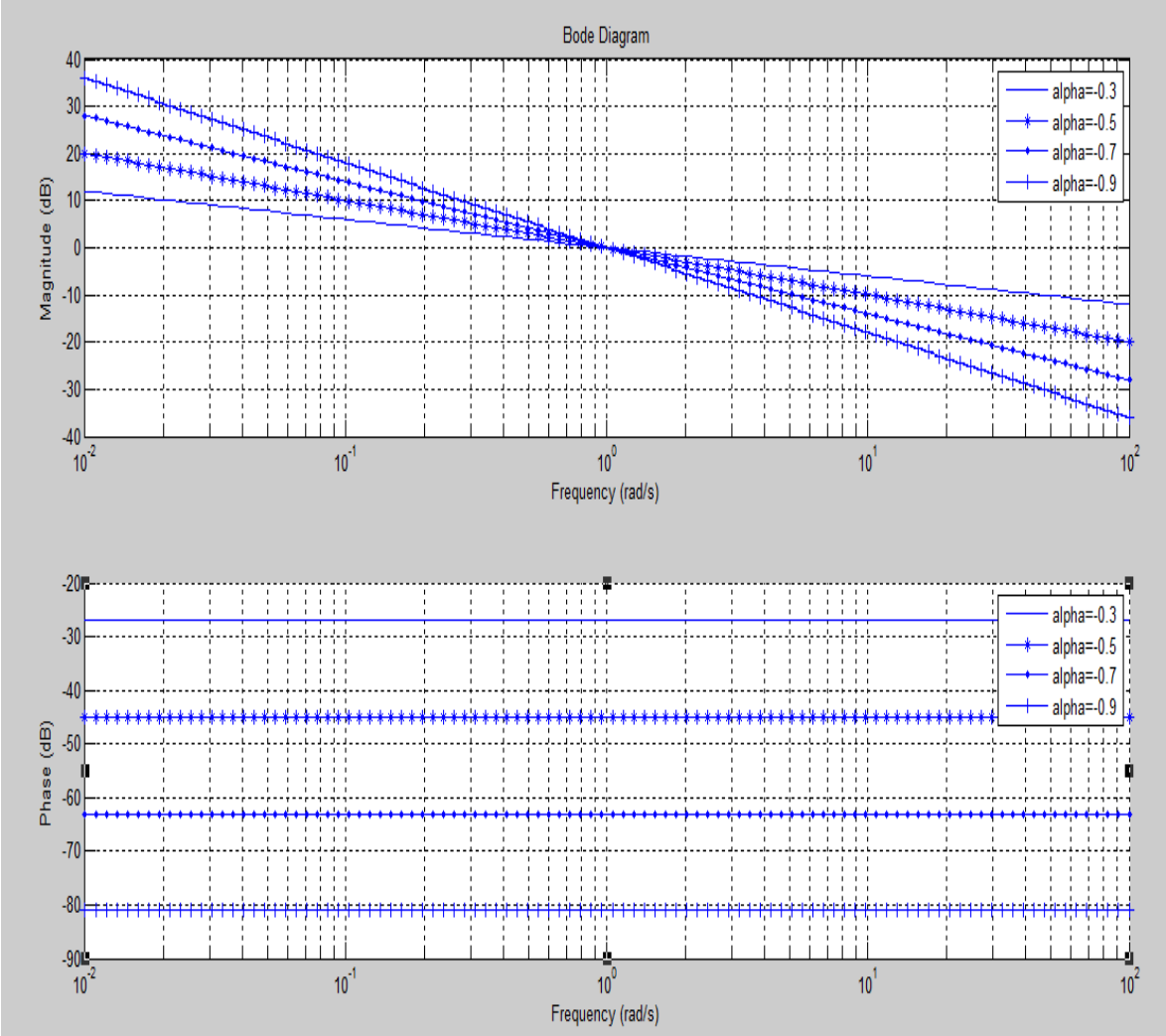

Figura 16: Diagramas de Bode de um filtro ideal para α={-0.3,-0.5,-0.7,-0.9} (Integrador Fraccionário)

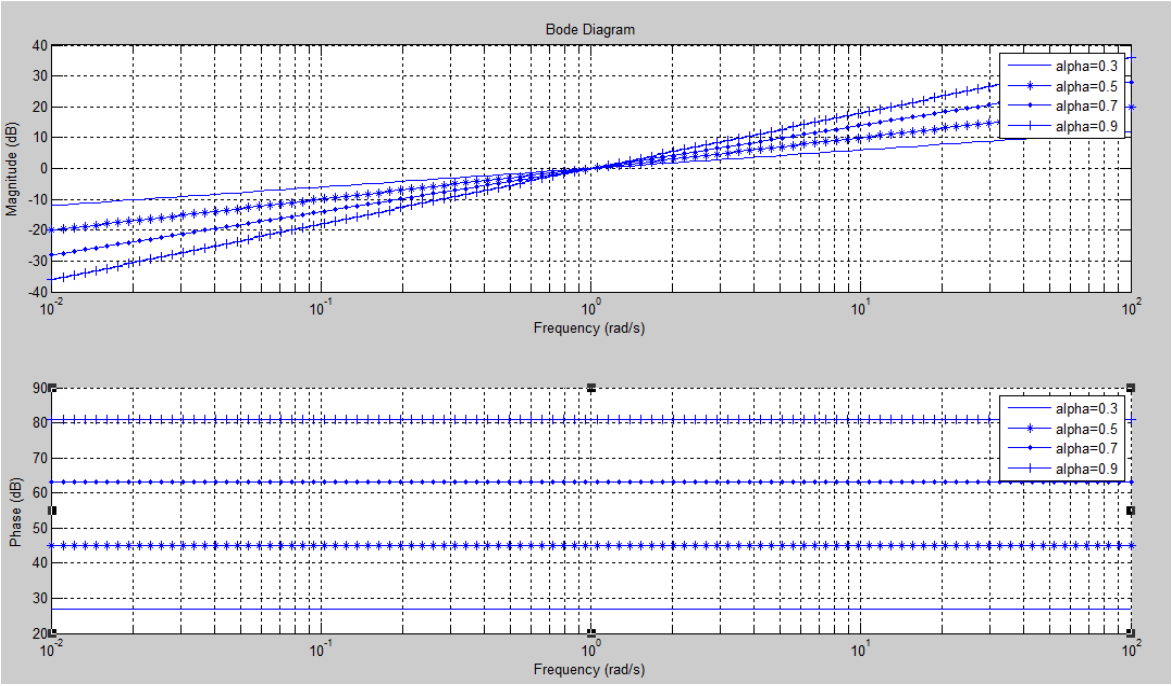

Figura 17: Diagramas de Bode de um filtro ideal para α={0.3,0.5,0.7,0.9} (Diferenciador Fraccionário)

2.10.2. FILTRO FIR FRACCIONÁRIO

As Figuras 18 e 19 ilustram as respostas em frequência de filtros FIR fraccionários para o caso do integrador e do diferenciador, respectivamente.

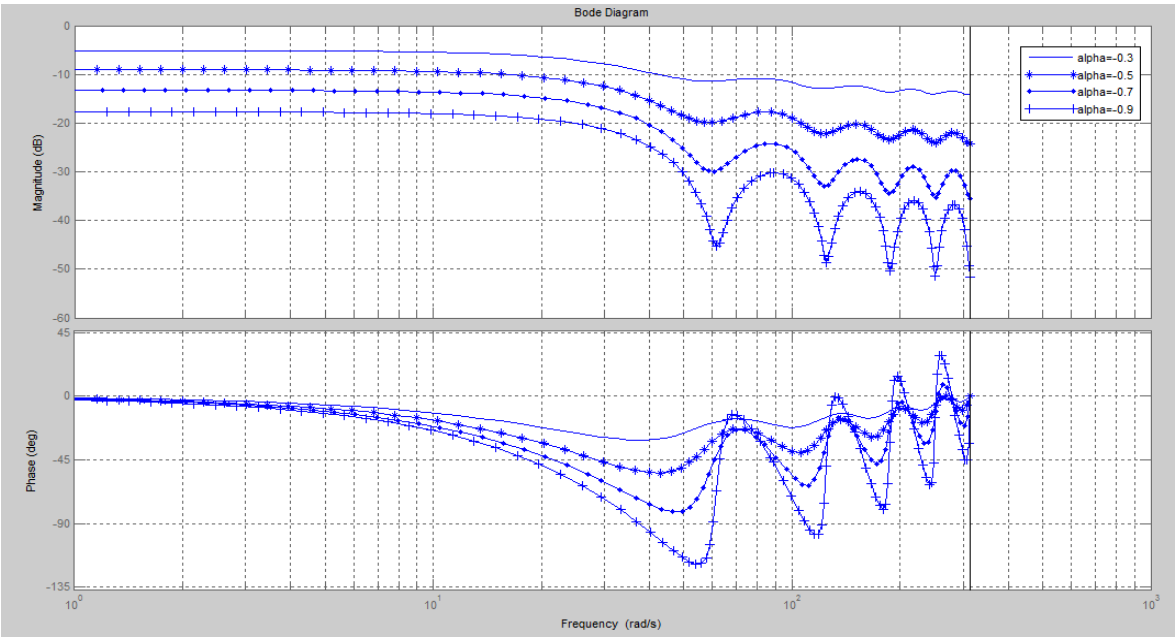

Figura 18: Diagramas de Bode de um filtro FIR com T=0.01s e N=10 para α={-0.3,-0.5,-0.7,-0.9}, aproximação de Euler (Integrador Fraccionário)

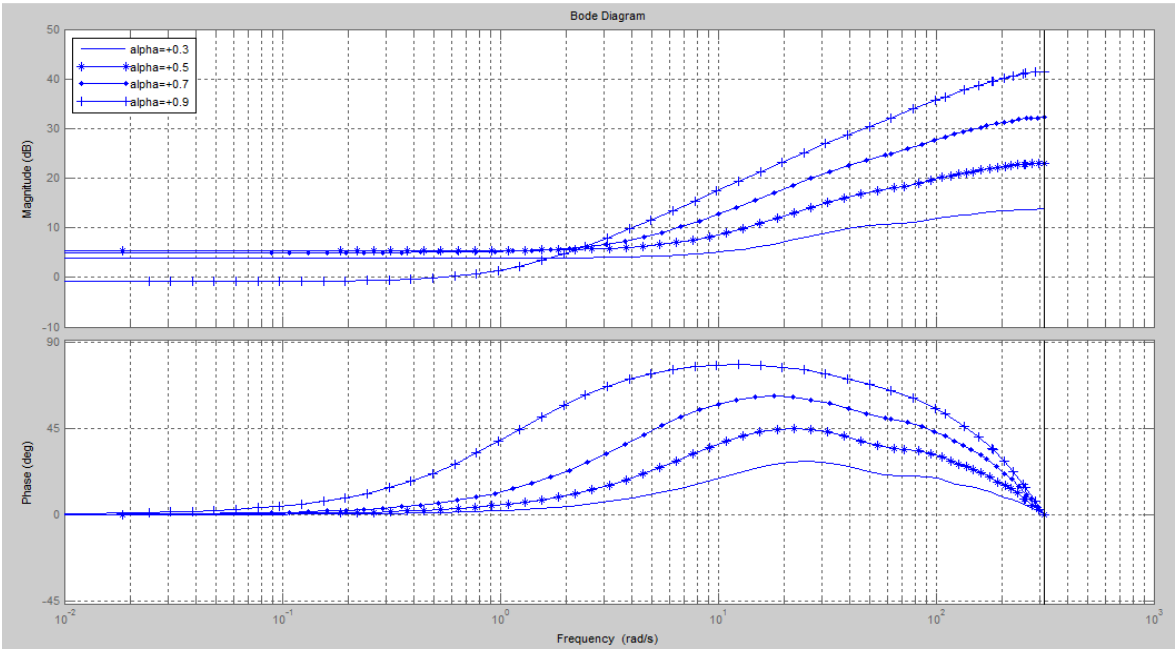

Figura 19: Diagramas de Bode de um filtro FIR com T=0.01s e N=10 para α={-0.3,-0.5,-0.7,-0.9}, aproximação de Euler (Diferenciador Fraccionário)

2.10.3. FILTRO IIR FRACCIONÁRIO

As Figuras 20 e 21 ilustram as respostas em frequência de filtros IIR fraccionários para o caso do integrador e do diferenciador, respectivamente.

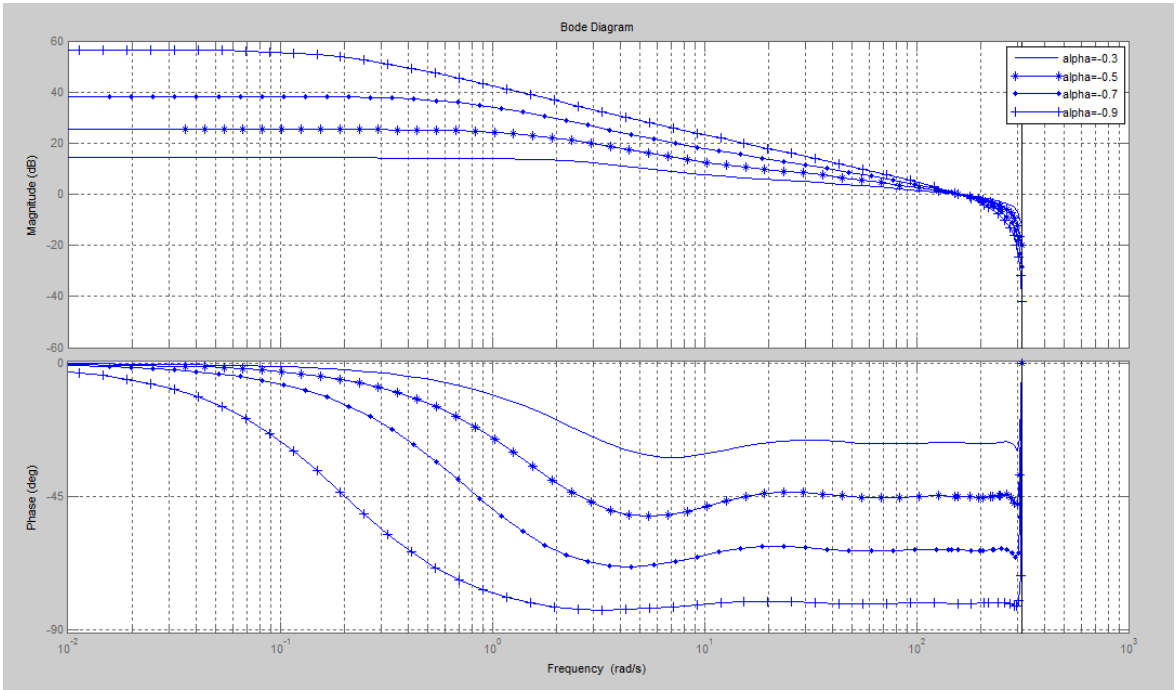

Figura 20: Diagramas de Bode de um filtro IIR com T=0.01s e OrdN=OrdD=4 para α={-0.3,-0.5,-0.7,-0.9}, aproximação de Tustin e método de Prony (Integrador Fraccionário)

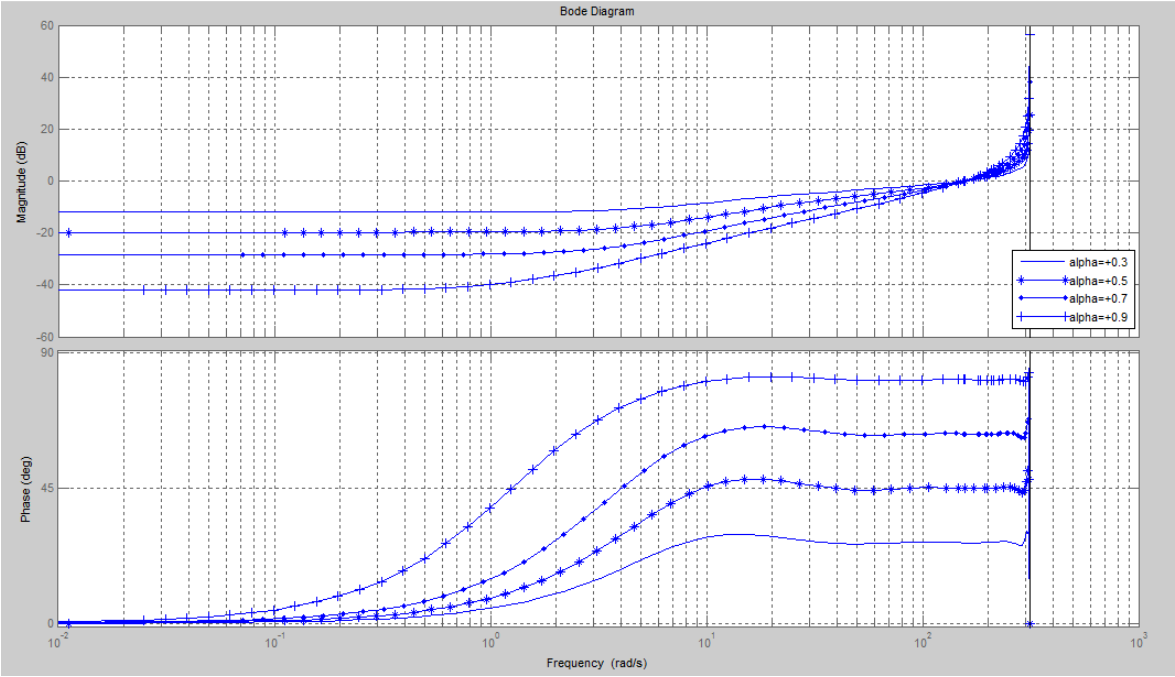

Figura 21: Diagramas de Bode de um filtro IIR com T=0.01 s e OrdN=OrdD=4 para α={0.3,0.5,0.7,0.9}, aproximação de Tustin e método de Prony (Diferenciador Fraccionário)

Pela observação das figuras anteriores é possível verificar que os valores esperados para as amplitudes e fases só são atingidos durante uma pequena gama limite de frequências, tanto para o filtro FIR como para o IIR. Os filtros ideais são caracterizados por terem uma amplitude com um declive dado por α20 dB/dec e por uma fase dada por $\alpha\pi/2$, em que $\alpha \in \mathcal{R}$.

3. IMPLEMENTAÇÃO DE FILTROS DIGITAIS E INTERFACE GRÁFICA

Neste capítulo é exposto a implementação de filtros digitais fracionários e as suas diferentes estruturas. Posteriormente, é apresentado a interface gráfica desenvolvida para a implementação dos filtros digitais e suas estruturas bem como a referência das suas potencialidades .Para concluir, são ilustrados os resultados obtidos através da interface gráfica desenvolvida.

3.1. IMPLEMENTAÇÃO DE FILTROS DIGITAIS

Para processar sinais é necessário conceber e implementar sistemas denominados filtros (ou analisadores de espectros em alguns contextos). A conceção de filtros depende de vários fatores, como o tipo de filtro (isto é, IIR ou FIR) ou o tipo de implementação (estrutura).

Vai ser considerada a importante classe dos sistemas lineares discretos invariantes no tempo, que é caracterizado pela seguinte equação das diferenças:

$$y(n) = \sum_{k=0}^{q} b_k x(n-k) - \sum_{k=1}^{p} a_k y(n-k) \tag{3.1}$$

Como foi demonstrado, a partir da transformada dos z, os sistemas invariantes no tempo também são caracterizados pela função de transferência (3.2), formada por uma razão de dois polinómios em z^{-1}.

$$H(z) = \frac{\sum_{k=0}^{q} b_k z^{-k}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{3.2}$$

Os pólos e os zeros da função do sistema podem ser obtidos dependendo da escolha efetuada para os parâmetros do sistema b_k e a_k e que determinam a resposta em frequência do sistema.

Em geral, pode ser definido (através da equação (3.1) um procedimento computacional (algoritmo) para determinar a sequência de saída ($y(n)$) do sistema a partir da sequência de entrada e de saída ($x(n)\ e\ y(n)$). Contudo, os algoritmos utilizados através da equação (3.1) podem ser dispostos em conjuntos equivalentes de equações das diferenças. Cada conjunto de equações define um procedimento computacional ou um algoritmo para implementação do sistema. A partir de cada conjunto de equações pode ser construído um diagrama de blocos constituído por interligação de elementos de atrasos, multiplicadores e somadores. Na Figura 22 é possível visualizar esses elementos básicos e os seus símbolos padrão.

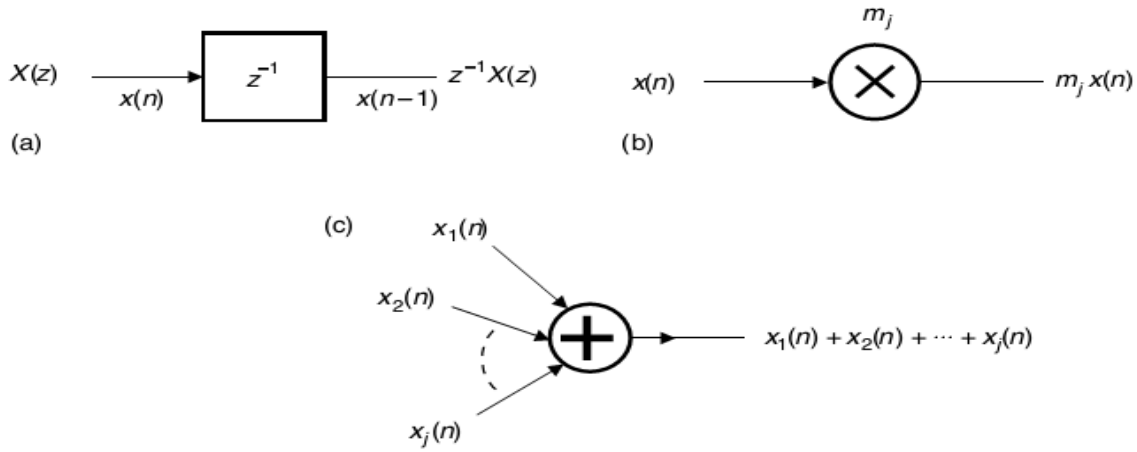

Figura 22: Representação clássica dos elementos básicos de um filtro digital: (a) Atraso; (b) Multiplicador; (c) Somador [4]

Na Figura 23 é apresentada uma representação em fluxograma de sinal que serve de alternativa à representação anteriormente observada.

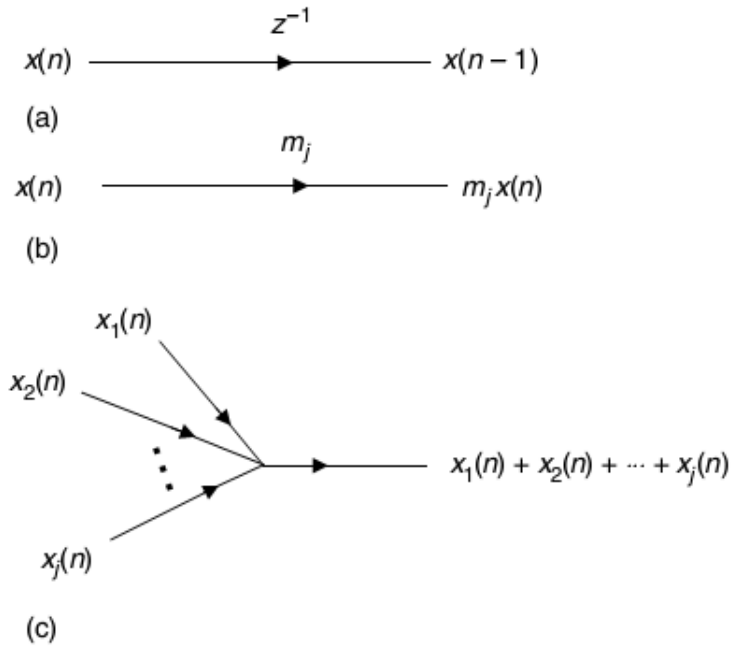

Figura 23: Representação dos elementos básicos de um filtro digital num fluxograma de sinal: (a) Atraso; (b) Multiplicador; (c) Somador [4]

Caso o sistema seja implementado através de *software*, o diagrama de blocos ou o conjunto de equações são obtidas através da reformulação da equação (3.1), que podem ser convertidos num programa para execução num computador digital. Como alternativa, a forma da estrutura dos diagramas de blocos implica a configuração do *hardware* para implementação do sistema.

Neste ponto do projeto é necessário ter em conta os principais fatores que influenciam a escolha da estrutura necessária para a implementação de filtros digitais. Estes fatores são a complexidade computacional, requisitos de memória e o comprimento finito da palavra para efetuar os cálculos.

A complexidade computacional consiste no número de operações aritméticas (multiplicadores, divisores e somadores) necessárias para obter o valor de saída do sistema ($y(n)$). No início, estes fatores eram os únicos métodos utilizados para medir a complexidade computacional. No entanto, com os recentes desenvolvimentos do projeto de filtros digitais e a fabricação de sofisticados e programáveis dispositivos de processamento digital de sinal, é necessário ter em conta outros fatores como o número de vezes em que

existe uma comparação entre dois valores na saída do sistema e o número de vezes que é efetuada uma pesquisa a partir da memória [19][21].

Outro fator a ter em conta diz respeito aos requisitos de memória, que consiste no número de memória necessária para armazenar os parâmetros do sistema, da entrada, da saída e outros utilizados em cálculos intermédios [12].

O tamanho da palavra é outro fator que deve ser considerado, o qual refere-se aos efeitos da quantização que estão inerentes a qualquer implementação digital, quer por *hardware* quer por *software*. Os parâmetros do sistema são necessariamente representados com uma precisão finita. Os cálculos que são executados no processo de cálculo na saída devem ser arredondados ou truncados conforme os limites de precisão do computador ou do *hardware* utilizado na implementação, se esses cálculos forem realizados através de um ponto fixo ou de uma vírgula flutuante é necessário ter outras considerações. Todos esses problemas são denominados por comprimento de palavra finito e são de extrema importância, porque influenciam a escolha da estrutura a utilizar no sistema [14].

3.1.1. ESTRUTURAS DOS FILTROS DIGITAIS NÃO-RECURSIVOS (FIR)

Os filtros não recursivos (FIR) são caracterizados pela seguinte equação das diferenças:

$$y(n) = \sum_{k=0}^{N} b_k x(n-k) \tag{3.3}$$

O coeficiente b_k está diretamente relacionado com a resposta impulsional do sistema, isto é, $b_k = h(k)$. Devido ao comprimento finito da resposta impulsional, os filtros não recursivos estão relacionados com a duração finita da resposta impulsional. Logo, a equação (3.3) pode ser rescrita da seguinte forma:

$$y(n) = \sum_{k=0}^{N} h(k) x(n-k) \tag{3.4}$$

Aplicando a transformada de z à equação (3.4) obtém-se a equação onde é possível visualizar a relação entre a entrada e saída:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{N} b_k z^{-k} = \sum_{k=0}^{N} h(k) z^{-k} \tag{3.5}$$

Na prática, a equação (3.5) pode ser implementada de várias formas distintas, utilizando como elementos básicos os blocos somadores, blocos de multiplicação e blocos de atraso (*Delay*).

3.1.1.1. IMPLEMENTAÇÃO DE FILTROS FIR NA ESTRUTURA DIRETA

A implementação mais simples de um filtro digital FIR deriva da equação (3.3). Na Figura 24 é possível visualizar a estrutura resultante da implementação de um filtro digital não recursivo na topologia direta, sendo os coeficientes multiplicadores obtidos diretamente da função de transferência.

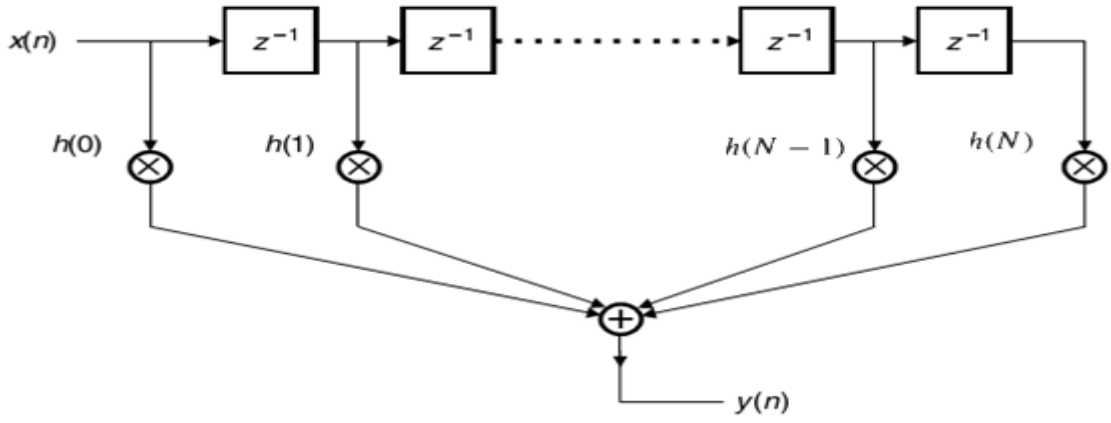

Figura 24: Implementação em forma direta de filtro digital FIR [4]

Esta estrutura também é conhecida como a forma canónica direta, onde está subjacente que qualquer estrutura realiza uma função de transferência com o mínimo de atrasos, multiplicadores e somadores.

Uma alternativa à equação da forma direta canónica, seria fatorizando a expressão $H(z)$:

$$H(z) = \sum_{k=0}^{N} h(k) z^{-k} \tag{3.6}$$

$$= h(0) + z^{-1}(h(1) + z^{-1}\big(h(2) + \cdots z^{-1}\big(h(N-1) + z^{-1}h(N)\big) \ldots\big)$$

Na Figura 25 pode visualizar-se a representação alternativa de uma implementação na forma direta de filtro digital FIR correspondente à equação (3.6).

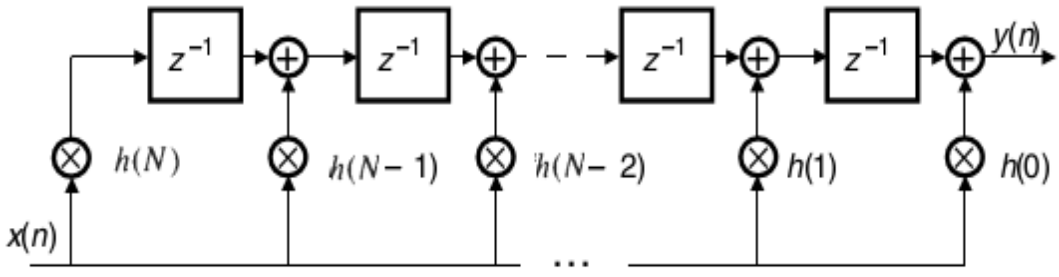

Figura 25: Representação alternativa de uma implementação na forma direta de filtro digital FIR [4]

3.1.1.2. IMPLEMENTAÇÃO DE FILTROS FIR NA ESTRUTURA EM CASCATA

Como foi referido anteriormente, a equação (3.3) pode ser utilizada para realizar uma série de estruturas equivalentes. No entanto, os coeficientes dessas realizações não podem expor a resposta impulsional do filtro ou função de transferência correspondente. Um exemplo importante desta implementação denomina-se implementação em forma de cascata, que como o nome indica, corresponde a uma série de filtros FIR, normalmente de segunda ordem, ligados em cascata, como é possível visualizar na Figura 26.

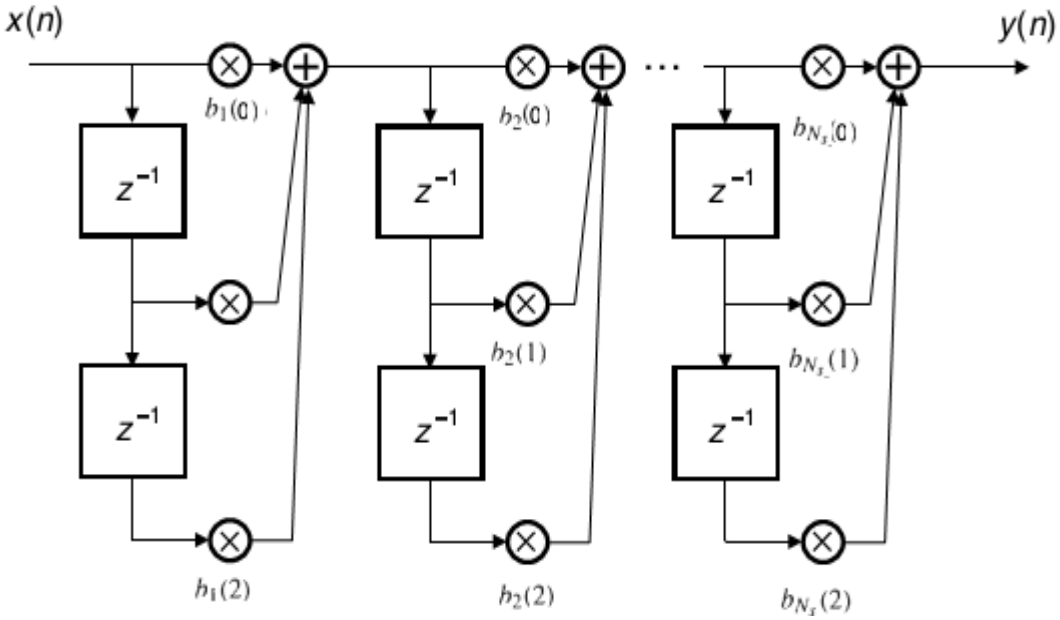

Figura 26: Implementação em forma de cascata de um filtro digital FIR [4]

A função de transferência correspondente à implementação em forma de cascata de um filtro digital FIR pode ser expressa da seguinte forma:

$$H(z) = A\prod\nolimits_{k=1}^{N_s}(1 + b_k(1)z^{-1} + b_k(2)z^{-2}) \tag{3.7}$$

Onde N_s representa o numero de blocos em cascata.

3.1.2. ESTRUTURA DE FILTROS DIGITAIS RECURSIVOS (IIR)

Nesta secção, serão exploradas as diferentes estruturas dos filtros IIR descritas pelas equações (3.1) e (3.2). Tal como no caso dos sistemas FIR, existem vários tipos de estruturas ou realizações, tais como a estrutura na forma direta e em cascata. Além disso, os sistemas IIR também podem ser implementados com uma estrutura em forma paralela.

Um filtro causal de resposta infinita (IIR) é dado pela função do sistema racional apresentada na equação (3.8):

$$H(z) = \frac{B(z)}{A(Z)} = \frac{\sum_{k=0}^{q} b(k)z^{-k}}{1 + \sum_{k=1}^{p} a(k)z^{-k}} \tag{3.8}$$

A entrada $x(n)$ e a saída $y(n)$ relacionam-se pela equação das diferenças com coeficientes constantes, como é possível visualizar pela seguinte equação:

$$y(n) = \sum_{k=0}^{q} b(k)x(n-k) - \sum_{k=0}^{p} a(k)y(n-k) \tag{3.9}$$

3.1.2.1. IMPLEMENTAÇÃO DE FILTROS IIR NA ESTRUTURA DIRETA

Existem dois tipos de estruturas de filtro digital na forma direta, nomeadamente a forma direta I e a forma direta II. A de forma direta I é uma implementação que resulta da equação (3.9), podendo ser transcrita como um par de equações de diferenças:

$$w(n) = \sum_{k=0}^{q} b(k)x(n-k) \tag{3.10}$$

$$y(n) = w(n) - \sum_{k=0}^{p} a(k)y(n-k) \tag{3.11}$$

A primeira equação (3.10) corresponde a um filtro digital de entrada x(n) e saída w(n) e a segunda equação (3.11) corresponde a um filtro com polos de entrada $w(n)$ e saída $y(n)$. Portanto, esse par de equações correspondem a uma cascata de dois sistemas, como é ilustrado na Figura 27, sendo a função de transferência dada por:

$$Y(z) = \frac{1}{A(z)}[B(z)X(z)] \tag{3.12}$$

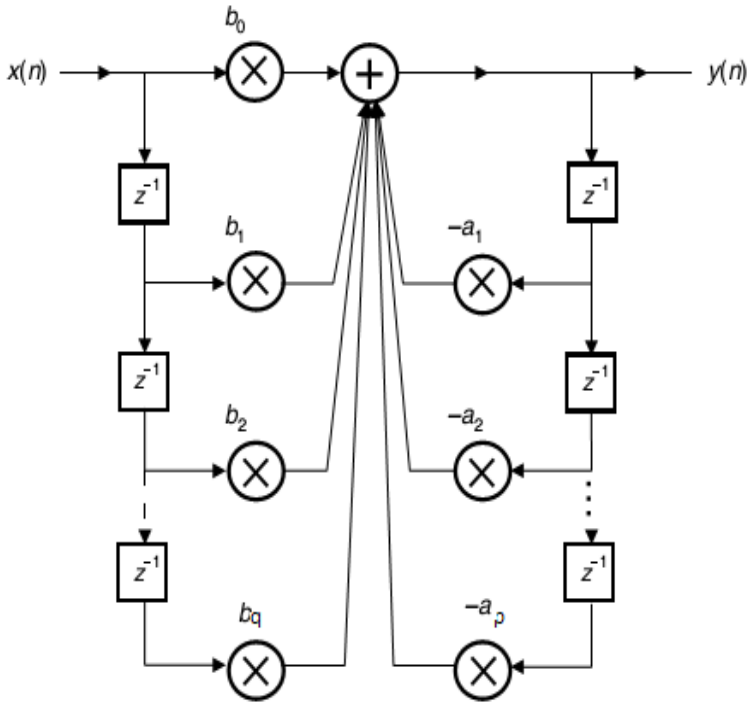

Figura 27: Implementação direta I de um filtro IIR [4]

Os requisitos computacionais necessários para implementar um filtro IIR numa tipologia direta I são:

- Número de multiplicadores*:* $p+q+1$ por amostra de saída;
- Número de somadores: $p+q$ por amostra de saída;
- Número de atrasos: $p+q$.

A tipologia direta II é obtida através da inversão da ordem da cascata de $B(z)$ e $1/A(z)$, como é possível observar na Figura 28.

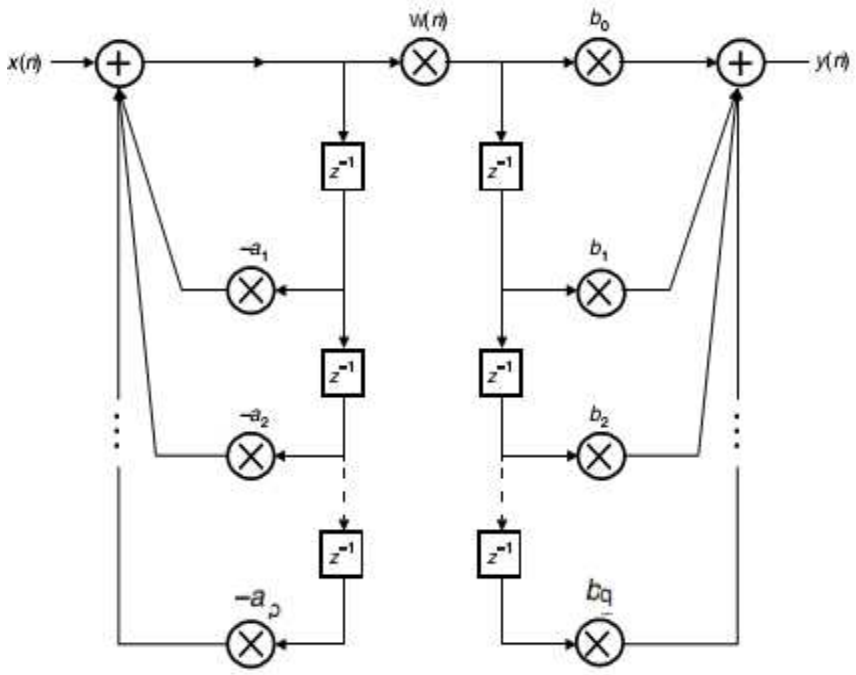

Figura 28: Inversão da ordem da cascata da implementação direta I de um filtro IIR [4]

Como já foi referido a estrutura anterior, representa a inversão da ordem da cascata, onde as equações das diferenças podem ser representadas pelas seguintes expressões:

$$w(n) = x(n) - \sum_{k=1}^{p} a(k)w(n-k) \tag{3.13}$$

$$y(n) = \sum_{k=0}^{q} b(k)w(n-k) \tag{3.14}$$

A estrutura apresentada na Figura 28 pode ser simplificada, devido ao facto de os três conjuntos de atraso serem o mesmo. Assim, esses atrasos podem ser combinados quando p=q. Os requisitos computacionais necessários para implementar um filtro IIR numa tipologia direta II são:

- Número de multiplicadores: $p+q+1$ por amostra de saída;
- Número de somadores: $p+q$ por amostra de saída;
- Número de atrasos: máx($p+q$).

A estrutura na forma direta II pode ser considerada como forma canónica, pois utiliza o mínimo número de atrasos para uma função $H(z)$. Na Figura 29 é possível visualizar a forma canónica numa estrutura na forma direta II.

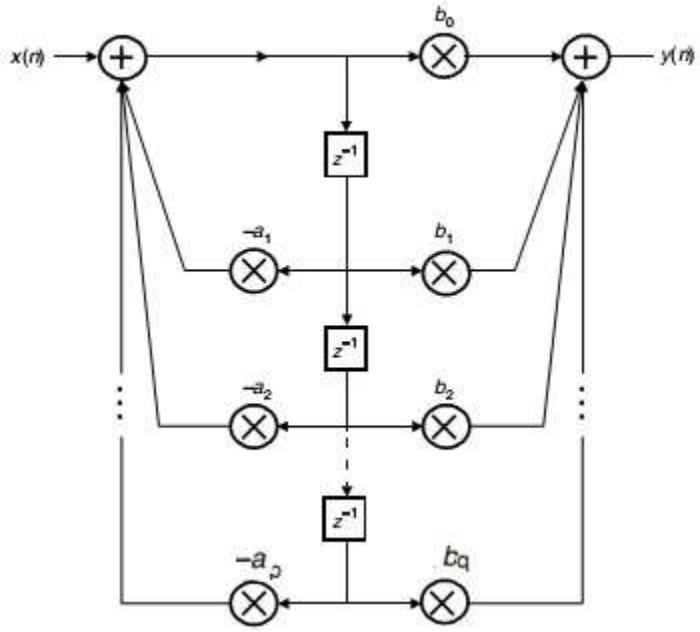

Figura 29: Implementação direta II de um filtro com *p=q* [4]

3.1.2.2. IMPLEMENTAÇÃO DE FILTROS IIR NA ESTRUTURA CASCATA

Da mesma forma que existem diversas implementações de filtros digitais FIR, os filtros IIR também podem ser implementados de formas variadas. De salientar, que a implementação na forma cascata, como é apresentada na Figura 30, os blocos representam funções de transferências de segunda ou de primeira ordem.

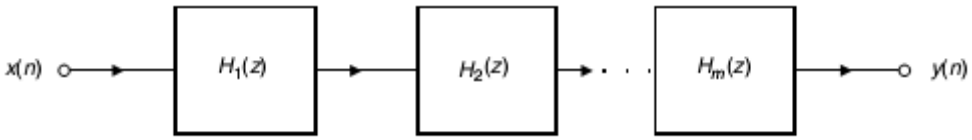

Figura 30: Diagrama de blocos em cascata

De facto, a forma em cascata com base em blocos de segunda ordem, está associada à decomposição da seguinte equação:

$$H(z) = \frac{\sum_{k=0}^{q} b(k) z^{-k}}{1 + \sum_{k=1}^{p} a(k) z^{-k}} = A \prod_{k=1}^{\max\{p,q\}} \frac{1 - \beta_k z^{-1}}{1 - \alpha_k z^{-1}} \tag{3.15}$$

$$H_k(z) = \frac{1 + \beta_{1k} z^{-1} + \beta_{2k} z^{-2}}{1 + \alpha_{1k} z^{-1} + \alpha_{2k} z^{-2}} \tag{3.16}$$

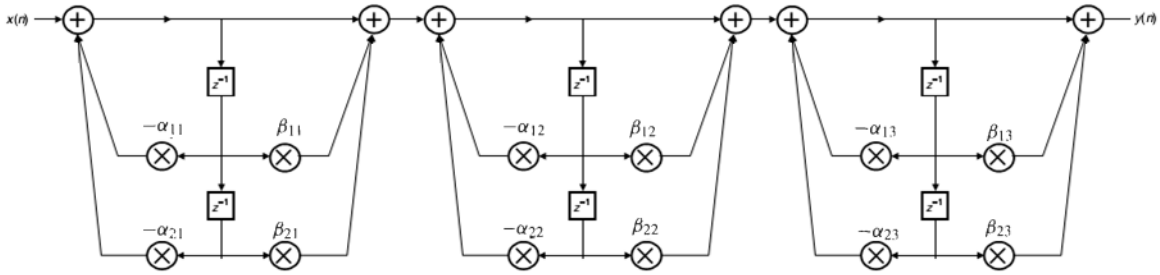

Figura 31: Filtro IIR de sexta ordem, implementado como uma cascata de três sistemas de segunda ordem com uma implementação direta II

Como exemplo, pode observar-se na Figura 31 um filtro digital de sexta ordem implementado com uma cascata de três sistemas de segunda ordem utilizando a forma direta II.

Existe uma flexibilidade em relação à forma como se implementa um sistema em cascata. Além disso, existem diversos modos de agrupar os pólos e zeros em pares e de estabelecer a ordem do encadeamento das estruturas [10].

3.1.2.3. IMPLEMENTAÇÃO DE FILTROS IIR NA ESTRUTURA PARALELA

Ao contrário da estrutura em cascata, em vez decompor a função $H(z)$ em alternativa será efetuada a expansão da função em frações parciais. Por exemplo:

$$H(z) = \frac{\sum_{k=0}^{q} b(k)z^{-k}}{1 + \sum_{k=1}^{p} a(k)z^{-k}} = A\frac{\prod_{k=1}^{q}(1 - \beta_k z^{-1})}{\prod_{k=1}^{p}(1 - \alpha_k z^{-1})} \tag{3.17}$$

Se a ordem do polinómio do numerador for inferior à ordem do denominador da função de transferência, $p>q$ e $\alpha_i \neq \alpha_k$, $H(z)$ pode ser expandida em uma soma de p fatores de primeira ordem como é apresentado na seguinte equação:

$$H(z) = \sum_{k=1}^{p} \frac{A_k}{1 - \alpha_k z^{-1}} \tag{3.18}$$

Nesta equação (3.18) os coeficientes A_k e α_k são em geral complexos. Essa expansão corresponde a uma soma de p funções do sistema de primeira ordem, podendo ser implementada pela conexão desses sistemas em paralelo. Se $h(n)$ for real, os pólos de $H(z)$ ocorrem em pares conjugados complexos e as raízes complexas da expansão em

frações parciais podem ser combinadas para formar sistemas de segunda ordem com coeficientes reais:

$$H(z) = \sum_{k=1}^{N_s} \frac{\gamma_{0k} + \gamma_{1k} z^{-1}}{1 + \alpha_1 z^{-1} + \alpha_2 z^{-2}} \tag{3.19}$$

Na Figura 32 é possível visualizar como exemplo um filtro de sexta ordem implementado com uma estrutura em paralelo de três sistemas de segunda ordem utilizando a forma direta II.

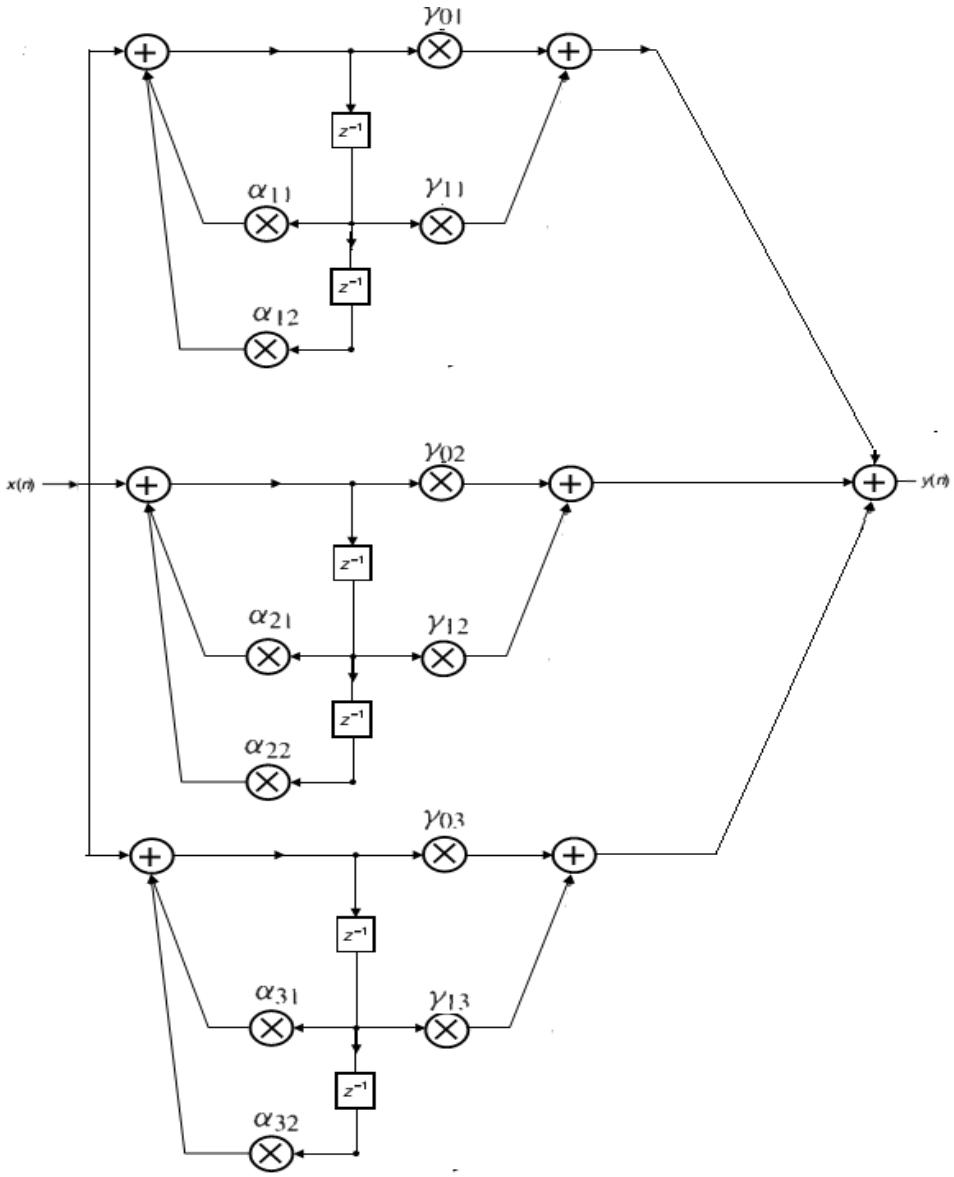

Figura 32: Filtro IIR de sexta ordem, implementado como um paralelo de três sistemas de segunda ordem com uma implementação direta II

3.2. APLICAÇÃO MATLAB

O MatLab é um programa interativo adequado para o cálculo numérico. O MatLab íntegra entre outros, a análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos em ambiente fácil de usar.

A utilização do MatLab foi um ponto essencial no desenvolvimento do trabalho porque com o auxílio das suas potencialidades foi possível desenvolver o código necessário para a implementação do projeto e ainda a utilização da ferramenta Guide demonstrou ser de bastante utilidade para o desenvolvimento da interface gráfica.

3.2.1. INTERFACE GRÁFICA E O DESENVOLVIMENTO DE UM AMBIENTE GRÁFICO

A interface gráfica é de enorme importância em qualquer projeto, pois dispõe da funcionalidade de interagir com o utilizador. O utilitário GUIDE permite criar uma interface amigável para o utilizador.

Após a análise e estudo das funcionalidades procedeu-se ao desenvolvimento da interface gráfica para o desenvolvimento de filtros digitais fraccionários [8][9][13][15][18]. De notar que ao longo desta secção são apresentadas funções inerentes à implementação da interface gráfica, as quais se encontram listadas no Anexo A.

Na Figura 33 é possível visualizar o resultado final da implementação do Menu Principal, onde pode ser escolhido o filtro pretendido ou selecionar o desenvolvimento de estruturas.

Figura 33: Menu Principal

O menu principal foi desenvolvido com as seguintes funcionalidades:

- Filtro IIR/FIR: Onde pode ser desenvolvido um filtro IIR/FIR e ainda visualizar as respostas normalizadas dos mesmos e obter a função de transferência do filtro;
- Estrutura: Onde pode ser visualizado os diferentes tipos de estruturas dependendo da escolha do tipo do filtro e ainda obter os coeficientes para cada estrutura;
- Sair: Opção que permite sair da interface gráfica.

Como fase seguinte no desenvolvimento da interface gráfica procedeu-se à realização da interface para o filtro IIR. Na Figura 34 é possível visualizar a interface gráfica para a opção do filtro IIR.

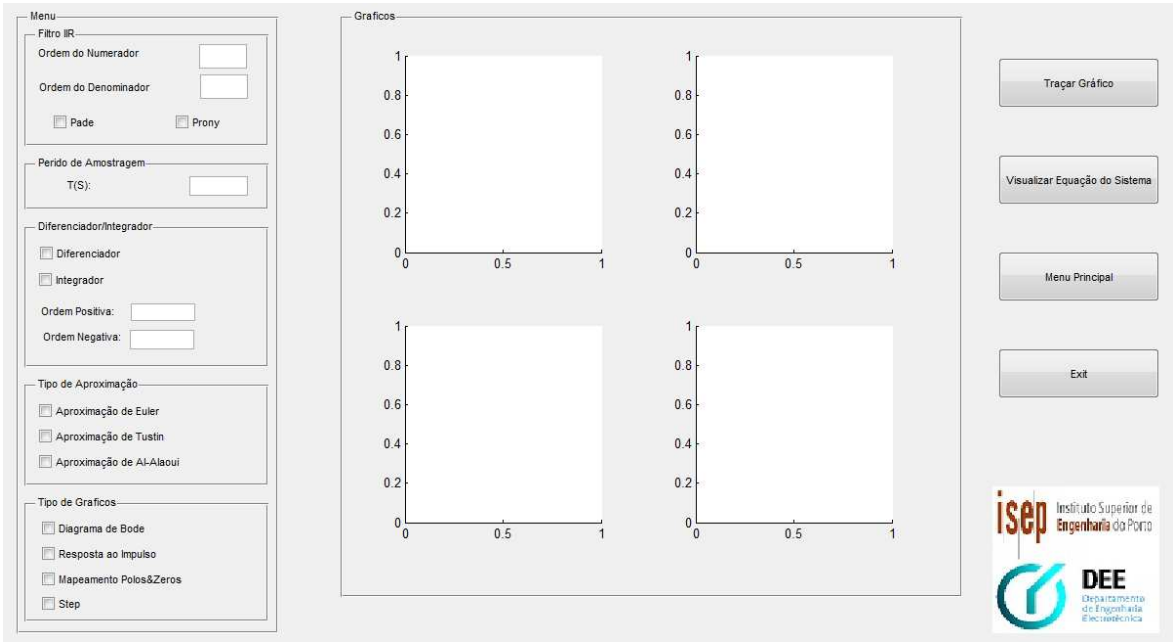

Figura 34: Interface gráfica da opção filtro IIR

A interface gráfica do filtro IIR pode ser dividida em três blocos principais: o menu, a janela dos gráficos e os botões. O menu pode ser dividido ainda em cinco blocos:

- Filtro IIR: onde é possível introduzir o valor da ordem do numerador/denominador e ainda escolher o método pretendido de Padé e Prony;
- Período de Amostragem: introdução do valor pretendido para o período de amostragem;
- Diferenciador/Integrador: escolha entre um Diferenciador ou Integrador e a introdução do valor da ordem positiva/negativa correspondente;

- Tipo de Aproximação: escolha entre a aproximação de Euler, Tustin e Al-Alaoui;
- Tipo de Gráficos: o utilizador pode escolher o Diagrama de Bode, a Resposta ao Degrau Unitário, o Mapeamento dos Polos&Zeros e, por último, a representação em Step (Degrau).

De seguida, nas Figuras 35 e 36 é possível visualizar a atribuição de cada tipo de representações. No caso seja selecionada a opção Mapeamentos Polos&Zeros surgirá uma nova janela (Figura 36) onde é possível visualizar essa opção. Isso ocorre porque existe uma limitação do GUIDE, em que o tamanho do ambiente de desenvolvimento depende no monitor e da resolução no computador onde essa interface é desenvolvida.

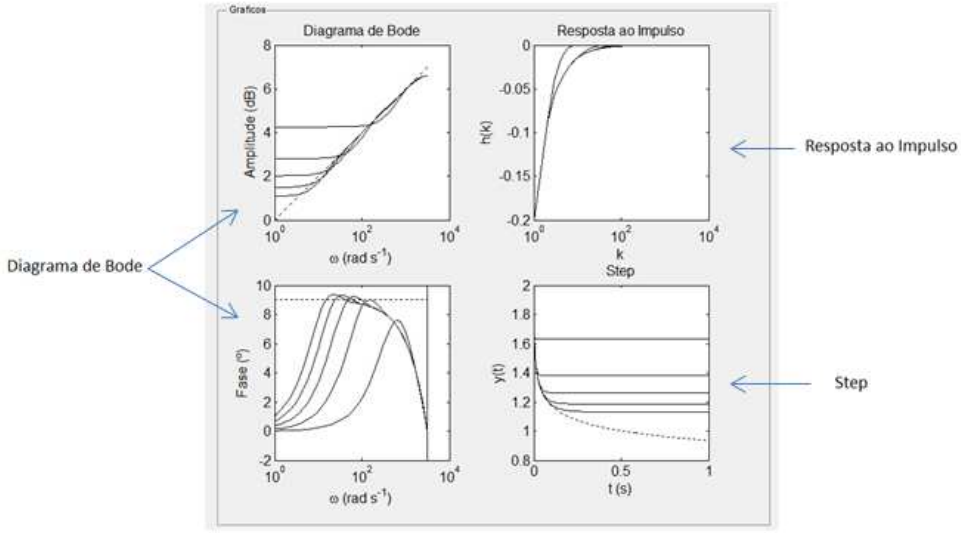

Figura 35: Localização das representações no Filtro IIR

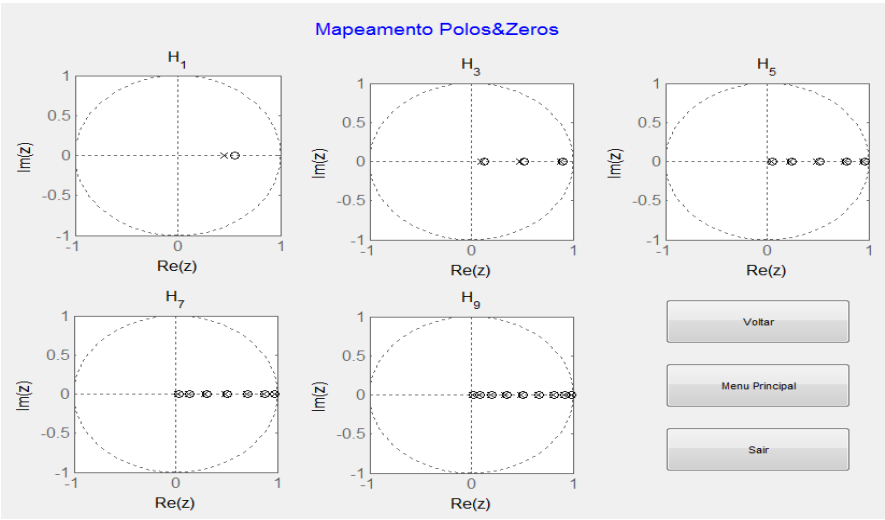

Figura 36: Janela da representação do Mapeamento Polos&Zeros no filtro IIR

Como foi referido anteriormente, pode ser considerado como terceiro bloco principal do menu principal o bloco dos botões ondem pode ser selecionados as seguintes opções:

- Traçar Gráficos: permite a visualização das representações normalizadas;
- Visualizar Equação do Sistema: é possível visualizar e guardar a equação do sistema;
- Menu Principal: retroceder ao menu principal;
- Exit: sair da interface gráfica.

Caso seja escolhido a opção Visualizar Equação, pode ser visto a equação do sistema e ainda existe a possibilidade de guardar a equação num ficheiro de texto. Na Figura 37 é possível ver o aspeto final da opção visualizar equação do sistema.

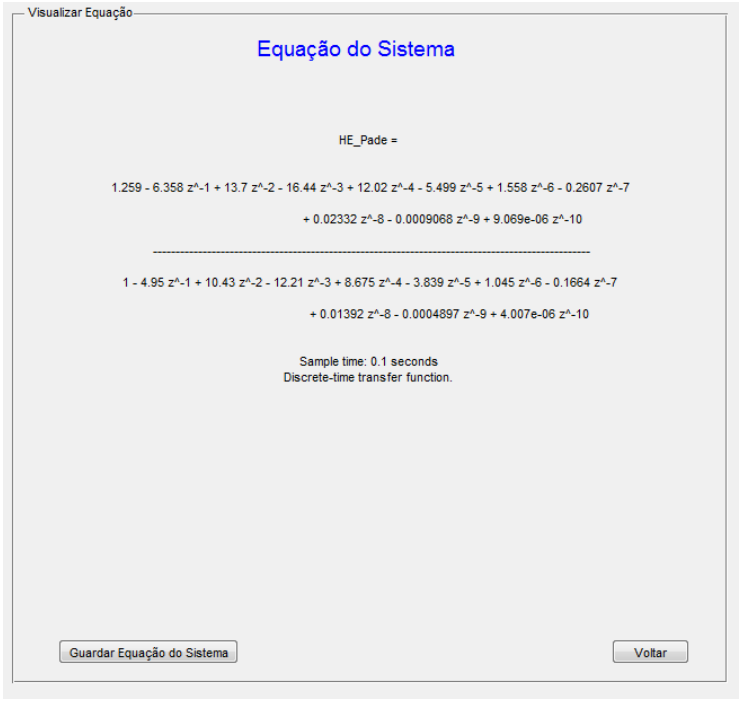

Figura 37: Visualizar a equação na opção IIR

De seguida foi desenvolvido a opção Filtro FIR (Figura 33) apresentada no menu principal. De notar que a opção Filtro FIR é semelhante à opção Filtro IIR, em que as diferenças consistem nos parâmetros de entrada. Na opção Filtro IIR insere-se a ordem do numerador e do denominador e na Opção FIR insere-se o tamanho da amostra pretendida. A outra diferença consiste nos tipos de representações apresentadas que são Mapa de Polos&Zeros, Resposta Impulsional e Resposta em Frequência. Na imagem da Figura 38 é possível visualizar o aspeto final da opção Filtro FIR.

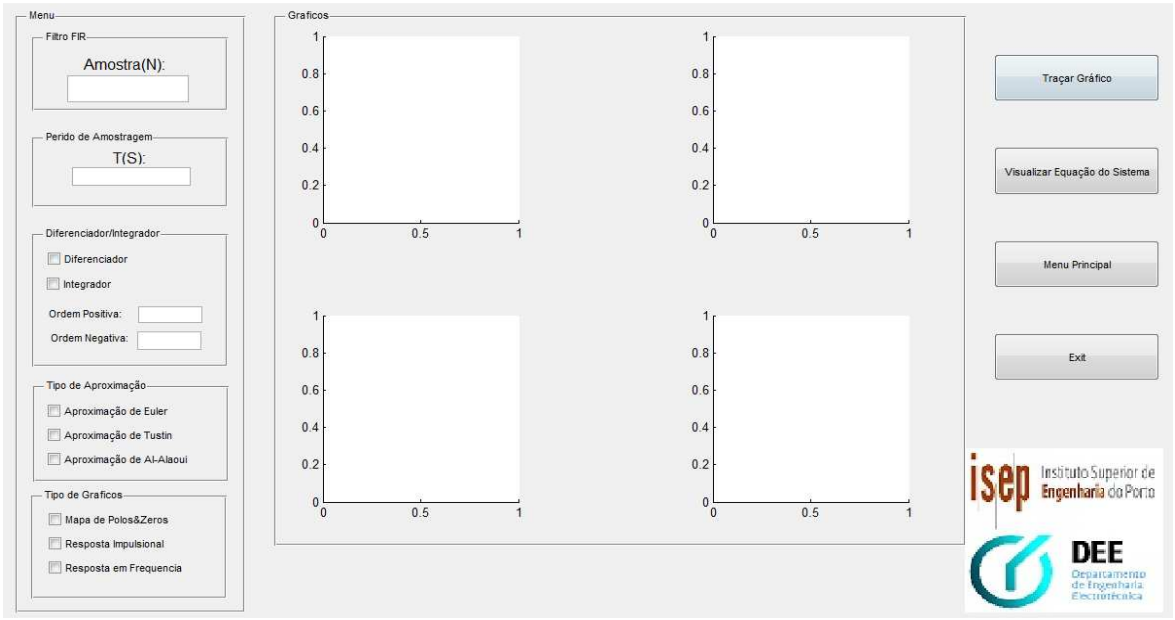

Figura 38:Aspeto final da opção Filtro FIR

A Figura 39 ilustra a localização do tipo de gráfico pretendido e a Figura 40 o aspeto da opção Visualizar Equação do Sistema.

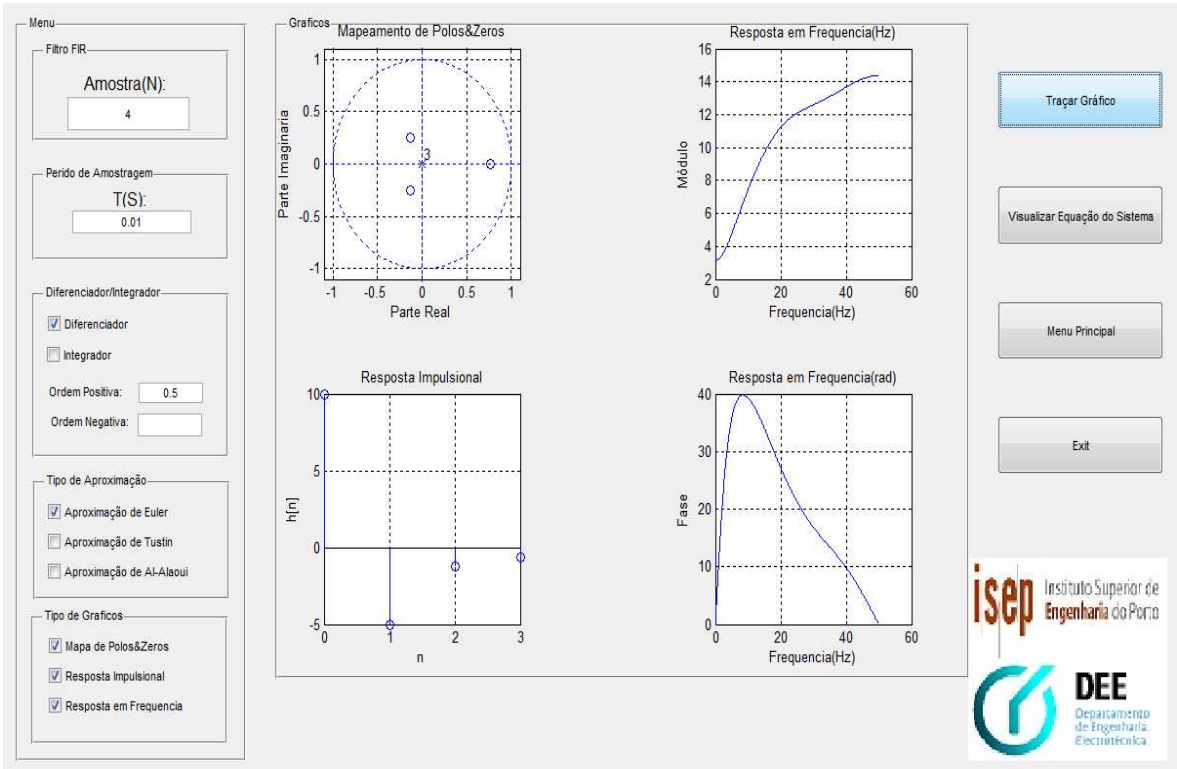

Figura 39: Aspeto final das representações normalizadas do Filtro FIR

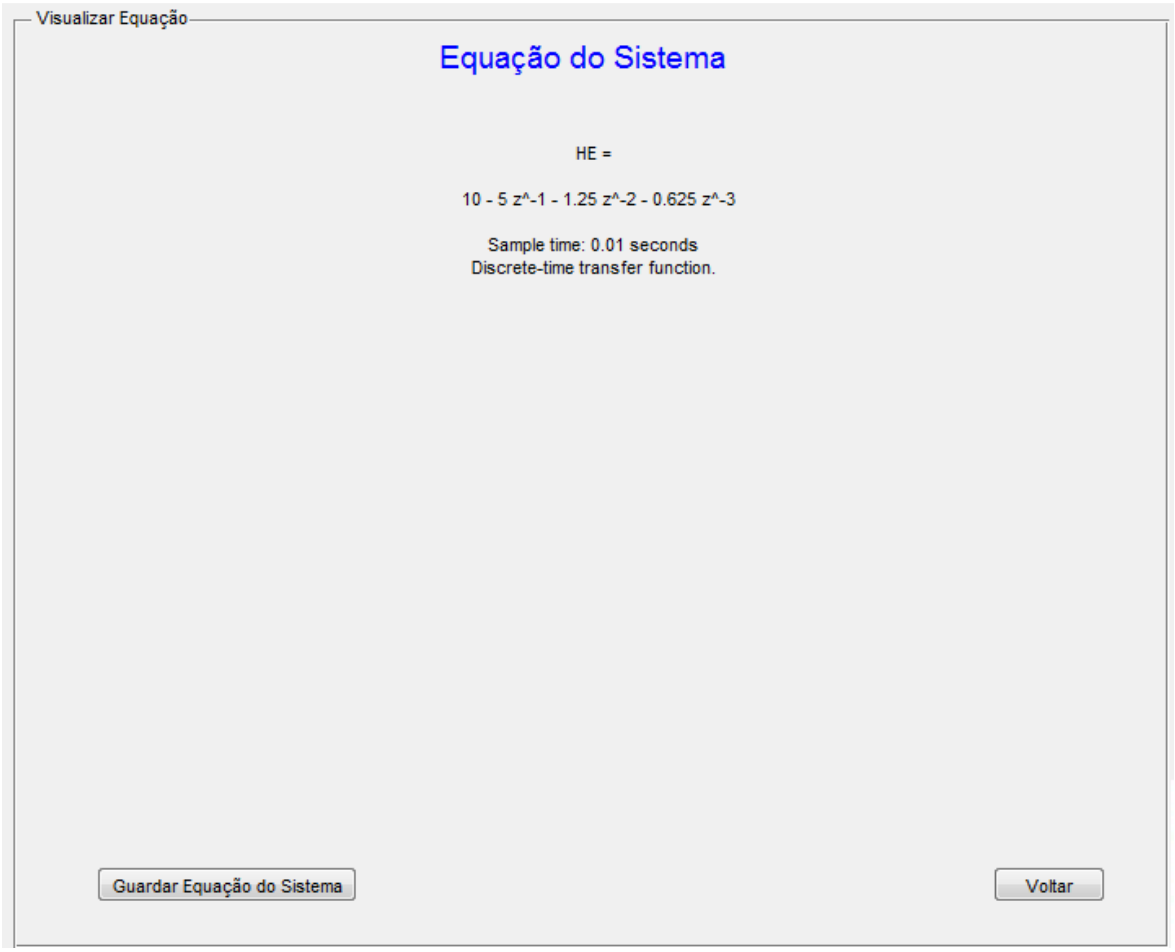

Figura 40: Aspeto final da opção visualizar equação do sistema do Filtro FIR

Em comum com a opção do filtro IIR, existe também na opção FIR quatro botões localizados do lado direito, onde o primeiro permite traçar o gráfico, o segundo visualizar a equação do sistema, o terceiro voltar ao menu principal e o último sair da interface gráfica.

Mas para o desenvolvimento desta interface, ou melhor destas duas fases, a implementação de um filtro FIR e IIR não se baseou apenas no desenho desta interface. Para obter as representações anteriores não basta apenas efetuar o desenho da mesma. A obtenção destes gráficos acontece porque ao efetuar o traçar gráfico e/ou visualizar a equação do sistema existem operações que utilizador não vê.

Inicialmente efetuou-se a programação da interface para traçar os gráficos. Para tal, foi utilizado código previamente desenvolvido para a implementação dos filtros FIR/IIR adaptando este para a interface gráfica.

Na Figura 41 é possível visualizar a função que permite o cálculo da resposta impulsional de Euler. Através dos valores introduzidos (tamanho da amostra (N), período da

amostragem (T) e valor da ordem (OrdP)) esta função devolve a função de transferência de um filtro FIR com uma aproximação de Euler (HE), a amplitude da resposta em frequência (H1) e respetiva fase (F1) e hE os coeficientes do filtro.

```
function [HE,H1,F1,hE] = FIR_EULER(OrdP,N,T)
```

Figura 41:Função do cálculo da resposta impulsional de Euler

Caso o tipo de aproximação escolhido seja a aproximação de Tustin é utilizado outro código para obter as especificações do filtro. A função utilizada pode ser visualizada na Figura 42.

```
function [HT,H2,F2,hT] = FIR_TUSTIN(OrdP,N,T)
```

Figura 42: Função do cálculo da resposta impulsional de Tustin

Na Figura 43 está apresentado a função utilizada para calcular as especificações do filtro com uma aproximação de Al-Alaoui.

```
function [HA,H3,F3,hA] = FIR_Alaoui(OrdP,N,T)
```

Figura 43: Função do código do cálculo da resposta impulsional de Al-Alaoui

Nos parágrafos anteriores foram apresentados as funções utilizadas para calcular as especificações do filtro FIR em função da sua aproximação. Após o cálculo das especificações do filtro pretendido é possível efetuar as representações normalizadas, para tal, como o Matlab possui funções próprias para traçar gráficos, foi conveniente a utilização das mesmas. As funções utilizadas foram as seguintes:

- `[Pfe,Zfe]=pzmap(HE)` – Através desta função obtém-se os polos e os zeros da função de transferência;
- `zplane(Pfe,Zfe)` – Através desta função traça-se a representação dos polos e zeros com um circulo unitário de referência;
- `[Hfe,Tfe]=impz(hE,1)` – A utilização desta função permite o calculo da resposta impulsional;
- `stem(Hfe,Tfe)` – Permite o gráfico da resposta impulsional;
- `plot(F1,abs(H1))` – Faz o gráfico da amplitude da resposta em frequência;
- `plot(F1,unwrap(angle(H1))*180/pi)` – Faz o gráfico da fase da resposta em frequência em graus;

Na Figura 44 é possível visualizar um excerto de código que permite guardar a equação num ficheiro de texto.

```
fir_dif_Euler=fopen('fir_dif_Euler.txt','wt');
fprintf(fir_dif_Euler,'%s',EQ);
fclose(fir_dif_Euler);
```

Figura 44: Extrato de código para guardar a equação num ficheiro de texto

Este excerto corresponde à situação de o utilizador escolher um filtro FIR diferenciador utilizando a aproximação de Euler. Como é possível visualizar pelas seguintes sequências de instruções o método que permite guardar a equação baseia-se nas seguintes funções:

- `fir_dif_Euler=fopen('fir_dif_Euler.txt','wt')` - Esta função vai abrir um ficheiro com o nome *fir_dif_Euler* e este vai ser definido com permissão de escrita;

- `fprintf(fir_dif_Euler,'%s',EQ)` - Com esta instrução é escrito no ficheiro de texto denominado por `fir_dif_Euler` o conteúdo da variável EQ, neste caso corresponde à equação do sistema;

- `fclose(fir_dif_Euler)` – Ao contrário da primeira instrução, esta vai fechar o ficheiro de texto após a escrita no mesmo.

Caso a opção escolhida no Menu Principal seja o Filtro IIR, o código inerente para a implementação do filtro IIR é muito semelhante ao código apresentado anteriormente. Ao contrário do filtro FIR, os coeficientes são obtidos parcialmente através das seguintes funções dependendo do tipo de aproximação:

- `IRR_Euler(OrdP/N,OrdNumerador,OrdDenominador,T)`: Cálculo parcial dos coeficientes utilizando a aproximação de Euler;
- `IRR_Tustin(OrdP/N,OrdNumerador,OrdDenominador,T)`: Cálculo parcial dos coeficientes utilizando a aproximação de Tustin;
- `IRR_Al-ALaoui(OrdP/N,OrdNumerador,OrdDenominador,T)`: Cálculo parcial dos coeficientes utilizando a aproximação de Al-Alaoui.

Após a obtenção parcial dos coeficientes são utilizadas as seguintes funções para calcular os coeficientes do filtro dependendo do método de Padé ou Prony*:*

- `IRR_Pade(hE/hT/ha,T)`: Função que permite o cálculo dos coeficientes utilizando o método de Padé dependendo do tipo de aproximação;

- `IRR_Prony(hE/hT/ha,T ):` Função que permite o cálculo dos coeficientes utilizando o método de Prony dependendo do tipo de aproximação.

O código utilizado para implementar as restantes funcionalidades da interface (representações normalizas e visualizar/guardar equação do sistema) é idêntico à opção do Filtro FIR.

Na Figura 33 é possível verificar que existe além da opção Filtro FIR/IIR a opção estruturas. Quando escolhida a opção Estruturas no Menu Principal surgirá uma nova janela onde é possível escolher qual o tipo de filtro pretendido. Na Figura 45 é possível visualizar essa nova janela.

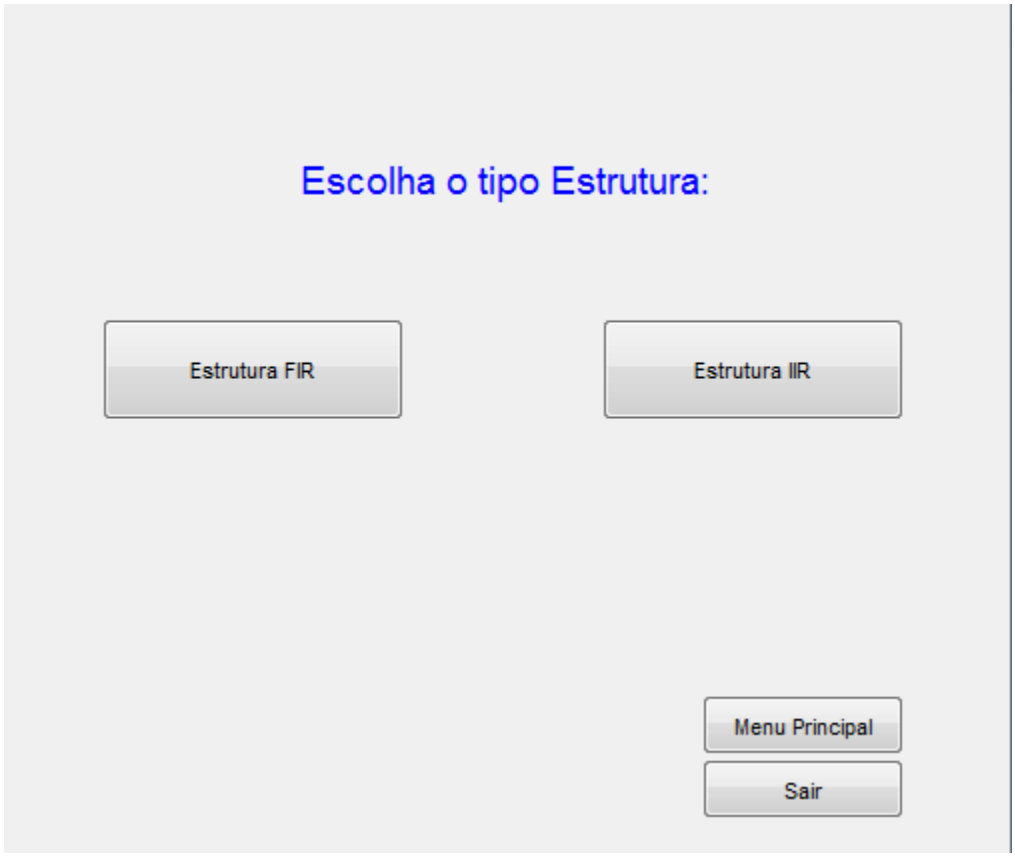

Figura 45: Aspeto final da opção escolher tipo de estrutura FIR/IIR

Como passo seguinte no desenvolvimento da interface gráfica foi a realização das opções Estrutura FIR e Estrutura IIR.

Inicialmente procedeu-se ao desenvolvimento da interface para uma estrutura IIR e posteriormente para uma estrutura FIR. Como primeiro passo para a opção Estrutura IIR ou FIR procedeu-se ao desenho da própria interface como mostrado nas Figuras 56 e 57.

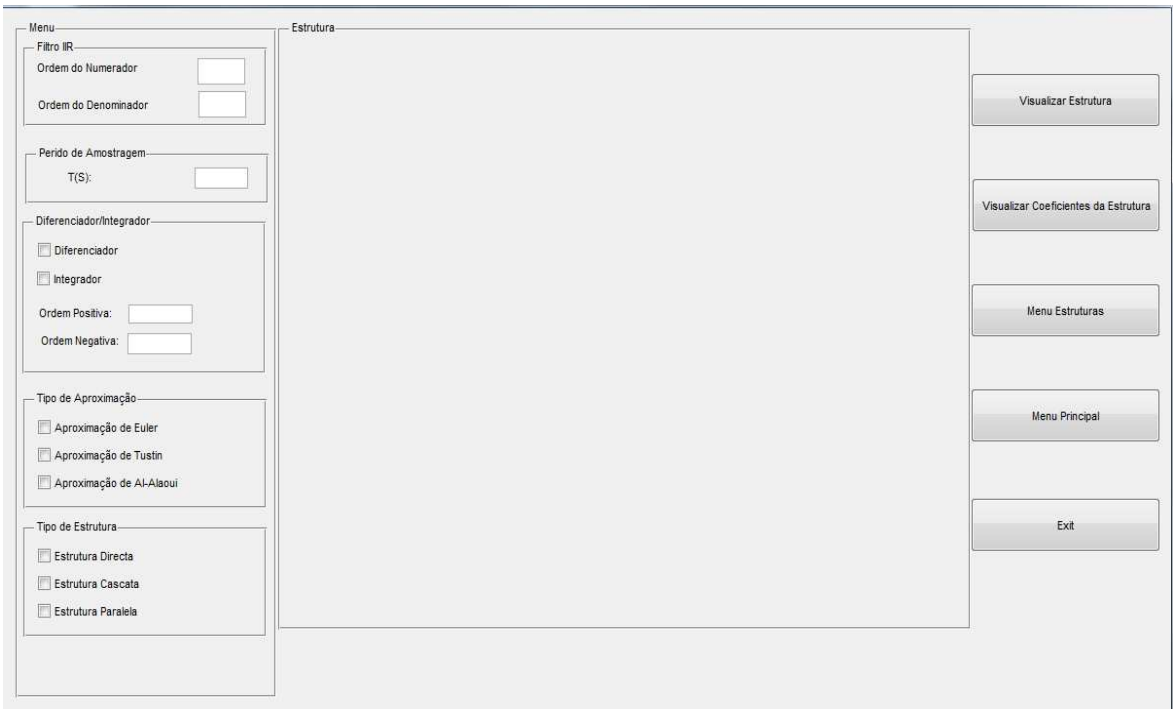

Figura 46: Aspeto final da interface Estrutura IIR

Figura 47: Aspeto final da interface Estrutura FIR

No caso da escolha da estrutura ser o do filtro IIR, o funcionamento da interface e o código inerente será o mesmo da interface gráfica Filtro IIR só que em vez de efetuar as

representações normalizadas será apresentado a estrutura do filtro pretendido. Na Figura 48 é possível visualizar um exemplo do aspeto final de uma estrutura em cascata para:

- Ordem do Numerador = Ordem do Denominador=8;
- Período de Amostragem=0.01;
- Diferenciador com uma ordem α=0.5;
- Aproximação de Euler;
- Estrutura em cascata.

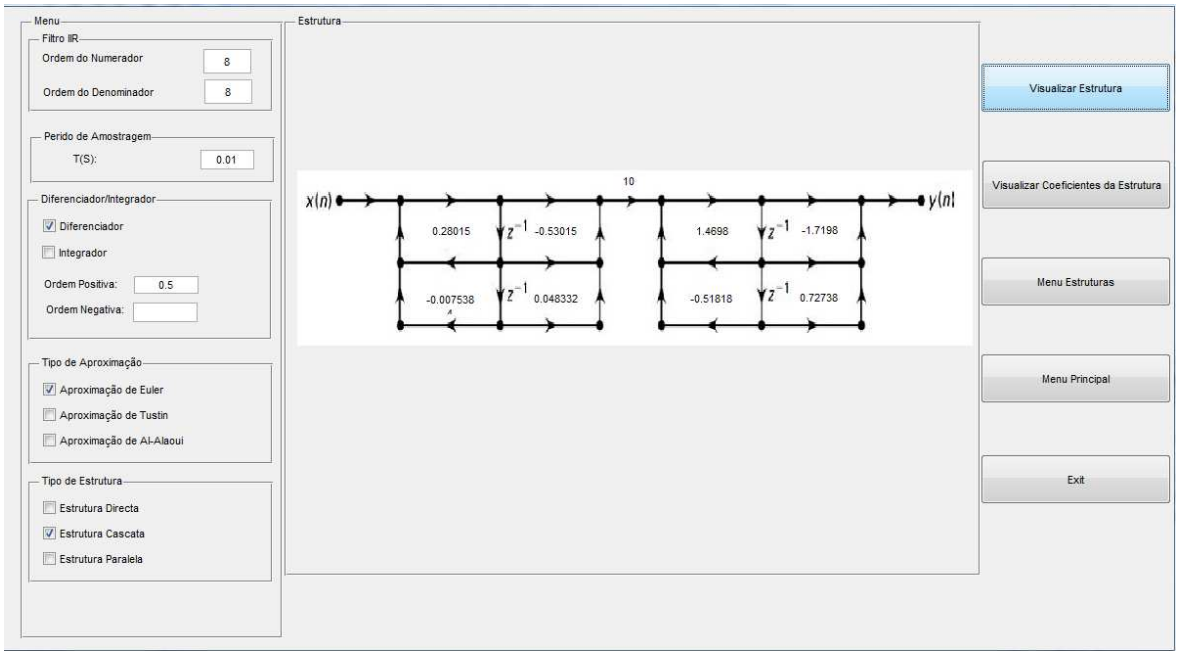

Figura 48: Exemplo de estrutura cascata Filtro IIR

Inicialmente é efetuado o cálculo do filtro IIR segundo as especificações introduzidas (utilizando unicamente o método de Padé) e posteriormente é calculado os coeficientes para cada tipo de estrutura através das seguintes funções:

- Estrutura direta utiliza os coeficientes calculados;
- Estrutura cascata utiliza a função: `dir2cas(b,a)`;
- Estrutura paralelo utiliza a seguinte função: `dir2par(b,a)`.

Os paramentos *b* e *a* são os valores do coeficientes calculados previamente devolvendo as funções os coeficientes correspondentes ao tipo de estrutura. Estas funções encontram-se no Anexo A e foram baseadas do livro *Digital Signal Processing using Matlab* do Vinay K. Ingle e John G. Proakis [6].

No caso de o utilizador selecionar a opção visualizar coeficientes da estrutura surgirá uma nova janela onde será possível ver os coeficientes da estrutura. Na Figura 49 é possível visualizar o exemplo anterior mas agora escolhendo a opção visualizar coeficientes.

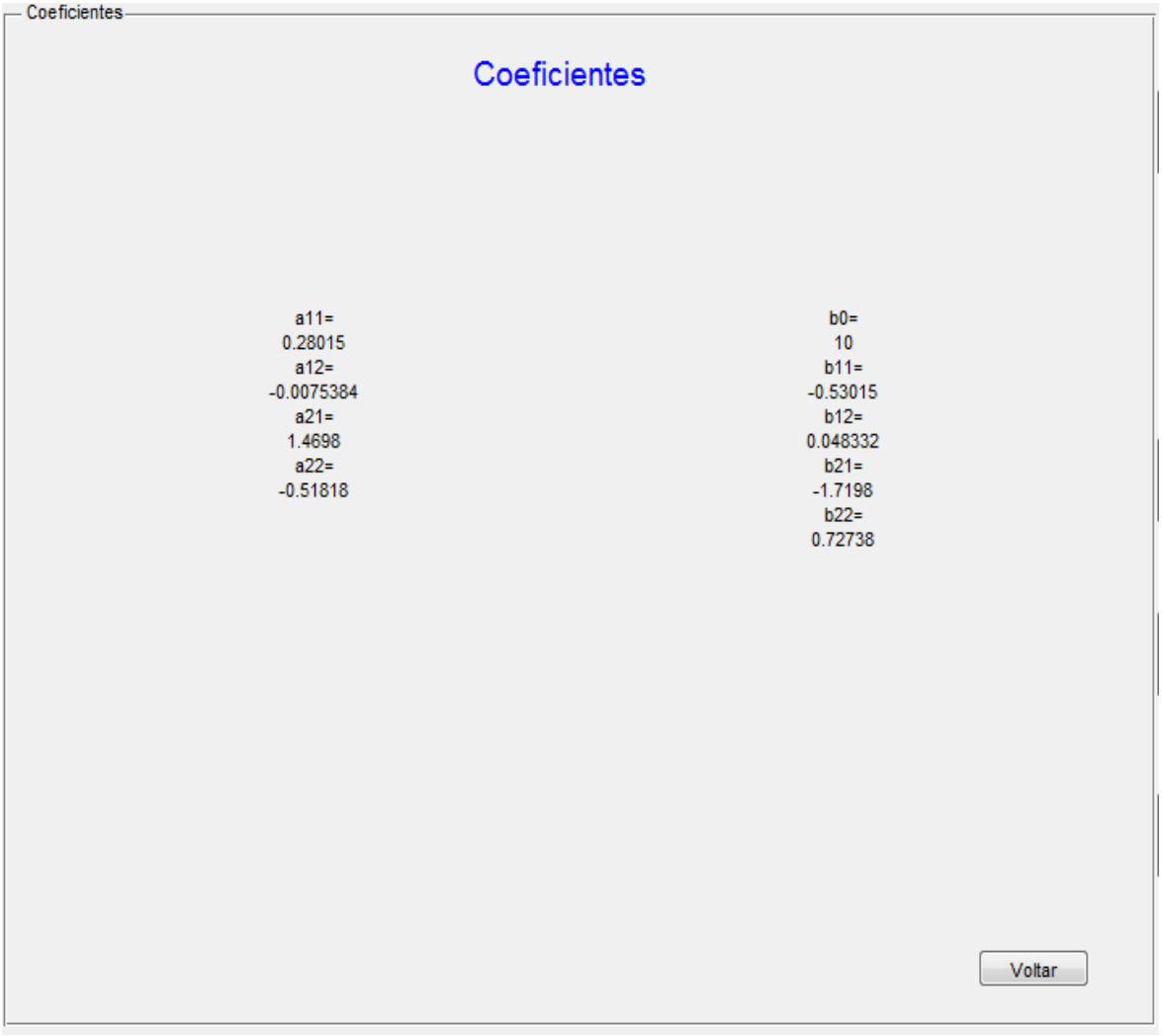

Figura 49: Aspeto final da opção visualizar coeficientes de um filtro IIR

No caso de o utilizador escolher a opção estrutura FIR, o funcionamento é idêntico ao funcionamento da opção Filtro FIR. As principais diferenças são que nesta interface não vai ser permitido o utilizador escolher qual o tamanho de amostra, o tipo de estrutura substitui o tipo de gráficos e a opção visualizar equação do sistema é substituída pela opção visualizar coeficientes do sistema.

Nas Figuras 50 e 51 é possível visualizar dois exemplos: uma estrutura direta de um filtro FIR (Figura 50) e uma estrutura cascata de um filtro FIR (Figura 51) caso selecione a opção Visualizar Estrutura.

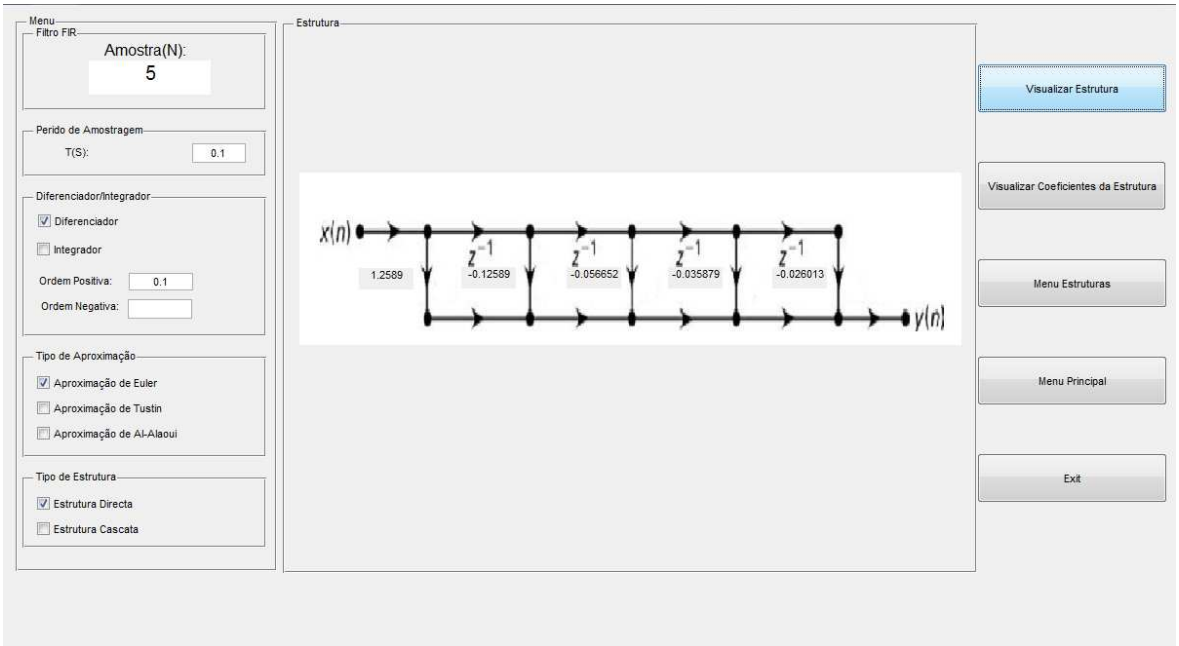

Figura 50: Exemplo de estrutura direta filtro FIR

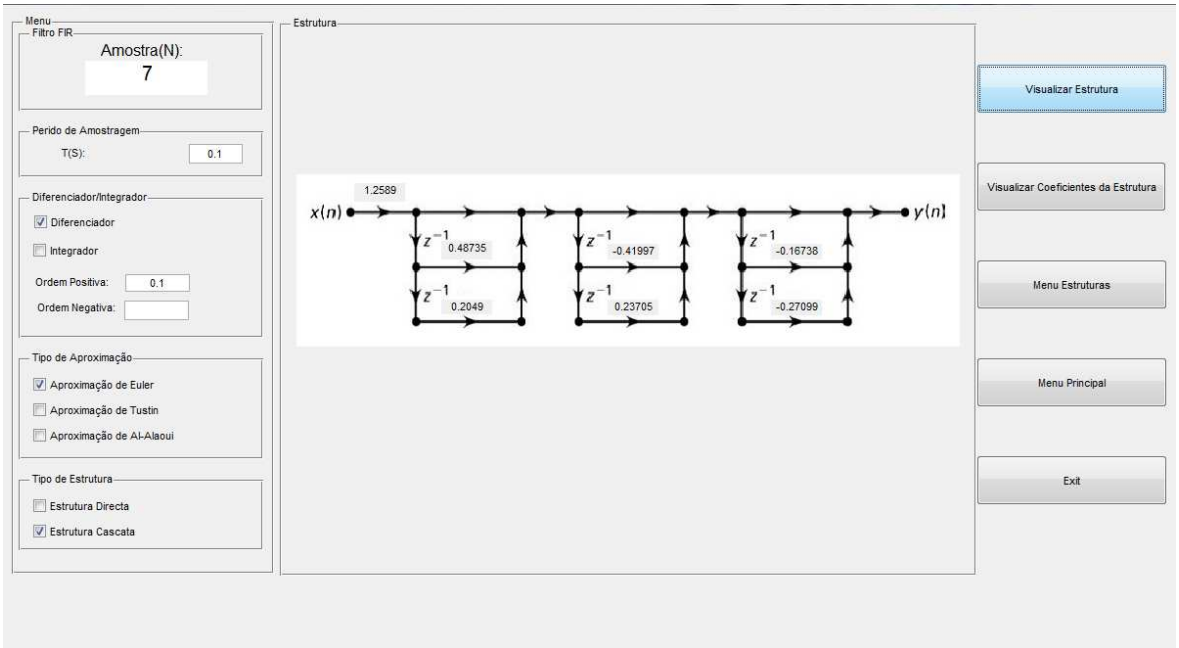

Figura 51: Exemplo de estrutura cascata filtro FIR

Os exemplos apresentados anteriormente foram desenvolvidos considerando os seguintes valores introduzidos:

- Período de Amostragem T=0.1 s;
- Diferenciador;
- Ordem Positiva: 0.1;

- Aproximação de Euler;
- Amostra (N) : 5 (direta) e 7 (cascata).

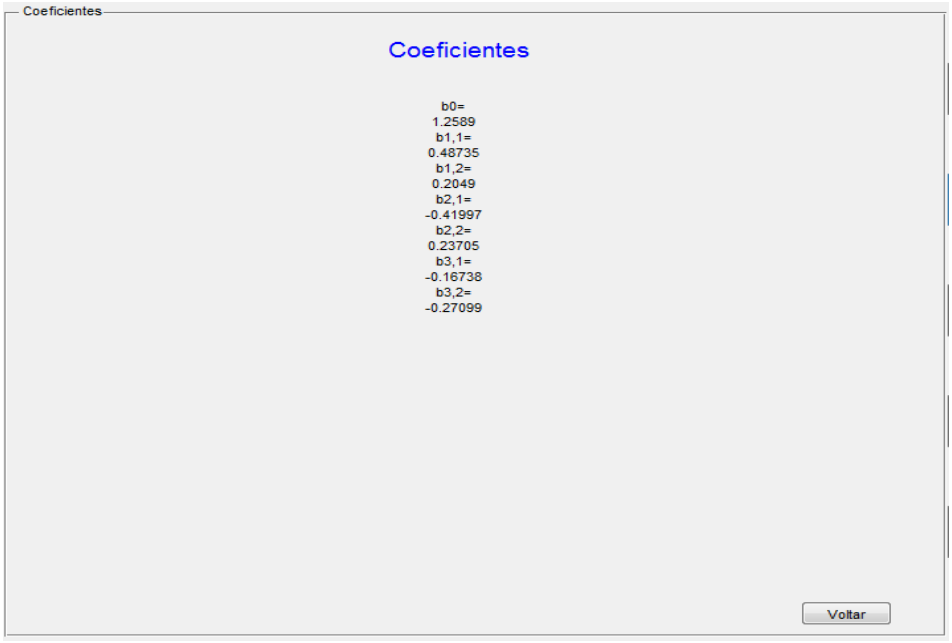

Figura 52: Aspeto final da opção visualizar coeficientes de um filtro FIR

Se após compilar o sistema (carregar na opção Visualizar Estrutura) pressionar o botão Visualizar Coeficientes da Estrutura surgirá um novo painel (Coeficientes). Na Figura 52 é possível visualizar os coeficientes no seu aspeto final.

O funcionamento do programa desenvolvido para a opção visualizar a estrutura é idêntico ao funcionamento da mesma opção na estrutura IIR. No entanto, o cálculo dos coeficientes de um filtro IIR é diferente. Na opção IIR inicialmente é calculado parcialmente os coeficientes através da função `IRR_EULER` e depois através na função `IRR_Pade`. No caso de um Filtro FIR os coeficientes são obtidos unicamente através da função `FIR_EULER`.

Como já foi referido anteriormente é possível escolher dois tipos de estruturas. Caso a opção escolhida seja a estrutura direta através da leitura dos valores inseridos no vetor `hE` podemos obter os coeficientes e imprimir na figura da estrutura direta. No caso de escolher uma estrutura cascata, é utilizado a função `dir2cas` para converter os coeficientes de uma estrutura direta para cascata [6]. Nesta função `dir2cas(hE,1)` os parâmetros de entrada do numerador assume o valor `hE` e o parâmetro do denominador assume o valor 1 porque a função de transferência de um Filtro FIR assume apenas valores no numerador.

No caso de uma estrutura em paralelo no caso de um filtro IIR a função utilizada para efetuar essa conversão é `dir2par`, como no caso da função cascata para o filtro IIR.

3.2.2. EXEMPLOS ILUSTRATIVOS

Nesta secção vão ser apresentados alguns exemplos ilustrativos de filtros FIR e IIR, e suas estruturas com diferentes tipos de implementação.

3.2.2.1. FILTRO FIR

Aqui são apresentados dois exemplos ilustrativos da interface gráfica desenvolvida. Neste caso a opção escolhida vai ser o filtro FIR, onde nas Figuras 53 e 54 é possível visualizar a implementação de um diferenciador e integrador, respectivamente. Os parâmetros de entrada foram:

- N=10;
- T=0.000125 s;
- α=±45;
- Aproximação de Euler.

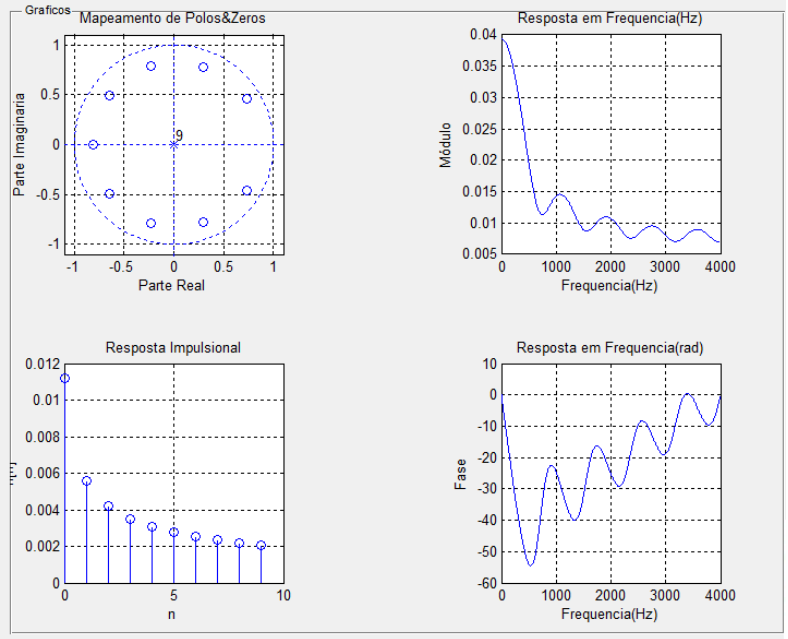

Figura 53: Respostas do filtro FIR, N=10, T=0.000125 s, α=-0.5 e aproximação de Euler (Integrador)

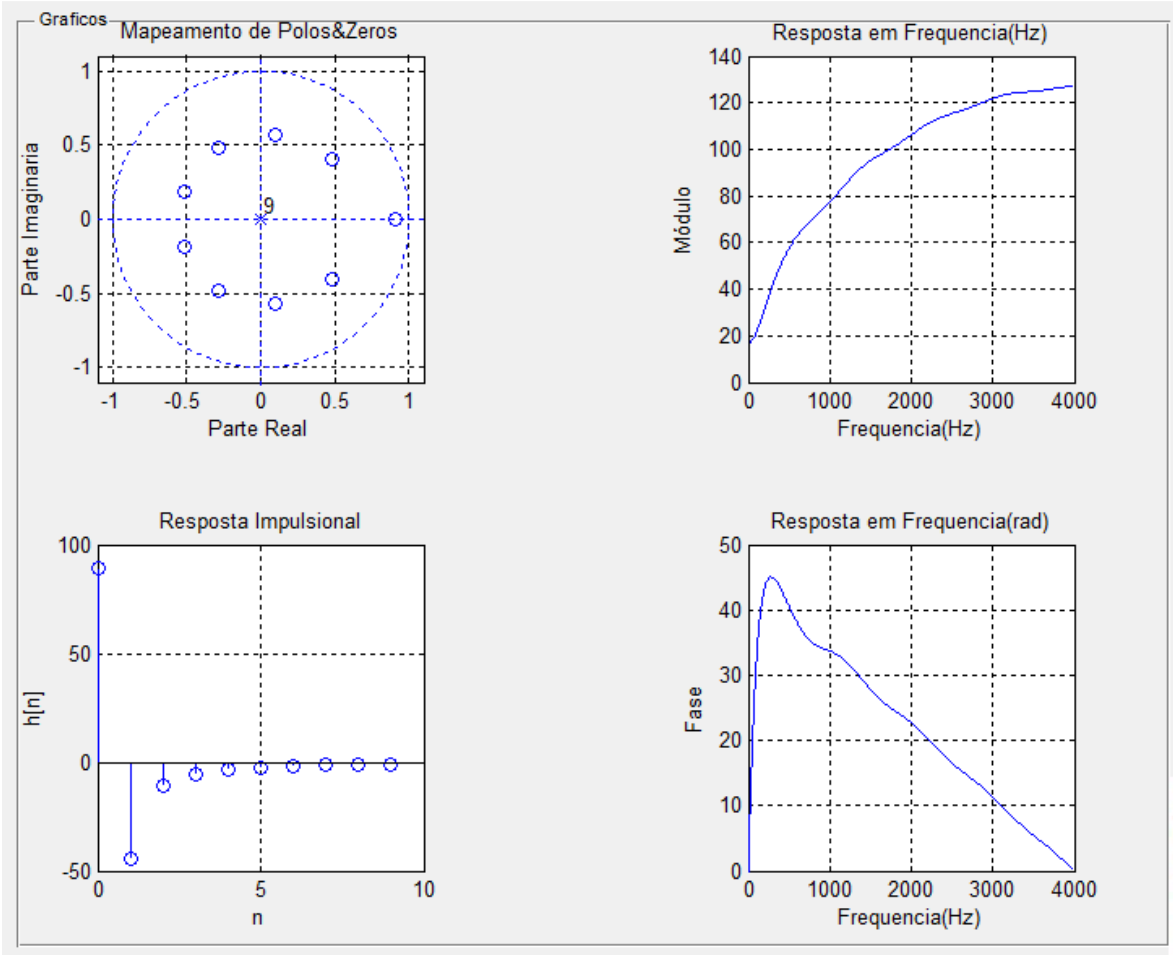

Figura 54: Respostas do filtro FIR, N=10, T=0.000125 s, α=0.5 e aproximação de Euler (Diferenciador)

As equações do sistema para cada um dos exemplos anteriores estão representados nas Figuras 55 e 56.

Equação do Sistema

HE =

0.01118 + 0.00559 z^-1 + 0.004193 z^-2 + 0.003494 z^-3 + 0.003057 z^-4 + 0.002751 z^-5

+ 0.002522 z^-6 + 0.002342 z^-7 + 0.002196 z^-8 + 0.002074 z^-9

Figura 55: Equação do filtro FIR, N=10, T=0.000125 s, α=-0.5 e aproximação de Euler (Integrador)

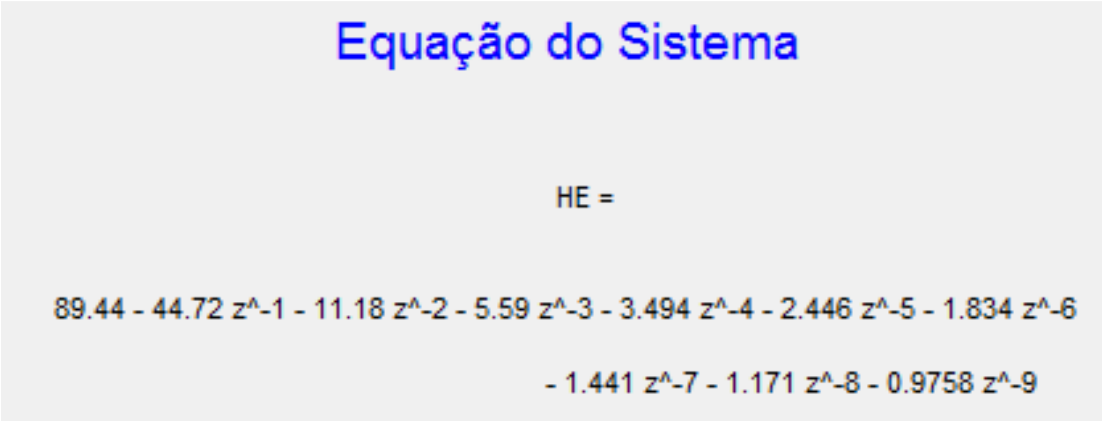

Figura 56: Equação do filtro FIR, N=10, T=0.000125s, α=-0.5 e aproximação de Euler (Diferenciador)

Como é possível visualizar nas Figuras 53 e 54 a interface gráfica possibilita a análise da resposta em frequência (amplitude e fase) do filtro quanto à amplitude e a fase e a sua resposta impulsional. Estas respostas vão prestar um grande auxílio para a implementação prática, visto que através das respostas obtidas vai ser possível confirmar se o filtro encontra-se bem implementado.

3.2.2.2. ESTRUTURA DE UM FILTRO FIR

No caso da opção escolhida no Menu Principal ser a opção Estruturas da opção filtro FIR vai ser possível visualizar a estrutura e os seus coeficientes dependendo dos valores introduzidos. Os parâmetros de entrada são: T=0.000125 s, e α=±0.5 e aproximação de Euler.

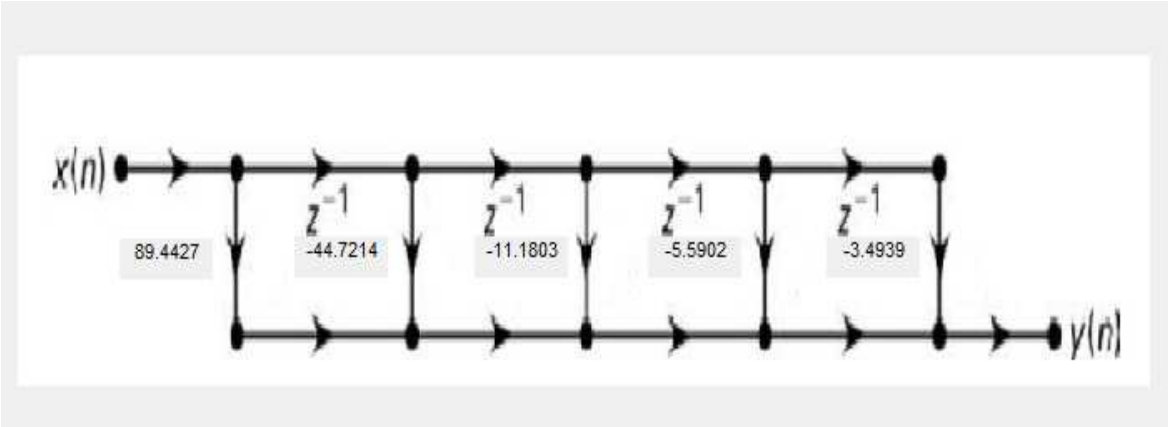

Figura 57: Estrutura direta de um filtro FIR, N=5, T=0.000125 s, α=0.5 e aproximação de Euler (Diferenciador)

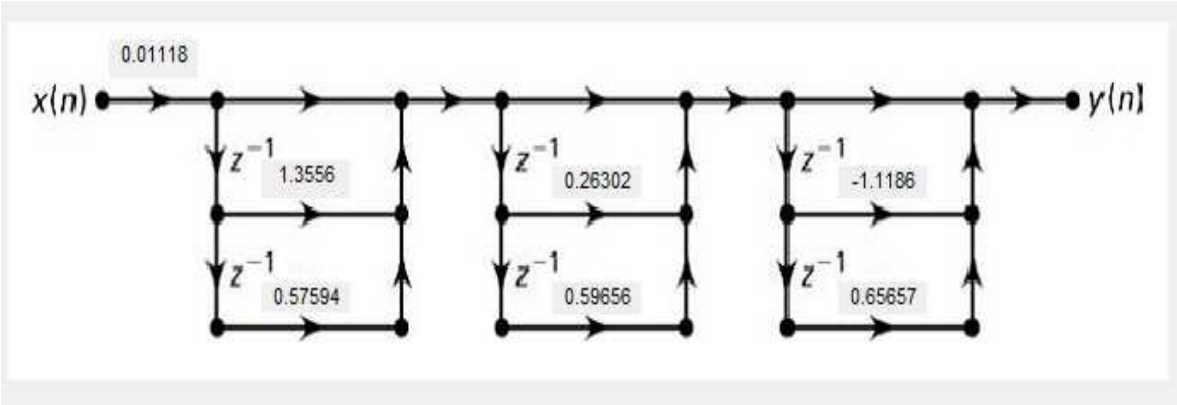

Figura 58: Estrutura cascata de um filtro FIR, N=7, T=0.000125 s , α=-0.5 e aproximação de Euler (Integrador)

Figura 59: a) Coeficientes da estrutura direta, filtro FIR; b) Coeficientes da estrutura cascata, filtro FIR

Nas Figuras 57 e 58 é possível visualizar a estrutura direta e a estrutura cascata para um filtro FIR. Esta interface gráfica é de bastante utilidade porque permite visualizar a estrutura pretendida e os seus coeficientes. A sua principal limitação devido ao filtro em questão ser um FIR, e ser necessário *n* figuras para os diferentes tamanhos de amostras. Por esta razão, como foi referido anteriormente, as amostras escolhidas são N=5 para a estrutura direta e N=7 para a estrutura cascata. Caso seja necessário implementar uma estrutura com valores diferentes, basta editar as funções apresentadas em anexo e efetuar o cálculo dos seus coeficientes.

3.2.2.3. FILTRO IIR

Como na opção anterior do filtro FIR (ponto 3.2.2.1) vai ser apresentado aqui um exemplo ilustrativo sobre a implementação de um filtro IIR.

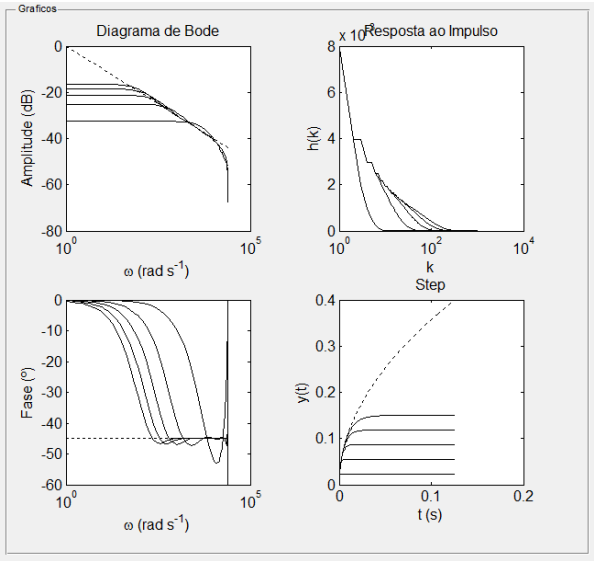

Figura 60: Respostas do filtro IIR, OrdN=OrdD=1,3,5,7,9, T=0.000125 s, α=-0.5 e aproximação de Tustin (Integrador)

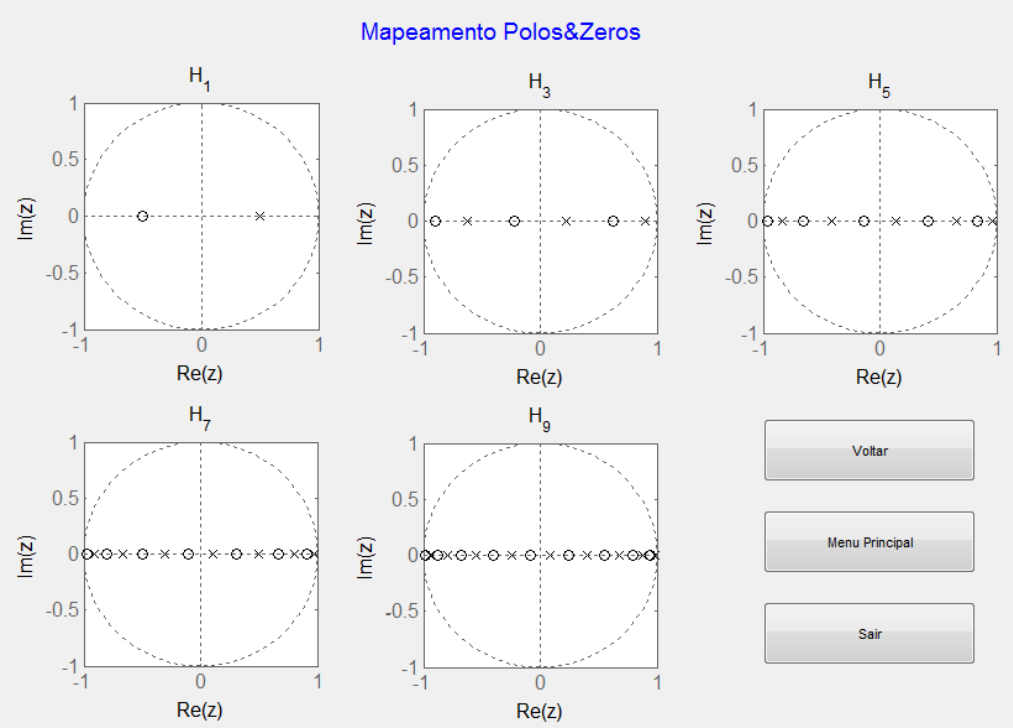

Figura 61: Respostas do filtro IIR, OrdN=OrdD=1,3,5,7,9, T=0.000125 s, α=-0.5 e aproximação de Tustin (Integrador)

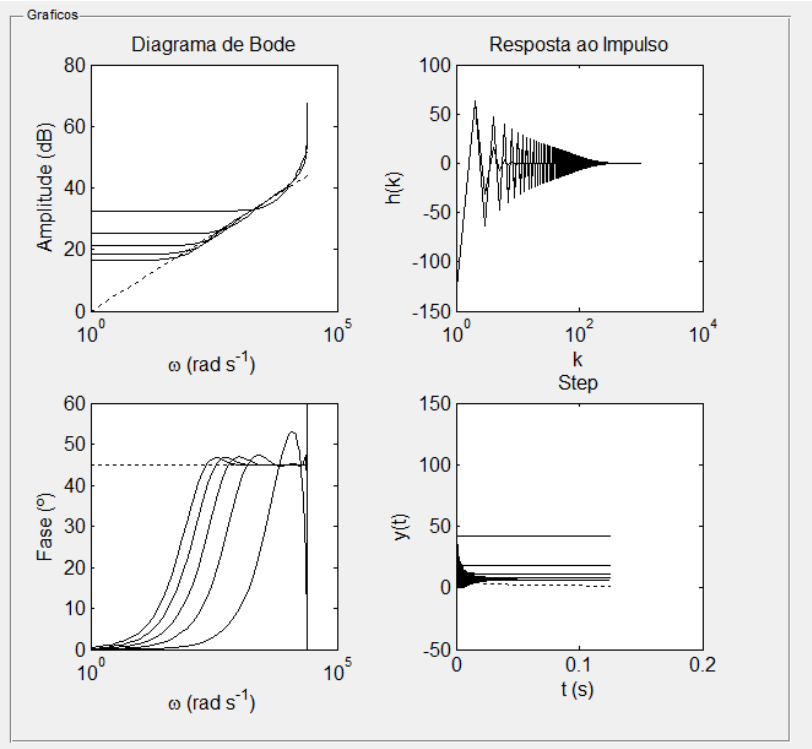

Figura 62: Respostas do filtro IIR, OrdN=OrdD=1,3,5,7,9, T=0.000125 s, α=0.5 e aproximação de Tustin (Diferenciador)

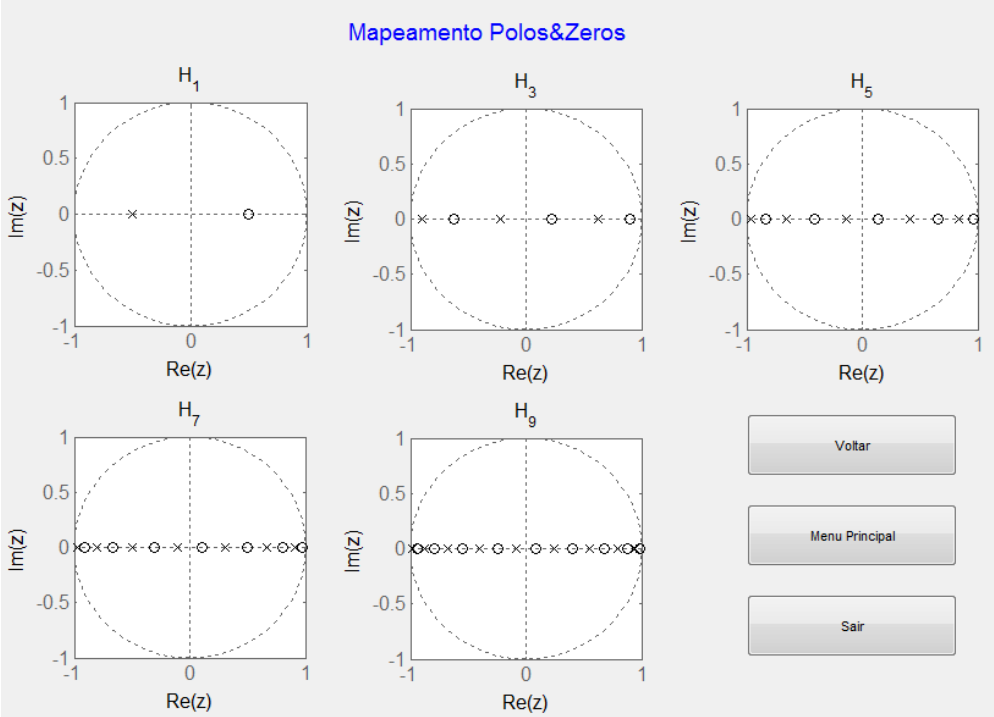

Figura 63: Respostas do filtro IIR, OrdN=OrdD=1,3,5,7,9, T=0.000125 s, α=0.5 e aproximação de Tustin (Diferenciador)

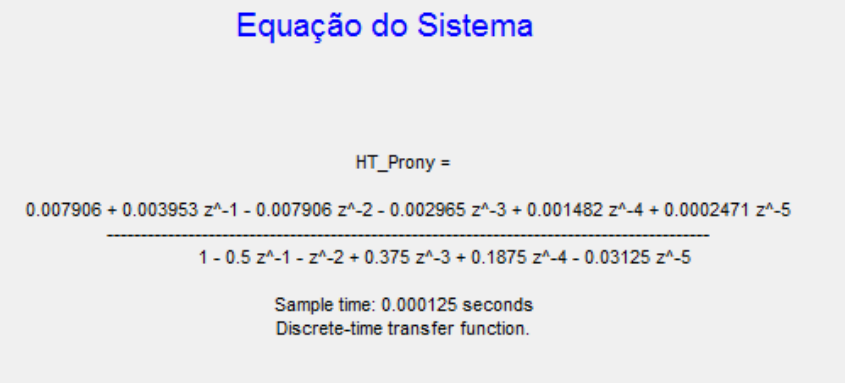

Figura 64: Equação do filtro IIR, OrdN=OrdD=5, T=0.000125s, α=-0.5 e aproximação de Tustin (Integrador)

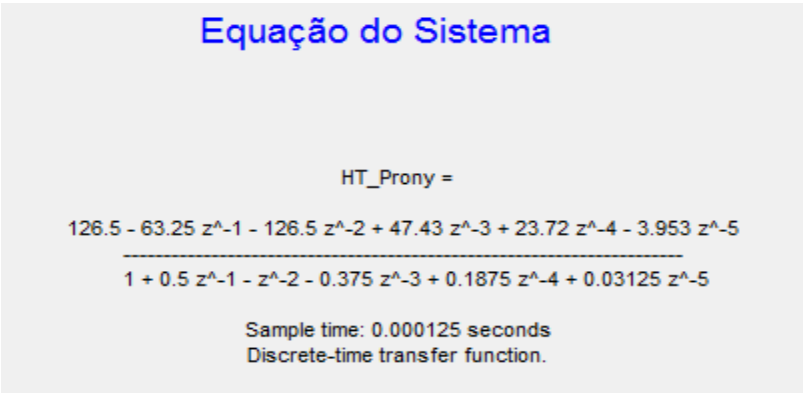

Figura 65: Equação do filtro IIR, OrdN=OrdD=5, T=0.000125 s, α=0.5 e aproximação de Tustin (Diferenciador)

Nas Figuras 60, 61, 62, 63, 64 e 65os parâmetros de entradas escolhidos foram:

- Ordem Numerador=Denominador=5;
- T=0.000125 s;
- Coeficientes obtidos através do método de Prony;
- α=±0.5;
- Aproximação de Tustin.

Através das respostas obtidas anteriormente e em comum com o filtro FIR é possível analisar as respostas para o filtro pretendido e comparar/confirmar os valores simulados com os obtidos experimentalmente.

3.2.2.4. ESTRUTURA DE UM FILTRO IIR

Em comum com a estrutura do filtro FIR vai ser apresentado de seguida um exemplo da interface gráfica de uma estrutura de um filtro IIR.

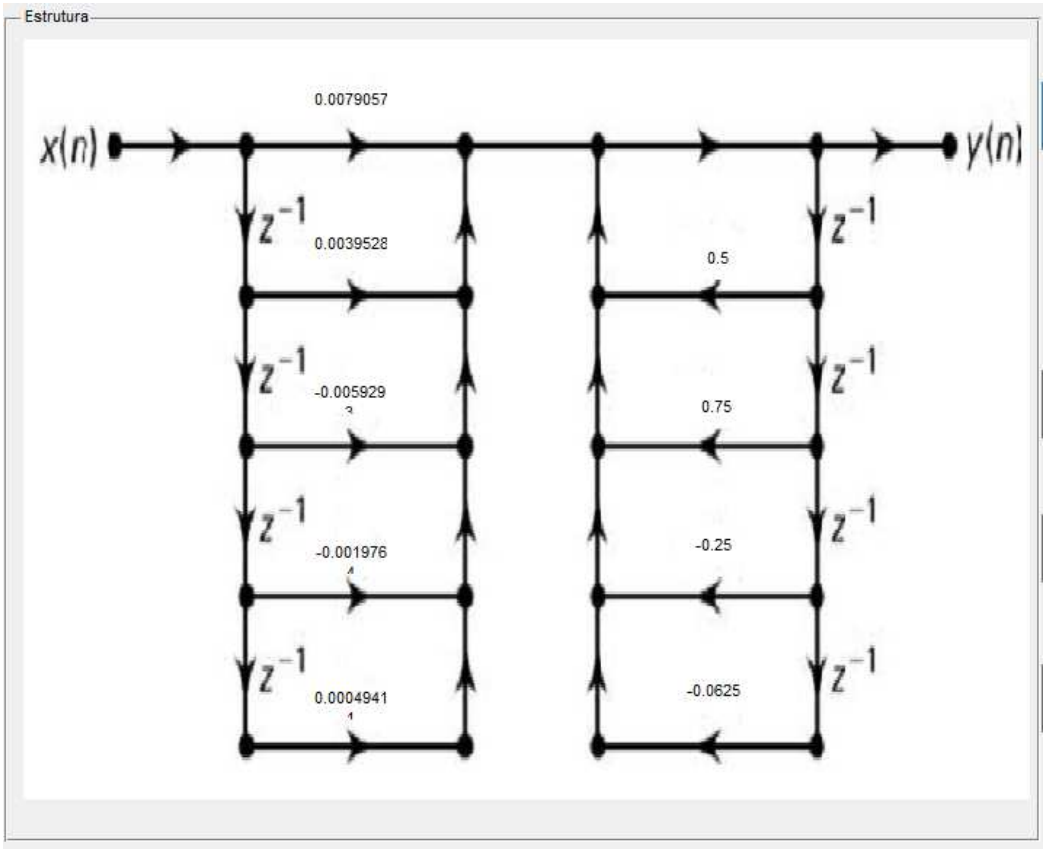

Figura 66: Estrutura direta de um filtro IIR, OrdN=OrdD=4, T=0.000125 s, α=-0.5 e aproximação de Tustin (Integrador)

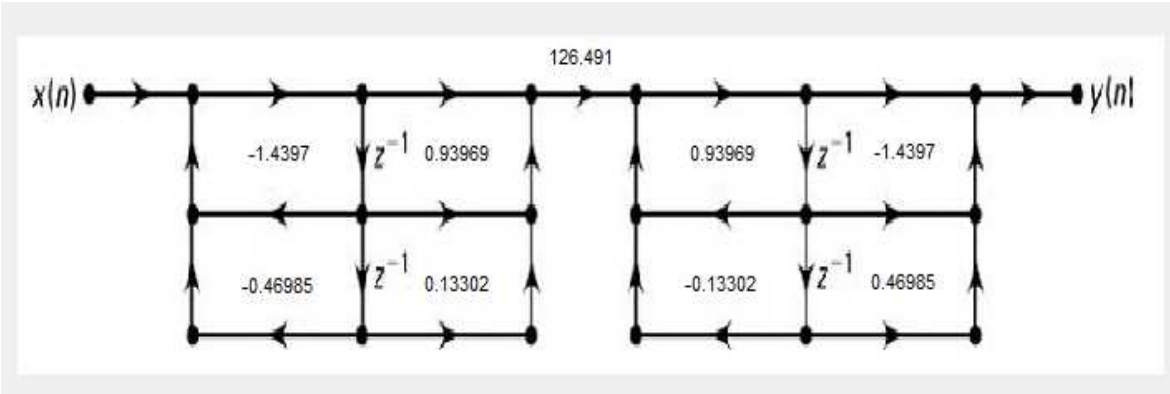

Figura 67: Estrutura cascata de um filtro IIR, OrdN=OrdD=4, T=0.000125 s, α=0.5 e aproximação de Tustin (Diferenciador)

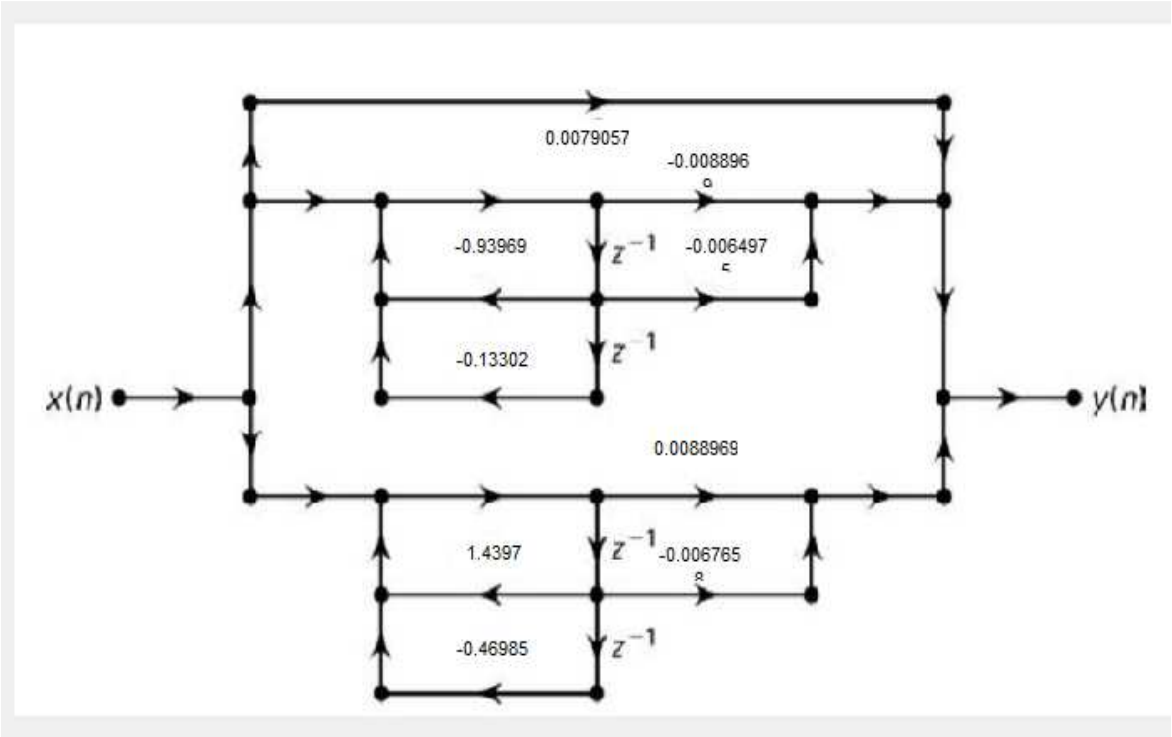

Figura 68: Estrutura paralelo de um filtro IIR, OrdN=OrdD=4, T=0.000125 s, α=0.5 e aproximação de Tustin (Integrador)

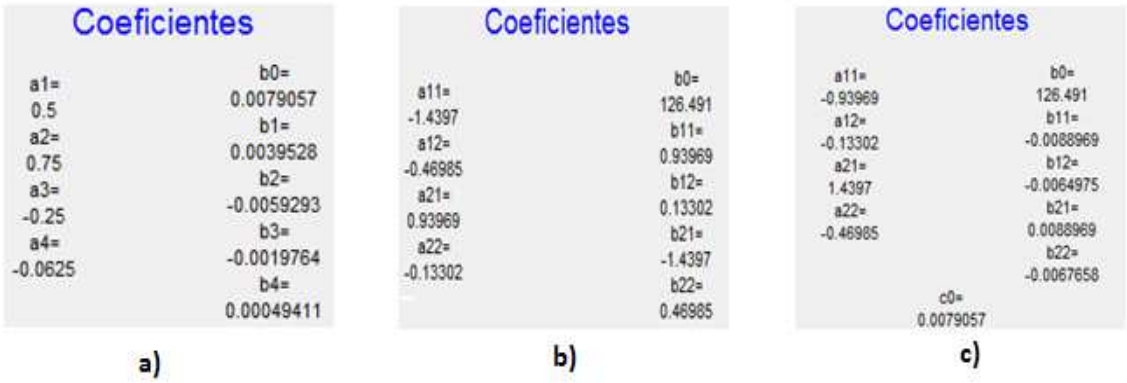

Figura 69: a) Coeficientes da estrutura direta, filtro IIR; b) Coeficientes da estrutura cascata, filtro IIR; c) Coeficientes da estrutura paralela, filtro IIR

Nas Figuras 66-68 é apresentado um exemplo das diversas estruturas possíveis para um filtro IIR. Como é possível visualizar as estruturas obtidas através da interface gráfica podem vir a dar um grande auxílio na implementação de filtros digitais IIR. Neste caso, e ao contrário da estrutura do filtro FIR, esta interface não possui a limitação de escolher apenas duas ordens, sendo possível selecionar qual a ordem do numerador e do denominador.

4. IMPLEMENTAÇÃO EM DSP

Neste capítulo vão ser apresentadas as ferramentas necessárias para implementar filtros digitais fraccionários em DSP. Inicialmente é efetuado uma introdução dos equipamentos escolhidos, e a seguir são apresentados os algoritmos desenvolvidos das implementações práticas. Por fim são ilustrados alguns exemplos.

4.1. ADSP 2181

A implementação dos filtros IIR e FIR fraccionários em estudo neste projeto utilizou a placa ADSP-2181-Kit Analog Devices (Ez-kit Lite), conforme apresentado na Figura 70. Este kit tem incorporado o ADSP-2181, onde serão implementados os filtros fraccionários.

O ADSP-2181 é um microcomputador de um único *chip* otimizado para processamento digital de sinal (DSP) de alta velocidade e aplicações numéricas de processamento.

Esta placa combina a arquitetura base da família ADSP-2100 (três unidades computacionais, geradores de dados de endereços e um sequenciador de programa) com portas séries, uma porta interna DMA de 16 bits, um *timer* programável, *flag I/O*, possibilidade de interrupção e um *chip* de memória [22][24][25].

Figura 70: Ez-Kit Lite - ADSP-2181[37][38]

Para a implementação dos filtros foi também necessário utilizar um gerador de sinais e um ScopeMeter. A necessidade da utilização de um gerador de sinais advém do objetivo de que a implementação dos filtros digitais fraccionários seja em tempo real. O gerador de sinais utilizado foi o Topward 8110 (Figura 71).

Figura 71: Gerador de sinais TopWard 8110 [39]

Além do gerador de sinais apresentado na Figura 71 houve a necessidade de utilizar um ScopeMeter. Este tem um papel também fundamental no projeto porque sem ele era impossível analisar as respostas obtidas e comparar os resultados práticos com os

resultados obtidos por simulação. Na Figura 72 é possível visualizar o ScopeMeter utilizado.

Figura 72: ScopeMeter-Fluke 123 [40]

A interligação do ADSP, gerador de sinais e ScopeMeter é muito simples. No *drive* de entrada do ADSP é efetuado a ligação entre o gerador de sinais e o ADSP e na *drive* de saída é ligado o Scope de modo a poder visualizar as respostas de saída do DSP.

Apos a análise e estudo dos três equipamentos anteriormente referidos, procedeu-se à implementação do código necessário para a implementação dos filtros.

4.2. IMPLEMENTAÇÃO DE FILTROS DIGITAIS EM DSP

Como foi referido no final da secção anterior, para implementar os filtros digitais em DSP foi necessário implementar os algoritmos correspondentes a cada filtro.

Inicialmente procedeu-se à implementação do algoritmo para o filtro FIR e posteriormente para o filtro IIR.

4.2.1. ALGORITMO DO FILTRO FIR

A implementação do algoritmo de um filtro FIR consistiu numa fase muito complexa apesar da sua aparente simplicidade. Isto ocorre visto não existir conhecimento prévio sobre a implementação do mesmo em DSP.

Após o estudo e a análise das capacidades do ADSP passou-se à implementação do algoritmo pretendido. No manual do fabricante [25] existem inúmeros exemplos de aplicações no ADSP-2181 que foram um grande auxílio na implementação dos filtros.

```
{--- define sampling rate in kHz: --------------}
{0xc850 = 8 | 0xc851 = 5.5125  | 0xc852 = 16    }
{0xc853 = 11.025|0xc854=27.42857|0xc855 = 18.9  }
{0xc856 = 32 | 0xc857 = 22.05  | 0xc859 = 37.8  }
{0xc85b = 44.1 | 0xc85c = 48   | 0xc85d = 33.075}
{0xc85e = 9.6    | 0xc85f = 6.615               }
.const fs = 0xc850;
{-----------------------------------------------}
.include <C:\adi_dsp\macros\begin.dsp>;
{--- define constants, variables, and buffers --}
.const M = 9;
.var/dm/circ w[M+1];
.var/pm/circ h[M+1];

i2 = ^w; L2 = %w;
i4 = ^h; L4 = %h;

zero(i2, m2, L2);
.init h: <fir.hex>;
{--- start processing input samples ------------}

wait: idle; jump wait;
input_samples: ena sec_reg;

{--- read input samples from codec -------------}

ax1 = dm(rx_buf + 1);         {left input sample}
mx1 = dm(rx_buf + 2);        {right input sample}

{--- sample processing algorithm ---------------}

cfir(M, i4, m4, i2, m2, mx1);

{--- write output samples to codec -------------}

dm(tx_buf + 1) = ax1;        {left output sample}
dm(tx_buf + 2) = mr1;       {right output sample}

{--- return from interrupt ---------------------}
     rti;
.include <c:\adi_dsp\macros\end.dsp>;
```

Figura 73: Excerto de código do algoritmo do filtro FIR

Na Figura 73 é possível visualizar o excerto de código utilizado para implementar o filtro FIR no DSP. Como pode ser visualizado, a frequência de amostragem escolhida foi os 8 kHz. No código anterior, pode-se notar que o algoritmo chama alguns ficheiros externos, estes têm como função efetuar as inicializações do ADSP. São ainda utlizados macros como, por exemplo, a macro `cfir` que vai permitir calcular e implementar o filtro em questão. Estas macros são de enorme importância para o desenvolvimento do algoritmo para o filtro FIR bem como para o filtro IIR (ver Anexo D).

As macros utilizadas foram baseados no artigo "*ADSP-2181 Experiments*" do Sophocles J. Orfanidis [7]. Essas macros não passam de funções pré-desenvolvidas para implementar certas operações.

Além da macro de inicialização e da macro `cfir` é ainda utilizado um ficheiro externo denominado `FIR.hex`. Este ficheiro tem como responsabilidade importar os coeficientes calculados no MatLab.

```
2000
2000
1000
1000
0c00
0c00
0a00
0a00
08c0
08c0
```

Figura 74: Excerto de código dos coeficientes do filtro FIR.hex

Como é possível visualizar na Figura 74 os coeficientes não estão no mesmo formato do MatLab. Para efetuar a importação dos coeficientes para o DSP é utilizado a função `dec2hex` [7]. Esta função irá efetuar a conversão dos coeficientes do filtro FIR de decimal para hexadecimal no formato 1.15 (valores compreendidos entre -1.0 e 0.999969482121875).

Nos parágrafos anteriores foi apresentado o algoritmo desenvolvido para implementar um filtro FIR no DSP e a função externa responsável por armazenar os coeficientes calculados. Como passo seguinte foram efetuadas alterações no ficheiro "Autoexec.bat" do Windows para permitir a compilação do algoritmo anterior, tal como mostra a Figura 75.

```
SET ADI_DSP=C:\ADI_DSP
SET PATH=%PATH%;C:\ADI_DSP\21xx\BIN
```

Figura 75: Excerto de código do Autoexec.bat

Além de efetuar a alteração no "Autoexec.bat" procedeu-se à criação de um ficheiro *.bat* responsável pela compilação do algoritmo do filtro FIR, com os comandos ilustrados na Figura 76.

```
asm21 FIR.DSP -2181 -l
ld21 fir -a adsp2181 -e fir -x -g
```

Figura 76: Excerto de código do FIR.bat

4.2.2. ALGORITMO DO FILTRO IIR

O algoritmo do filtro IIR é muito idêntico ao algoritmo do filtro FIR, só que neste caso a macro utilizada para efetuar os cálculos necessários é a `ccan` [7].

```
{--- define sampling rate in kHz: --------------}
{0xc850 = 8 | 0xc851 = 5.5125  | 0xc852 = 16    }
{0xc853 = 11.025|0xc854=27.42857|0xc855 = 18.9  }
{0xc856 = 32 | 0xc857 = 22.05  | 0xc859 = 37.8  }
{0xc85b = 44.1 | 0xc85c = 48   | 0xc85d = 33.075}
{0xc85e = 9.6    | 0xc85f = 6.615               }
.const fs = 0xc850;
{-----------------------------------------------}

.include <C:\ADI_DSP\macros\begin.dsp>;

{--- define constants, variables, and buffers --}

.const M = 4;
.var/dm/circ  w[M+1];
.var/pm/circ  a[M+1], b[M+1] ;

.init  a: <A.hex>;     {denominator coefficients}
.init  b: <B.hex>;       {numerator coefficients}

.const ea = 0;
.const eb = 0;

i2 = ^w;  L2 = %w;
i7 = ^a;  L7 =2*(M+1
zero(i2, m2, L2);

{--- start processing input samples ------------}

wait: idle; jump wait;
input_samples: ena sec_reg;

{--- read input samples from codec -------------}

ax1 = dm(rx_buf + 1);         {left input sample}
```

```
mx1 = dm(rx_buf + 2);        {right input sample}

{--- sample processing algorithm -- process right
input only ------------}

ccan(M, i7, m7, i2, m2, ea, eb, ax1);

{--- write output samples to codec ------------}

dm(tx_buf + 1) = mx1;        {left output sample}
dm(tx_buf + 2) = sr1;       {right output sample}

{--- return from interrupt --------------------}

rti;

.include <c:\adi_dsp\macros\end.dsp>;
```

Figura 77: Excerto de código do algoritmo do Filtro IIR

Como é possível visualizar no excerto de código da Figura 77 o funcionamento do algoritmo do filtro IIR é muito idêntico ao filtro FIR. As únicas diferenças são que neste caso vão existir dois ficheiros externos para efetuar a leitura dos coeficientes do numerador e do denominador, a macro utilizada para efetuar os cálculos necessários é a `ccan` e ainda a inclusão de duas constantes (`ea` e `eb`).

As constantes `ea` e `eb` são utilizadas quando os valores dos coeficientes ultrapassem o limite do definido pelo formato 1.15. De seguida vai ser apresentado um exemplo da utilização destes parâmetros. Considere a função de transferência de um filtro IIR da Figura 78.

```
Gz1 =

  1 + 0.06686 z^-1 - 1.329 z^-2 - 0.0616 z^-3 + 0.3757 z^-4 + 0.007344 z^-5
  ---------------------------------------------------------------------------
  1 - 0.9331 z^-1 - 0.8957 z^-2 + 0.8007 z^-3 + 0.1144 z^-4 - 0.08461 z^-5
```

Figura 78: Exemplo da função de transferência para um filtro IIR

Como é possível observar no numerador existem coeficientes superiores a 1 e para efetuar a conversão para o formato 1.15 é necessário implementar a sua normalização. Para tal, todos os coeficientes são divididos por 4 (X=4) e na constante `eb` é posto com o valor 2 (`eb`=2). Isto ocorre porque quando são efetuadas normalizações tanto no denominador como no numerador a fase do sinal de saída não será afetada mas somente a amplitude vai sofrer alteração. Utilizando essas duas constantes é possível normalizar os coeficientes

para o formato 1.15 sem haver alteração na amplitude do sinal de saída. Os valores possíveis para efetuar a normalização são aqueles que cumprem a condição que $X = 2^{eb}$(ea=eb).

Os coeficientes calculados no MatLab normalizados são convertidos através do método anterior, utilizando a função dec2hex.

Em comparação com o filtro FIR, após a implementação do algoritmo e da importação dos coeficientes convertidos e normalizados é necessário efetuar a compilação do sistema. Para tal, foi criado outro ficheiro *.bat* que permite a compilação do algoritmo do filtro IIR, utilizando os comandos da Figura 79.

```
asm21 IIR.DSP -2181 -l
ld21 iir -a adsp2181 -e iir -x -g
```

Figura 79: Excerto de código do IIR.bat

4.2.3. ALGORITMO DA ESTRUTURA DE UM FILTRO IIR

Nesta secção são apresentados os algoritmos desenvolvidos para a implementação do filtro IIR para uma estrutura em cascata e paralelo.

4.2.3.1. ESTRUTURA CASCATA

A implementação do algoritmo de um filtro IIR com a topologia em cascata consistiu numa fase muito complexa apesar da sua aparente simplicidade.

```
{--- define sampling rate in kHz: --------------}
{0xc850 = 8 | 0xc851 = 5.5125  | 0xc852 = 16    }
{0xc853 = 11.025|0xc854=27.42857|0xc855 = 18.9  }
{0xc856 = 32 | 0xc857 = 22.05  | 0xc859 = 37.8  }
{0xc85b = 44.1 | 0xc85c = 48   | 0xc85d = 33.075}
{0xc85e = 9.6    | 0xc85f = 6.615              }
.const fs = 0xc850;
{-----------------------------------------------}
.include <C:\ADI_DSP\macros\begin.dsp>;
{--- define constants, variables, and buffers --}

.const M = 2;
.var/dm/circ  q[M+1];
.var/dm/circ  w[M+1];
.var/pm/circ  a[M+1], b[M+1] ;
.var/pm/circ  e[M+1], f[M+1] ;

.init  a: <A0.hex>;
.init  b: <B0.hex>;
.init  e: <A1.hex>;
.init  f: <B1.hex>;
```

```
.const ea = 1;
.const eb = 1;
.const ac = 0;
.const bc = 1;
.const b0 = 0;

i2 = ^w;  L2 = %w;
i7 = ^a;  L7 =2*(M+1);
i3 = ^q;  L3 =%q;
i5 = ^e;  L5 =2*(M+1);

zero(i3, m3, L3);
zero(i2, m2, L2);

{--- start processing input samples ------------}

wait: idle; jump wait;

input_samples: ena sec_reg;

{--- read input samples from codec -------------}

ax1 = dm(rx_buf + 1);         {left input sample}
mx1 = dm(rx_buf + 2);        {right input sample}
{--- sample processing algorithm ---------------}
ccan(M, i7, m7, i2, m2, ea, eb, ax1);
sr=ashift sr1 by b0 (hi);
ccan(M, i5, m5, i3, m3, ac, bc, sr1);

{--- write output samples to codec -------------}

dm(tx_buf + 1) = mx1;        {left output sample}
dm(tx_buf + 2) = sr1;       {right output sample}

{--- return from interrupt ---------------------}

rti;
.include <c:\adi_dsp\macros\end.dsp>;
```

Figura 80: Excerto de código da estrutura cascata para um filtro IIR

Na Figura 80 é possível visualizar o excerto de código desenvolvido para uma estrutura em cascata para um filtro IIR. De salientar, que este código é muito idêntico ao algoritmo do filtro IIR.

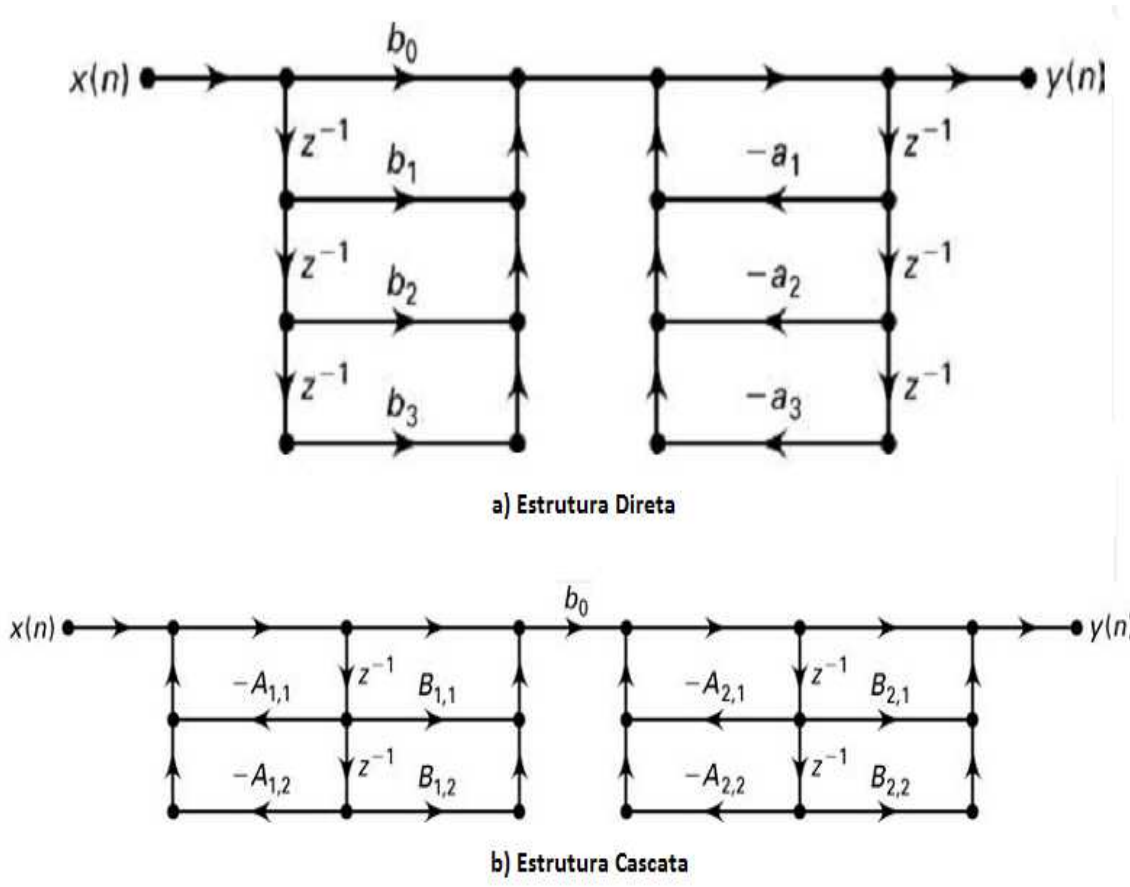

Figura 81: a) Estrutura direta de um filtro IIR (4,4), b) Estrutura cascata de um filtro IIR (2,2) por (2,2)

Na Figura 81 é possível visualizar uma estrutura de um filtro IIR fraccionário (OrdN,OrdD)=(4,4) e o equivalente para uma estrutura em cascata. Como foi referido anteriormente, o código desenvolvido para converter uma estrutura direta para uma estrutura cascata converte sempre para blocos (OrdN,OrdD)=(2,2). O algoritmo apresentado na Figura 80, foi desenvolvido tomando de base a Figura 81, convertendo uma estrutura direta (4,4) para uma estrutura em cascata constituída por dois blocos (2,2).

De igual modo ao algoritmo de um filtro IIR e FIR, após o seu desenvolvimento para a estrutura e o cálculo dos coeficientes é necessário compilar o mesmo. Para tal foi criado um ficheiro *.bat* com esse objetivo (Figura 82).

```
Asm21 IIR_Cascata.DSP -2181 –l
ld21 IIR_Cascata –a adsp2181 –e IIR_Cascata –x –g
```

Figura 82: Excerto de código do IIR_Cascata.bat

4.2.3.2. ESTRUTURA PARALELO

Na Figura 83 é possível visualizar o algoritmo de um filtro IIR com a topologia em paralelo.

```
 {--- define sampling rate in kHz: --------------
}
{0xc850 = 8 | 0xc851 = 5.5125  | 0xc852 = 16    }
{0xc853 = 11.025|0xc854=27.42857|0xc855 = 18.9  }
{0xc856 = 32 | 0xc857 = 22.05  | 0xc859 = 37.8  }
{0xc85b = 44.1 | 0xc85c = 48   | 0xc85d = 33.075}
{0xc85e = 9.6    | 0xc85f = 6.615               }
.const fs = 0xc850;
{-----------------------------------------------}

.include <C:\ADI_DSP\macros\begin.dsp>;

{--- define constants, variables, and buffers --}

.const M = 2;
.var/dm/circ  q[M+1];
.var/dm/circ  w[M+1];
.var/pm/circ  a[M+1], b[M+1] ;
.var/pm/circ  e[M+1], f[M+1] ;

.init  a: <A0.hex>;
.init  b: <B0.hex>;
.init  e: <A1.hex>;
.init  f: <B1.hex>;

.const ea = 0;
.const eb = 3;
.const ac = 2;
.const bc = 0;
.const b0 = 2;


i2 = ^w;  L2 = %w;
i7 = ^a;  L7 =2*(M+1);
i3 = ^q;  L3 =%q;
i5 = ^e;  L5 =2*(M+1);

zero(i3, m3, L3);
zero(i2, m2, L2);

{--- start processing input samples ------------}

wait: idle; jump wait;
input_samples: ena sec_reg;

{--- read input samples from codec -------------}

ax1 = dm(rx_buf + 1);
mx1 = dm(rx_buf + 2);
```

```
{--- sample processing algorithm ---------------}

ccan(M, i7, m7, i2, m2, ea, eb, ax1);
ax0=sr1;
ccan(M, i5, m5, i3, m3, ac, bc, ax1);
ay0=sr1;
ar=ax0+ay0;
sr1=ax1;
sr=ashift sr1 by b0 (hi);
ay1=sr1;
ar=ar+ay1;

{--- write output samples to codec -------------}

dm(tx_buf + 1) = mx1;
dm(tx_buf + 2) = ar;

{--- return from interrupt ---------------------}

   rti;

.include <c:\adi_dsp\macros\end.dsp>;
```

Figura 83: Excerto de código da estrutura paralelo para um filtro IIR

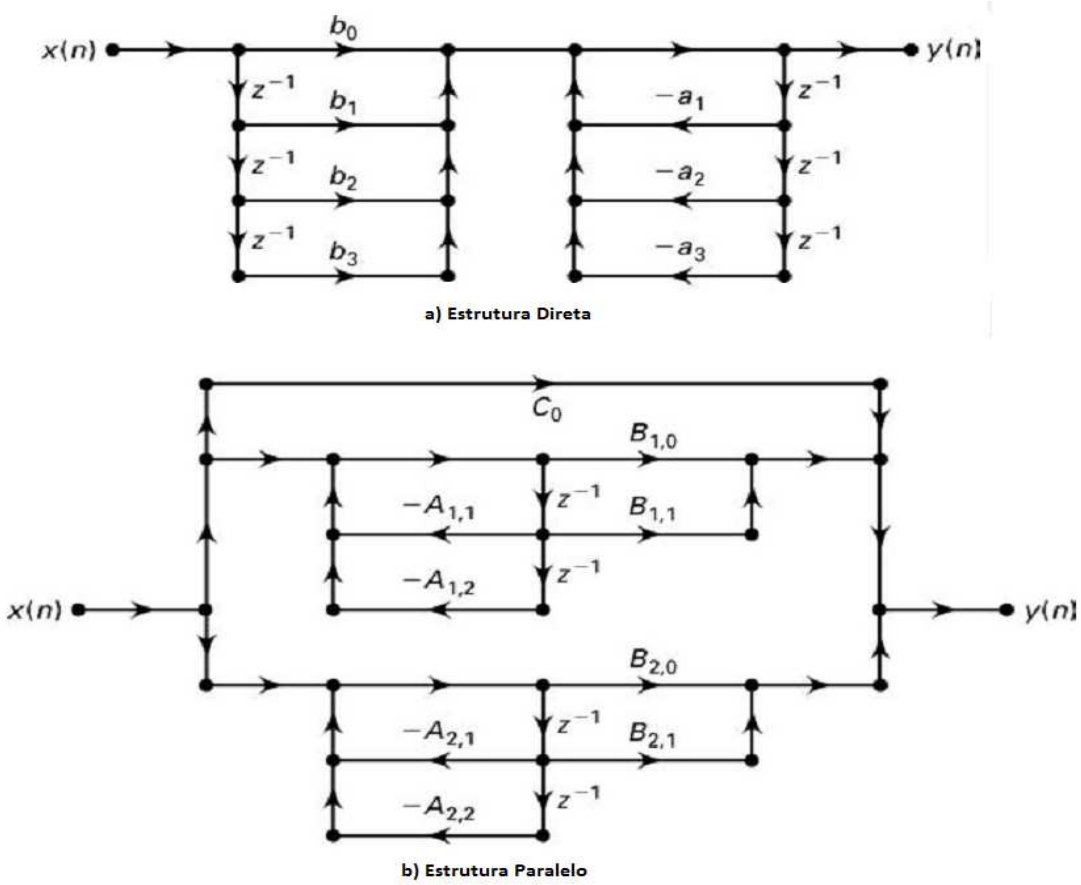

Figura 84: a) Estrutura direta de um filtro IIR (4,4), b) Estrutura paralelo de um filtro IIR (2,2) por (2,2)

Na Figura 84 é possível visualizar uma estrutura direta de um filtro IIR (4,4) e respetiva conversão para um filtro IIR com uma estrutura paralelo (2,2) por (2,2). Em comum com a estrutura em cascata, o algoritmo apresentado na Figura 83 foi desenvolvido com base nesta conversão. Tal como nos algoritmos anteriores foi criado um ficheiro *.bat* responsável pela compilação do algoritmo (Figura 85).

```
asm21 IIR_Paralelo.dsp -2181 -l
ld21 IIR_Paralelo  -a adsp2181 -e IIR_Paralelo -x -g
```

Figura 85: Excerto de código do IIR_Paralelo.bat

4.2.4. *LOADING* DO ALGORITMO

Após a compilação dos algoritmos desenvolvidos para o ADSP-2181 é necessário enviar o ficheiro executado para o DSP. Para tal, como já foi referido anteriormente, o ADSP-2181 tem incluído uma porta RS-232, onde através do *software Ez-Kite Lite Monitor* é possível enviar o executável. O *download* do executável é efetuado clicando na opção *Loading* e posteriormente na opção *Download user program and go* escolhendo qual o ficheiro a enviar (ver Figura 86). De salientar que sempre que seja necessário enviar um programa para a DSP é necessário efetuar o *reset* (clicando no botão *reset* da ADSP-2181).

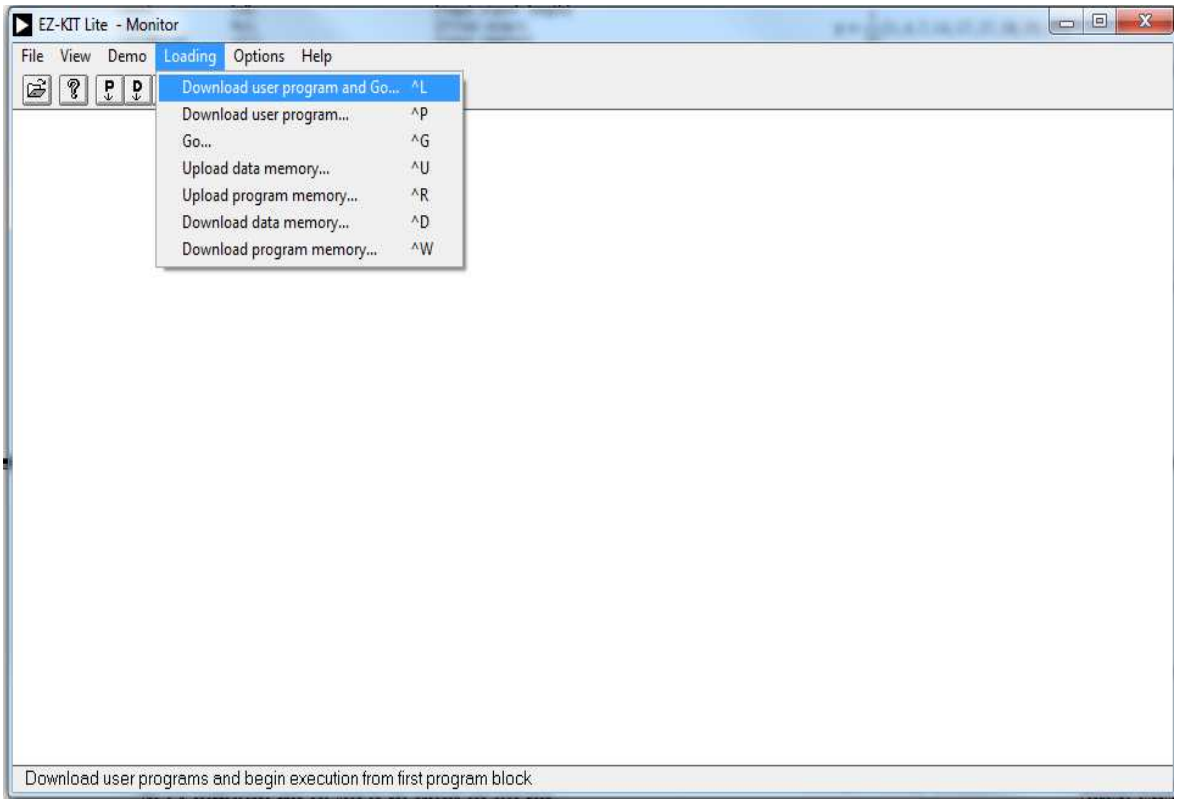

Figura 86: Software Ez-Kite Lite Monitor

4.3. EXEMPLOS DE FILTROS

Nesta secção vão ser apresentados alguns exemplos implementados na ADSP-2181, tendo em consideração que o único sinal que se pode comparar os resultados simulados em Matlab com os resultados obtidos no ADSP é uma onda sinusoidal, para qual, a fase é determinada corretamente utilizando os diagramas de Bode.

4.3.1. EXEMPLOS DO FILTRO FIR

Inicialmente é apresentado um exemplo de um filtro FIR para os casos diferenciador e integrador para uma onda sinusoidal de entrada.

A Figura 87 ilustra o diagrama de Bode para o filtro FIR integrador com α=-0.5, T=1/8000=0.000125 s , N=100 e utilizando a aproximação de Euler. Tal como ilustrado na figura, para a frequência de 280 Hz o valor da fase é de -45.5º. Na Figura 88 é mostrado o resultado da implementação no DSP onde se constata que o valor da fase é de aproximadamente -45º, como seria de esperar (ver Figura 87).

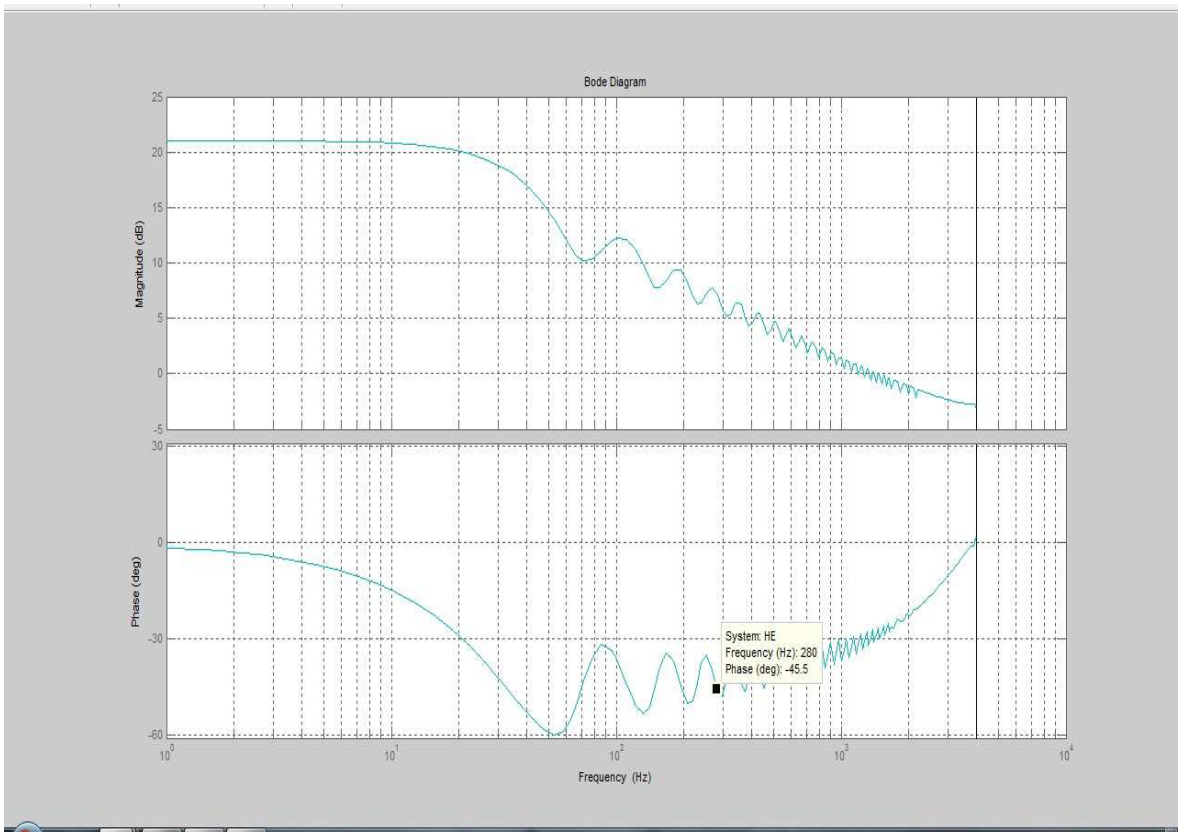

Figura 87: Diagramas de Bode de um filtro FIR com α=-0.5, T=0.000125 s, N=100, aproximação de Euler (Integrador)

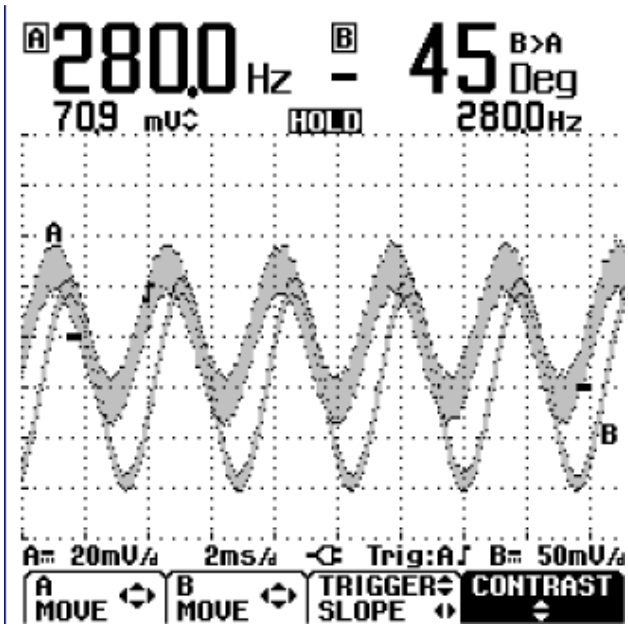

Figura 88: Resultado experimental para um filtro FIR com α=-0.5, T=0.000125 s, N=100 e aproximação de Euler (Integrador)

As Figuras 89 e 90 ilustram a situação anterior para um filtro FIR diferenciador com α=0.5 e os mesmos parâmetros do integrador. Através da observação da Figura 89 é possível verificar que para o valor da frequência de 18.1 Hz a fase assume o valor de 45º.

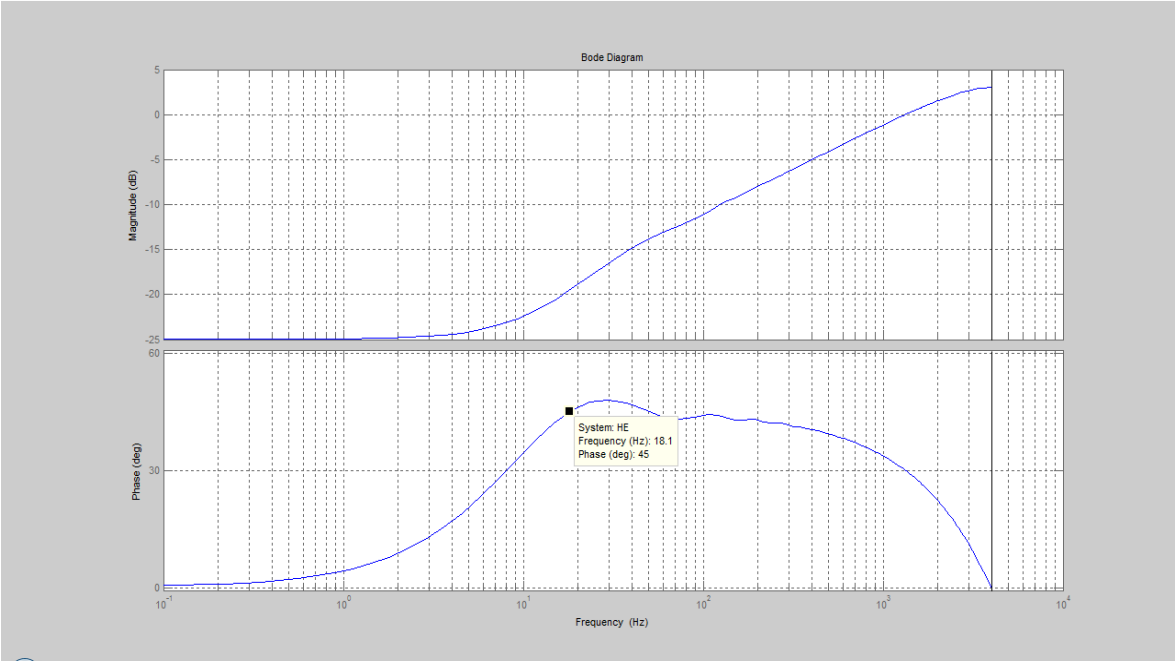

Figura 89: Diagramas de Bode de um filtro FIR com α=0.5, T=0.000125 s, N=100, aproximação de Euler (Diferenciador)

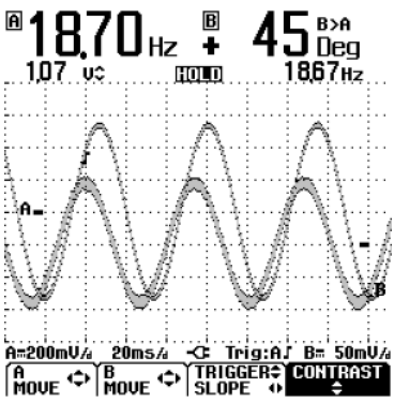

Figura 90: Resultado experimental para um filtro FIR com α=0.5, T=0.000125 s, N=100 e aproximação de Euler (Diferenciador)

Através da análise das Figura 87, 88, 89 e 90 é possível verificar que os valores obtidos experimentalmente coincidem com os valores obtidos por simulação no MatLab.

Como segunda experiência foi escolhido um valor para a frequência de *f*=35 Hz e variou-se o valor da ordem do filtro de N={10, 20, 50, 100, 200, 500, 1000}, comparando o valor do desfasamento obtido com o do MatLab. Para tal, considerou-se o valor de T=0.000125 s, α=-0.5 (integrador) e a aproximação de Tustin.

Na Figura 91 é possível visualizar que para os diferentes valores da ordem do filtro a fase aproxima-se do valor esperado de -45º. Para N=1000 e *f*=35 Hz o valor da fase é de -45.1º.

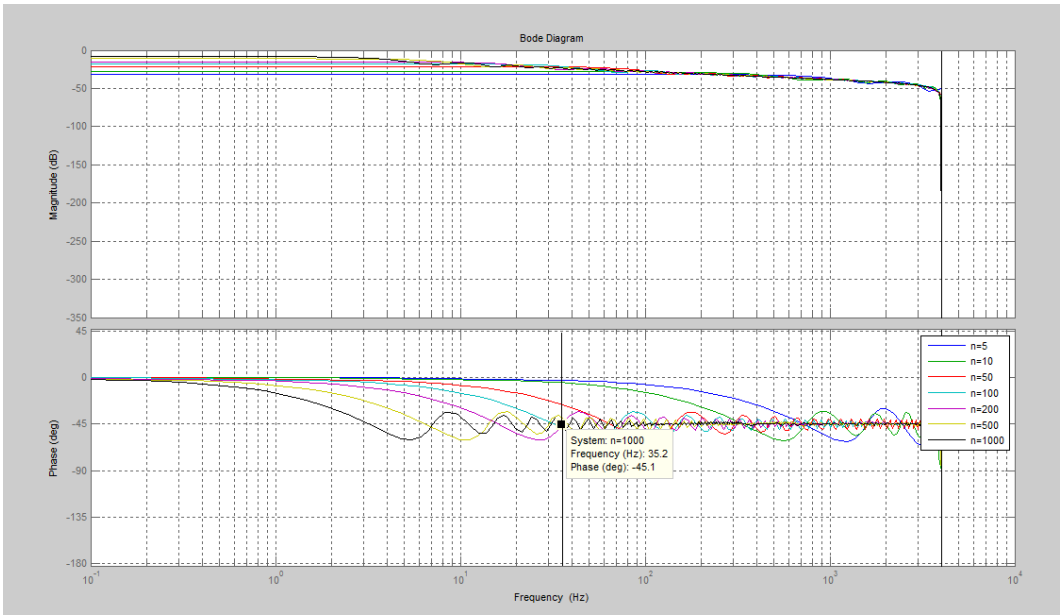

Figura 91: Diagrama de Bode de um Filtro FIR com α=-0.5, T=0.000125 s, *f*≅ 35 Hz, aproximação de Tustin (Integrador)

A Figura 92 mostra os resultados obtidos no DSP, em que se confirma que estes se encontram de acordo com os obtidos no MatLab (Figura 91).

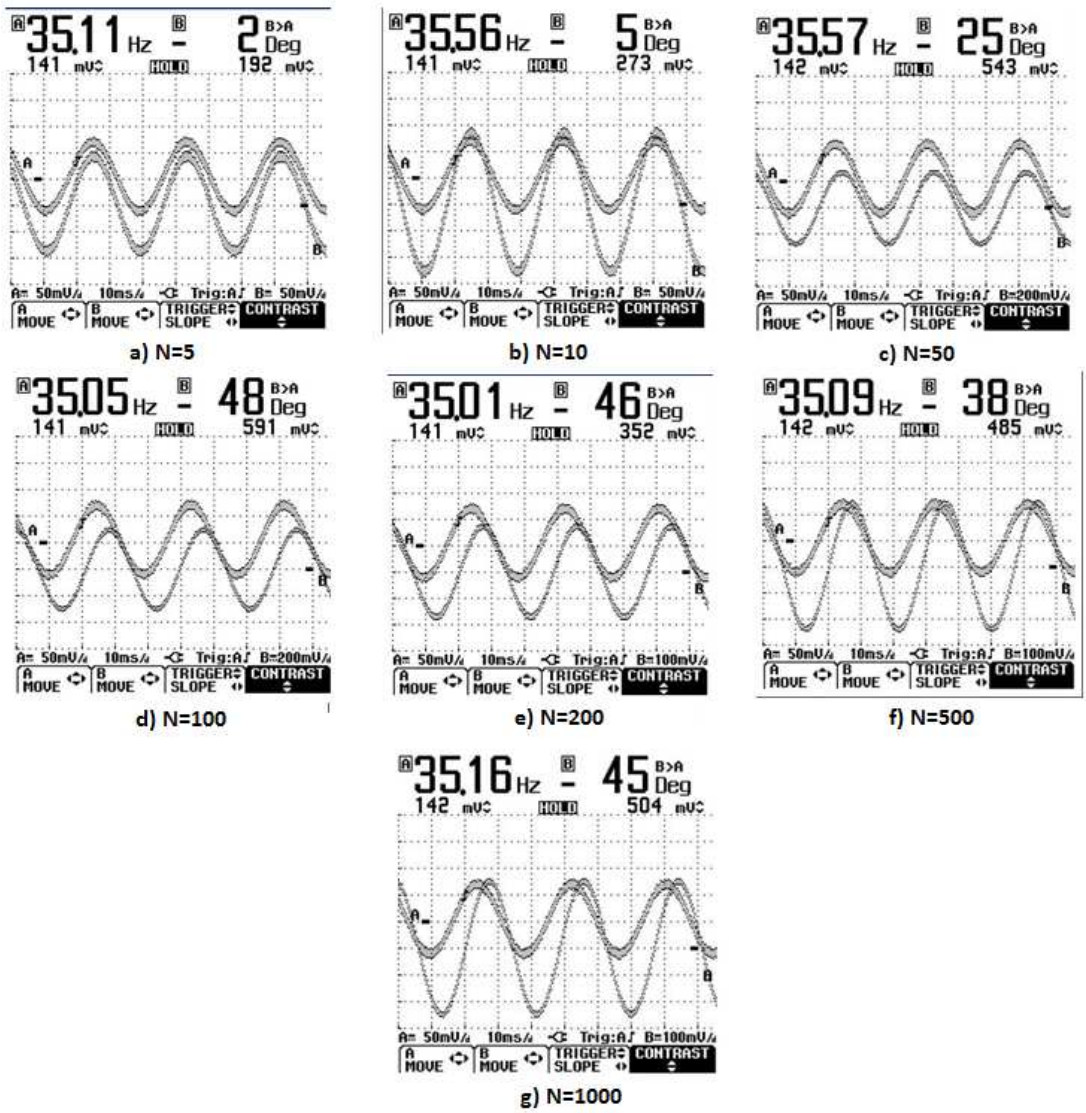

Figura 92: Resultado experimental para um filtro FIR com α=-0.5, T=0.000125 s, *f*≅ 35 Hz, aproximação de Tustin (Integrador) para N={5,10,50,100,200,500,1000}

Como segunda parte desta experiência foi escolhido um valor para a frequência de *f*=15 Hz e variou-se o valor da ordem do filtro de N={10, 20, 50, 100, 200, 500, 1000}, comparando o valor do desfasamento obtido com o do MatLab. Para tal, considerou-se o valor de T=0.000125 s, α=0.5 (diferenciador) e a aproximação de Tustin.

Na Figura 93 é possível visualizar que para os diferentes valores da ordem do filtro a fase aproxima-se do valor esperado de +45º. Para N=1000 e *f*=15.1 Hz o valor da fase é de 45.2º.

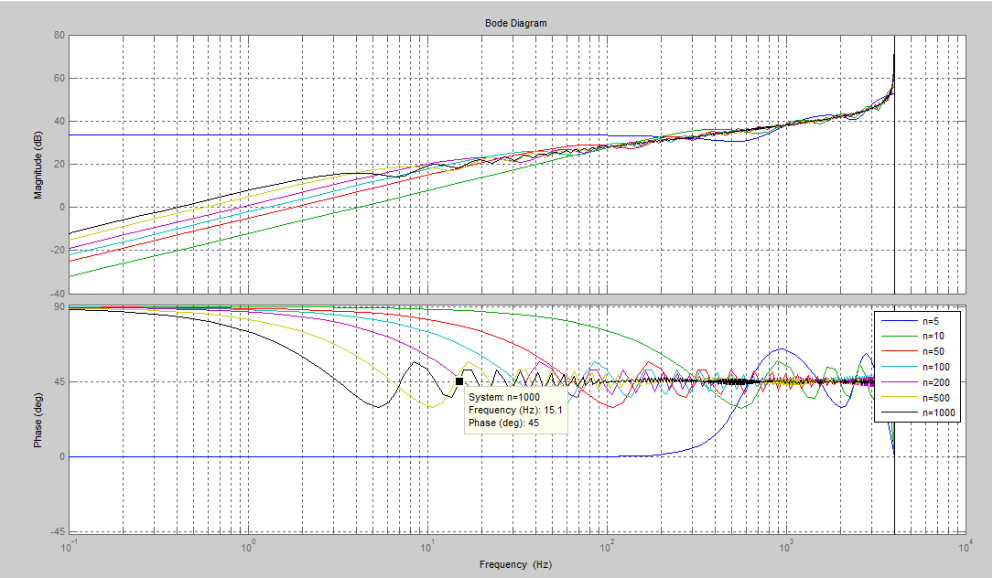

Figura 93: Diagramas de Bode de um filtro FIR com α=0.5, T=0.000125 s, $f \cong 15$ Hz, aproximação de Tustin (Diferenciador)

A Figura 94 ilustra os resultados obtidos no DSP em que mais uma vez se verifica a concordância com os simulados (Figura 93). Aqui observa-se que à medida que se aumenta a ordem do filtro mais próximo se consegue chegar ao valor de fase pretendido.

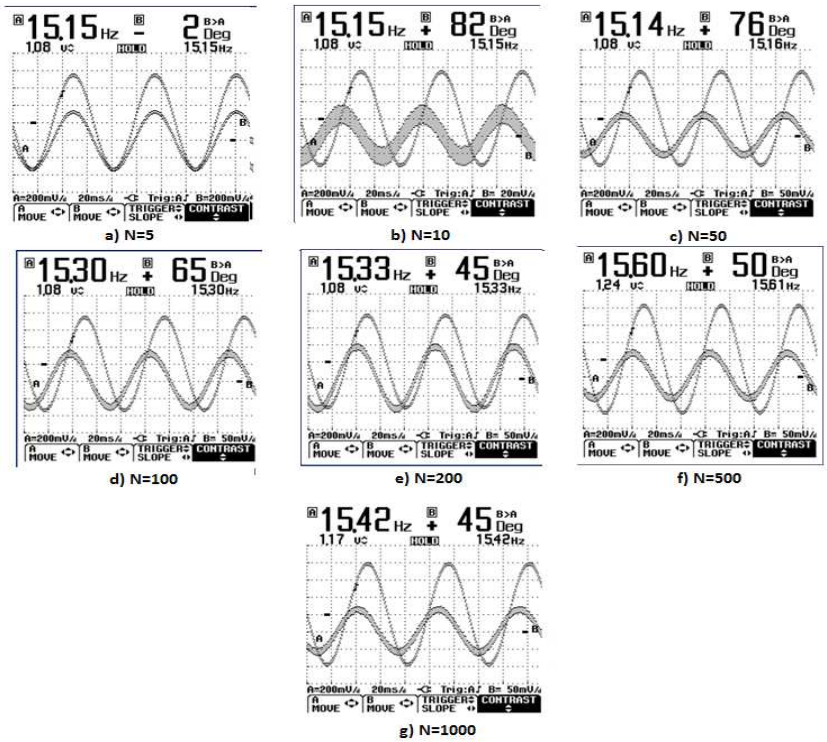

Figura 94: Resultado experimental para um filtro FIR com α=0.5, T=0.000125 s, $f \cong 15\ Hz$, aproximação de Tustin (Diferenciador) para N={5,10,50,100,200,500,1000}

Como terceira experiência foi escolhida a aproximação de Alaoui e manteve-se fixa o valor da frequência (17 Hz para o diferenciador e 27 Hz para o integrador). Como já foi referido anteriormente, só é possível comparar os valores de fase experimentais com os valores obtidos pelo diagrama de Bode no MatLab quando o sinal de entrada é uma sinusoide. Nesta experiência, pretende-se mostrar os resultados dos filtros (FIR e IIR) para um sinal triangular e para uma onda quadrada. Os parâmetros de entrada escolhidos foram os seguintes:

- T=0.000125 s;
- N=1000;
- α=±0.5 (Diferenciador e Integrador);
- Aproximação de Al-Alaoui.

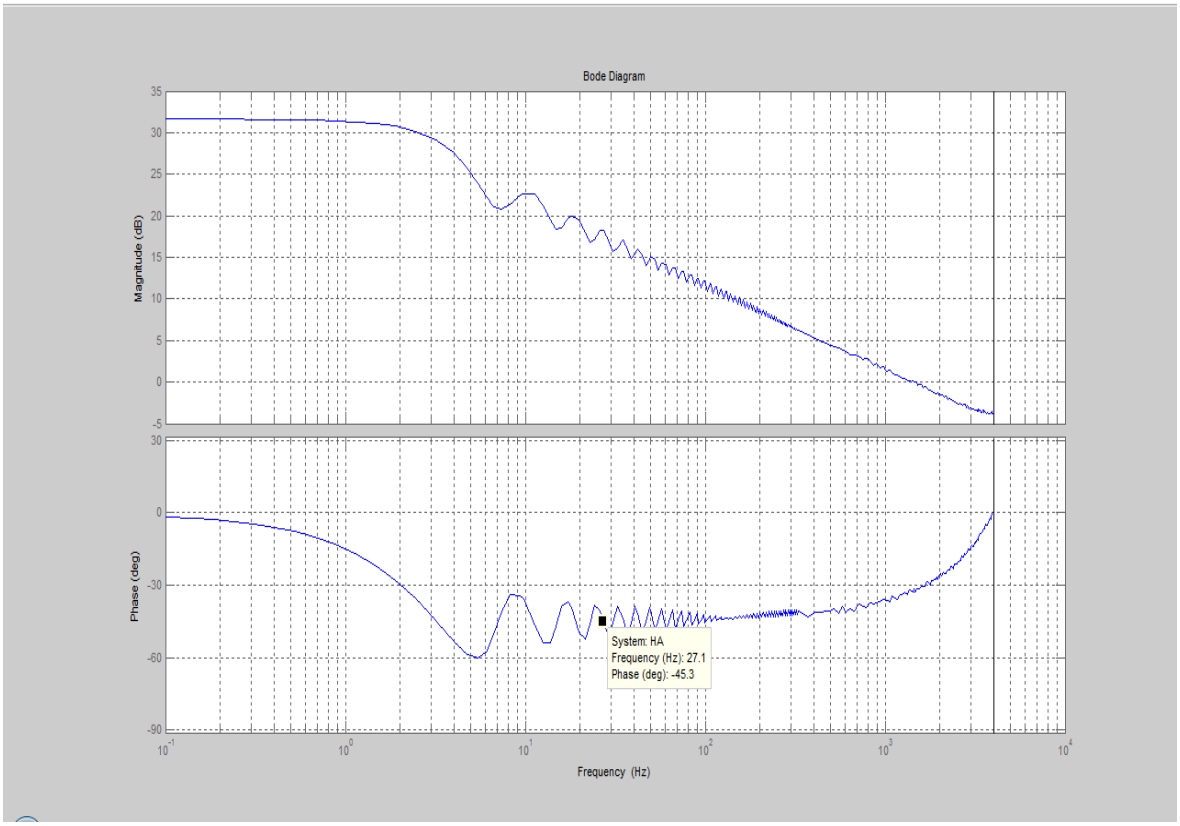

Figura 95: Diagrama de Bode de um filtro FIR com α=-0.5, T=0.000125 s , *f*≅ 27 Hz, aproximação de Al-Alaoui (Integrador)

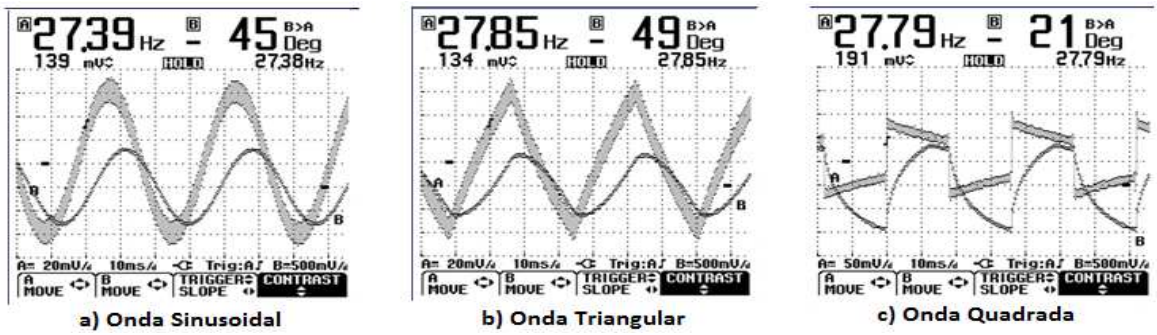

Figura 96: Resultado experimental para um filtro FIR com α=-0.5, T=0.000125 s, N=1000 $f \cong$ 27 Hz, aproximação de Al-Alaoui (Integrador) de uma onda sinusoidal, triangular e quadrada

Na Figura 95 é possível observar o diagrama de Bode para uma onda sinusoidal de um filtro FIR com α=-0.5 (Integrador), T=0.000125 s, $f \cong$ 27 Hz, e aproximação de Al-Alaoui (Integrador) onde o valor da fase para essa frequência é de -45º. Na Figura 96 é possível observar o resultado experimental anterior mas considerando a onda sinusoidal, triangular e quadrada. Verifica-se que para a onda sinusoidal e onda triangular a fase aproxima-se do valor esperado (Figura 95) enquanto para a onda quadrada o valor da fase é muito diferente, como seria de esperar.

Na Figura 97 é possível visualizar o diagrama de Bode para o caso anterior mas agora considerando a frequência f=17 Hz e o valor de α=+0.5 (Diferenciador). Para essa frequência o valor da fase assume +45º. A Figura 98 mostra os resultados obtidos para a aplicação de uma onda sinusoidal, triangular e quadrada.

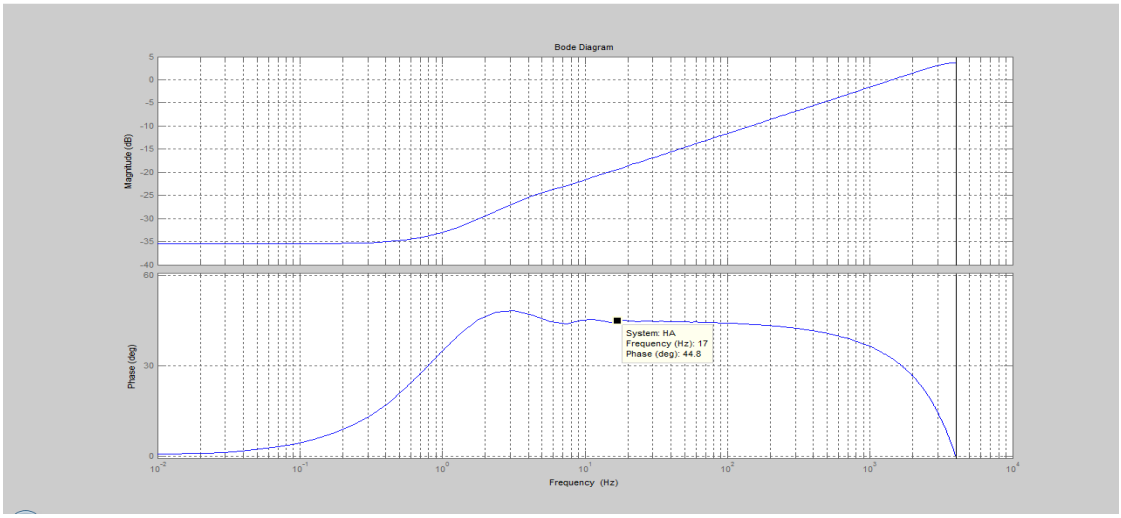

Figura 97: Diagrama de Bode de um Filtro FIR com α=+0.5, T=0.000125 s, $f \cong$ 17 Hz, aproximação de Al-Alaoui (Diferenciador)

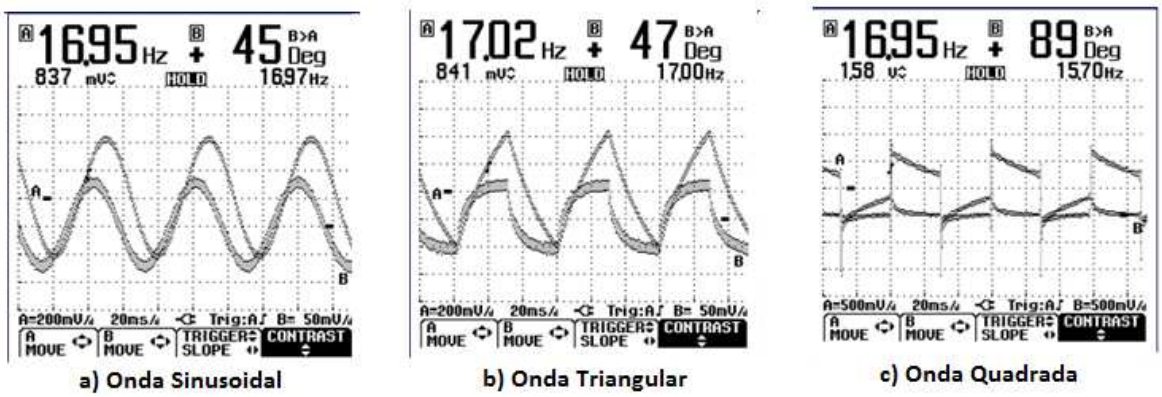

Figura 98: Resultado experimental para um filtro FIR com α=0.5, T=0.000125 s, N=1000 $f\cong$ 17 Hz, aproximação de Al-Alaoui (Diferenciador) de uma onda sinusoidal, triangular e quadrada

Nesta experiência é possível analisar que tanto para o caso de um filtro diferenciador ou integrador quando o sinal de entrada é alterado de uma sinusoide para uma triangular o valor de desfasamento é alterado mas ainda próximo do valor pretendido. No entanto, para uma onda quadrada é possível notar que o valor do desfasamento atinge valores muito diferentes, como esperado. No entanto verifica-se que em todas as situações o filtro está a diferenciar/integrar como pretendido.

4.3.2. EXEMPLOS DO FILTRO IIR

De seguida e em comum com o filtro FIR são apresentados dois exemplos da implementação de um Filtro IIR na ADSP-2181. Neste caso a experiência selecionada é idêntica à segunda experiência do Filtro FIR, que consiste em manter o valor fixo para a frequência, no caso do integrador de 330 Hz e do diferenciador de 77 Hz aproximadamente e alterar o valor da ordem do denominador e numerador para N=OrdN=OrdD=1,2..,5 e analisar qual o valor da fase obtido. Os restantes parâmetros de entrada foram, T=0.000125 s, α=±45 e utilizando a aproximação de Tustin. Nas experiências seguintes foi utilizado o método de Prony para 1000 amostras.

Na Figura 99 é possível visualizar que para os diferentes valores da ordem do filtro com α=-0.5 (Integrador), T=0.000125 s, $f\cong$ 330 Hz e aproximação de Tustin a fase aproxima-se do valor esperado de -45º. Para N=5 e f=330 Hz o valor da fase é de -45º.

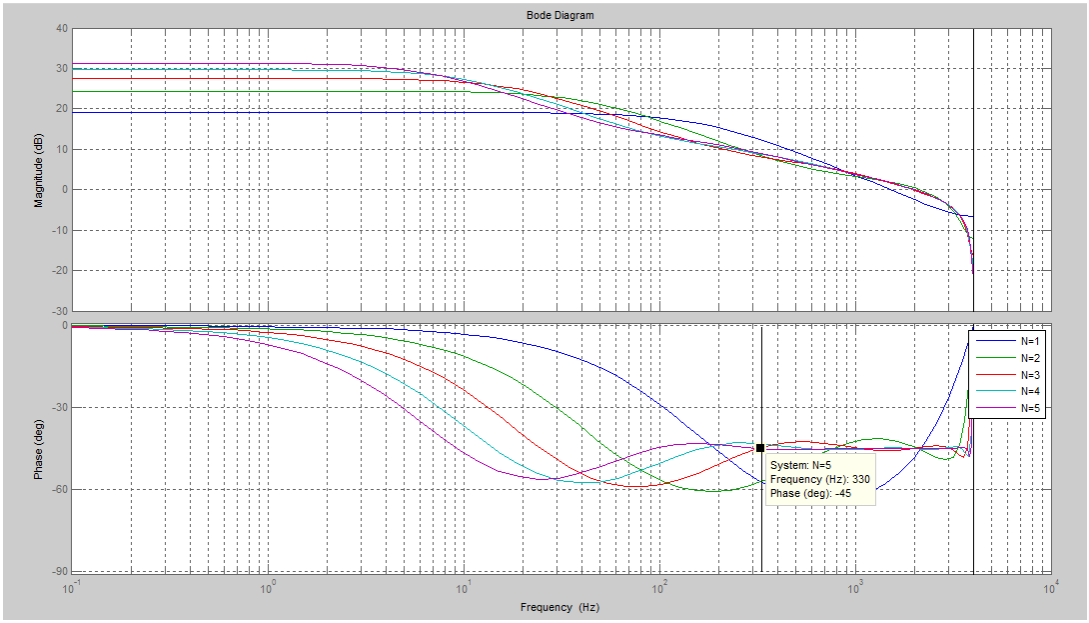

Figura 99: Diagramas de Bode de um Filtro IIR com α=-0.5, T=0.000125 s, *f*≅ 330 Hz, aproximação de Tustin (Integrador)

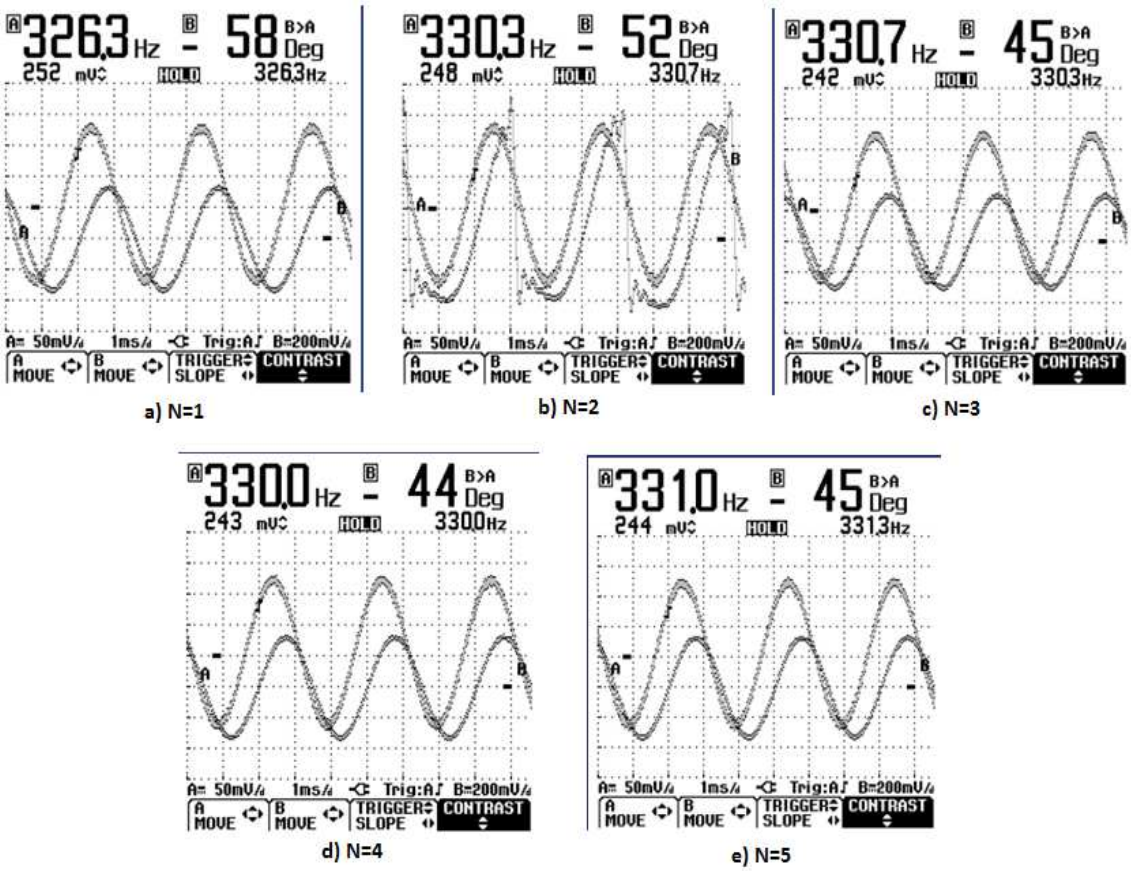

Figura 100: Resultado experimental para um filtro IIR com α=-0.5, T=0.000125 s, *f*≅ 330 Hz, aproximação de Tustin (Integrador) para N={1,2,3,4,5}

Pela observação da Figura 100 é possível concluir que à medida que se aumenta a ordem do filtro o valor da fase aproxima-se do valor esperado (Figura 99).

No caso de um filtro IIR diferenciador a experiência realizada utiliza os parâmetros de entrada do filtro IIR integrador. Na Figura 101 é possível visualizar que para os diferentes valores da ordem do filtro com α=0.5 (Diferenciador), T=0.000125 s, $f \cong 77$ Hz e aproximação de Tustin a fase aproxima-se do valor esperado de +45º. Para N=5 e f=77 Hz o valor da fase é de +45º.

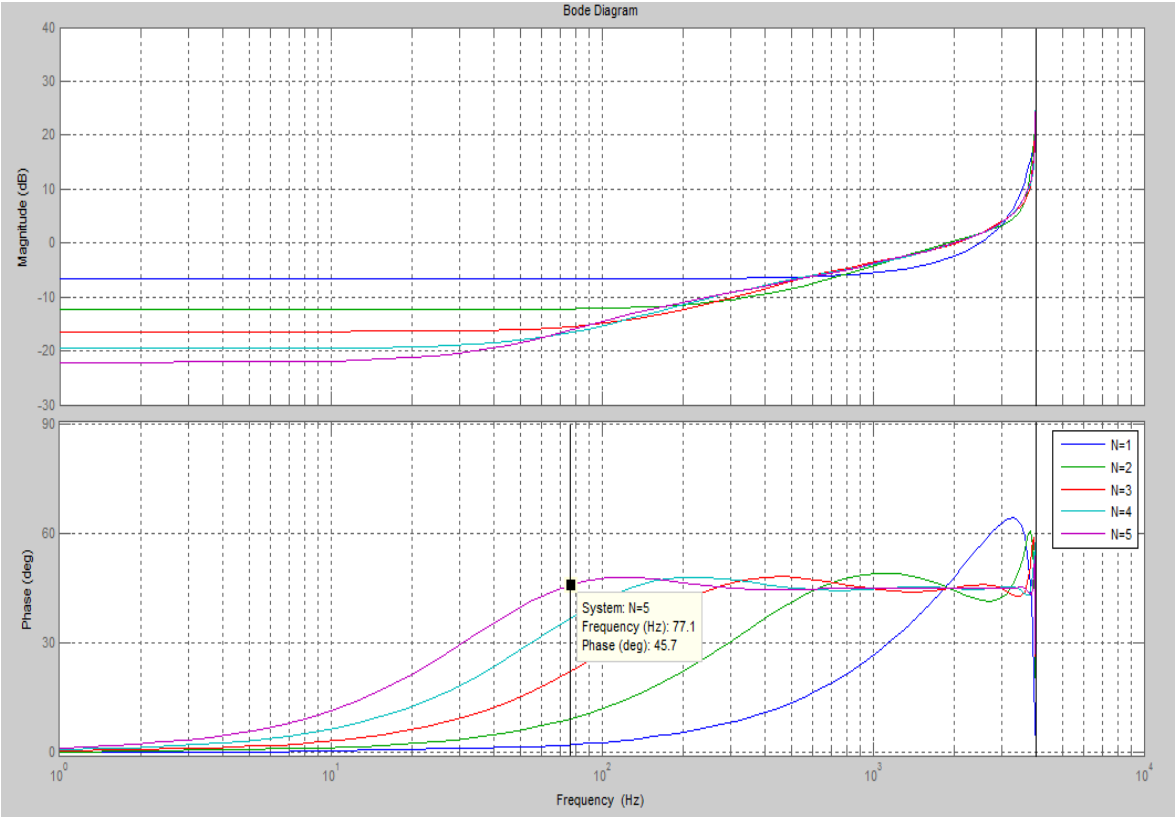

Figura 101: Diagramas de Bode de um Filtro IIR com α=+0.5, T=0.000125 s, $f \cong 77$ Hz, aproximação de Tustin (Diferenciador)

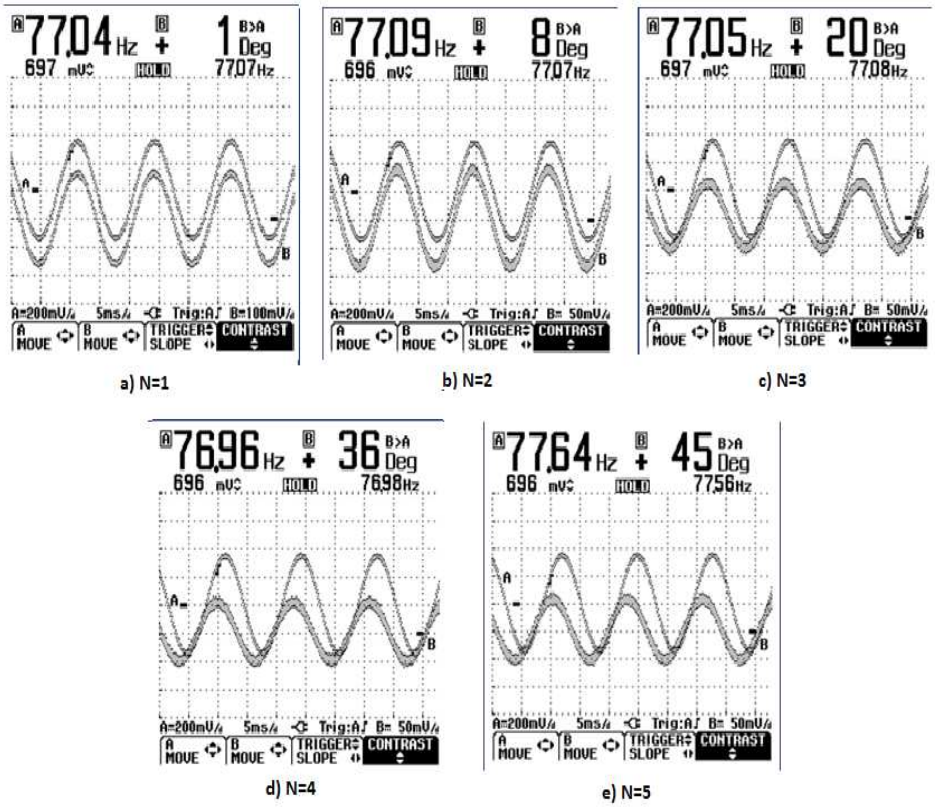

Figura 102: Resultado experimental para um filtro IIR com α=0.5, T=0.000125 s, $f \cong$ 77 Hz, aproximação de Tustin (Diferenciador) para N={1,2,3,4,5}

Como é possível analisar nas Figuras 101 e 102 anteriores sempre que se aumenta o valor da ordem dos coeficientes do numerador e denominador é possível ver que, no caso do diferenciador, a fase aproxima-se cada vez mais do valor pretendido, como seria de esperar.

De notar, que é preciso escolher uma amplitude de entrada certa de modo que a amplitude de saída não seja muito grande nem muito pequena. Também é necessário escolher um valor de frequência propício para que a amplitude de saída não tenha um ganho muito grande nem muito pequeno. Estas limitações advêm do DSP, para o sinal de saída não saturar ou apresentar ruído.

Os valores escolhidos para a ordem do numerador e do denominador no caso do filtro IIR não pode ser muito alta senão os coeficientes vão assumir valores muito pequenos e quando efetuada a conversão dos coeficientes para o formato 1.15 poderá ocorrer erros significativos de truncatura.

Como segunda experiência foi escolhida a aproximação de Tustin e manteve-se fixa o valor da frequência (1040 Hz para o diferenciador e 266 Hz para o integrador). Nesta

experiência, pretende-se mostrar os resultados dos filtros IIR para um sinal triangular e para uma onda quadrada. Os parâmetros de entrada escolhidos foram os seguintes:

- T=0.000125 s;
- N=1000;
- α=±0.5 (Diferenciador e Integrador);
- Aproximação de Tustin;
- Ordem do numerador (OrdN)= ordem do denominador (OrdD)=3.

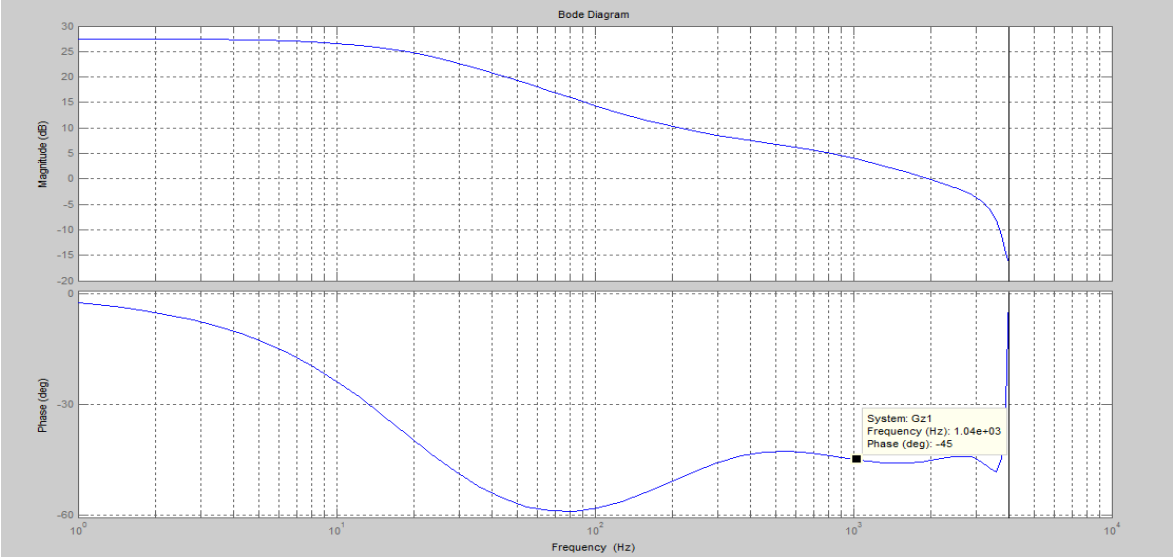

Figura 103: Diagramas de Bode de um Filtro IIR com α=-0.5, T=0.000125 s, OrdN=OrdD=3, *f*≅ 1040 Hz, aproximação de Tustin (Integrador)

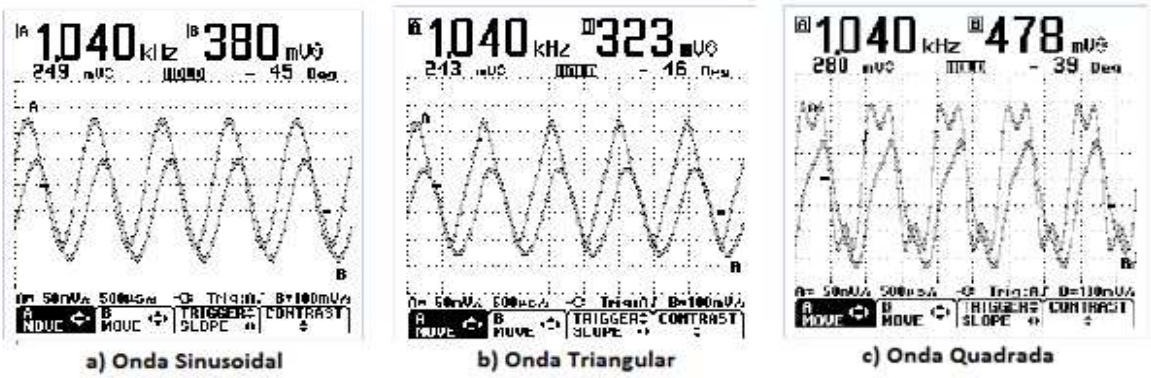

Figura 104: Resultado experimental para um filtro IIR com α=-0.5, T=0.000125 s, OrdN=OrdD=3, *f*≅ 1040 Hz, aproximação de Tustin (Integrador) de uma onda sinusoidal, triangular e quadrada

Na Figura 103 é possível observar o diagrama de Bode para uma onda sinusoidal de um filtro IIR com α=-0.5 (Integrador), T=0.000125 s, $f \cong 1040$ Hz, aproximação de Tustin onde o valor da fase para essa frequência é de -45º. Na Figura 104 é possível observar o resultado experimental anterior mas considerando a onda sinusoidal, triangular e quadrada. Verifica-se que para a onda a sinusoidal e onda triangular a fase aproxima-se do valor esperado (Figura 103).

Na Figura 105 é possível visualizar o diagrama de Bode para o caso anterior mas agora considerando a frequência de f=266 Hz e o valor de α=+0.5 (Diferenciador). Para essa frequência o valor da fase assume +45º.

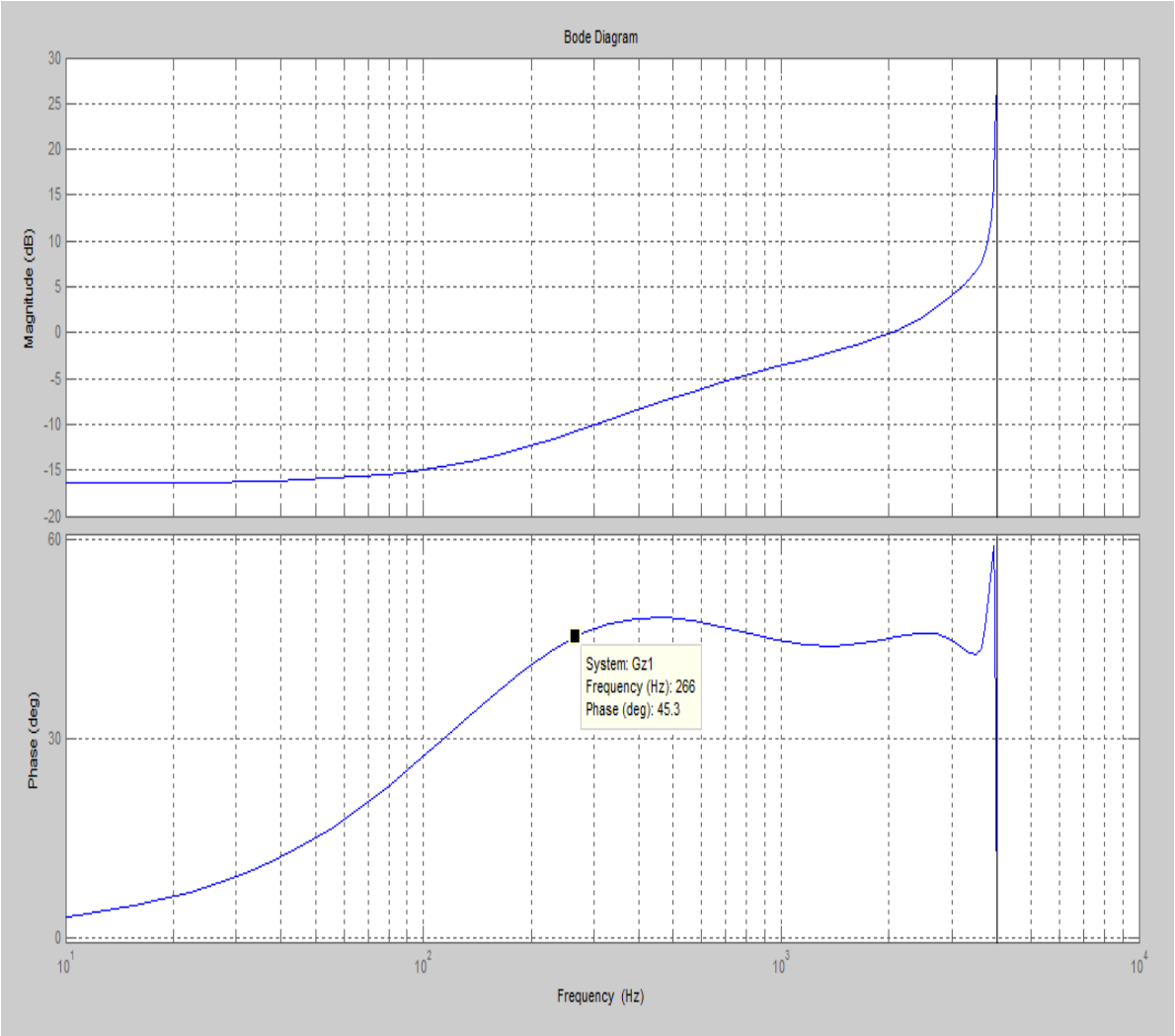

Figura 105: Diagramas de Bode de um filtro IIR com α=0.5, T=0.000125 s, OrdN=OrdD=3, $f \cong 266$ Hz, aproximação de Tustin (Diferenciador)

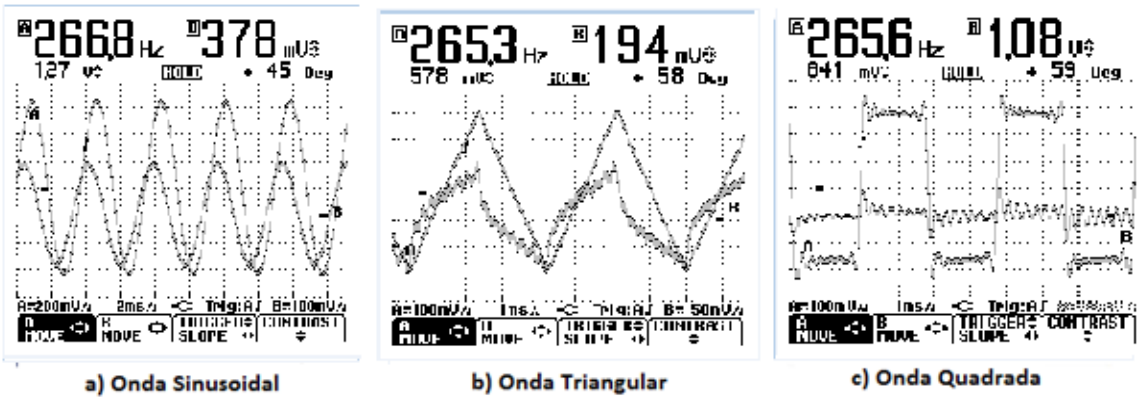

Figura 106: Resultado experimental para um filtro IIR com α=0.5, T=0.000125 s, OrdN=OrdD=3, $f\cong$ 266 Hz, aproximação de Tustin (Diferenciador) de uma onda sinusoidal, triangular e quadrada

Nesta experiência é possível analisar que, tanto para o caso de um filtro diferenciador ou integrador, quando o sinal de entrada é alterado de uma sinusoide para uma triangular o valor de desfasamento é alterado e diferente do valor pretendido. No entanto, para uma onda quadrada é possível notar que o desfasamento atinge valores muito diferentes. No entanto, verifica-se que para todos os casos do filtro IIR está derivar/integrar de forma correta.

4.3.3. EXEMPLOS DE ESTRUTURA DE UM FILTRO IIR

Nesta secção são apresentados alguns exemplos da implementação de um filtro IIR com uma topologia em cascata e paralelo.

4.3.3.1. ESTRUTURA CASCATA

A Figura 107 ilustra o diagrama de Bode para o filtro IIR integrador com α=-0.5, T=1/8000=0.000125 s, método de Prony com 1000 amostras, ordem do numerador e denominador igual a 4 e utilizando a aproximação de Tustin para uma estrutura em cascata. Para a frequência de 165 Hz o valor da fase é de -45º.

$$H_I(z) = \frac{1 + 0.06926z^{-1} - 1.037z^{-2} - 0.04331z^{-3} + 0.1479z^{-4}}{1 - 0.9307z^{-1} - 0.606z^{-2} + 0.3281z^{-3} + 0.01318z^{-4}} \tag{4.1}$$

$$H_{2,1}(z) = \frac{1 + 1.38z^{-1} - 0.4114z^{-2}}{1 + 0.7822z^{-1} + 0.01842z^{-2}} \tag{4.2}$$

$$H_{2,2}(z) = \frac{1 - 1.31z^{-1} + 0.3594z^{-2}}{1 - 1.713z^{-1} - 0.7154z^{-2}} \tag{4.3}$$

$$Ganho = 1 \tag{4.4}$$

Na equação (4.1) é possível visualizar a função de transferência do filtro apresentado para a ordem do numerador e denominador igual a 4. Sendo as restantes equações (4.2), (4.3) e (4.4) as expressões obtidas através da função `dir2cas` desenvolvida para converter estrutura direta para estrutura em cascata. Na Figura 108 é mostrado o resultado da implementação no DSP onde se constata que o valor da fase é de aproximadamente -45º, como seria de esperar (Figura 107).

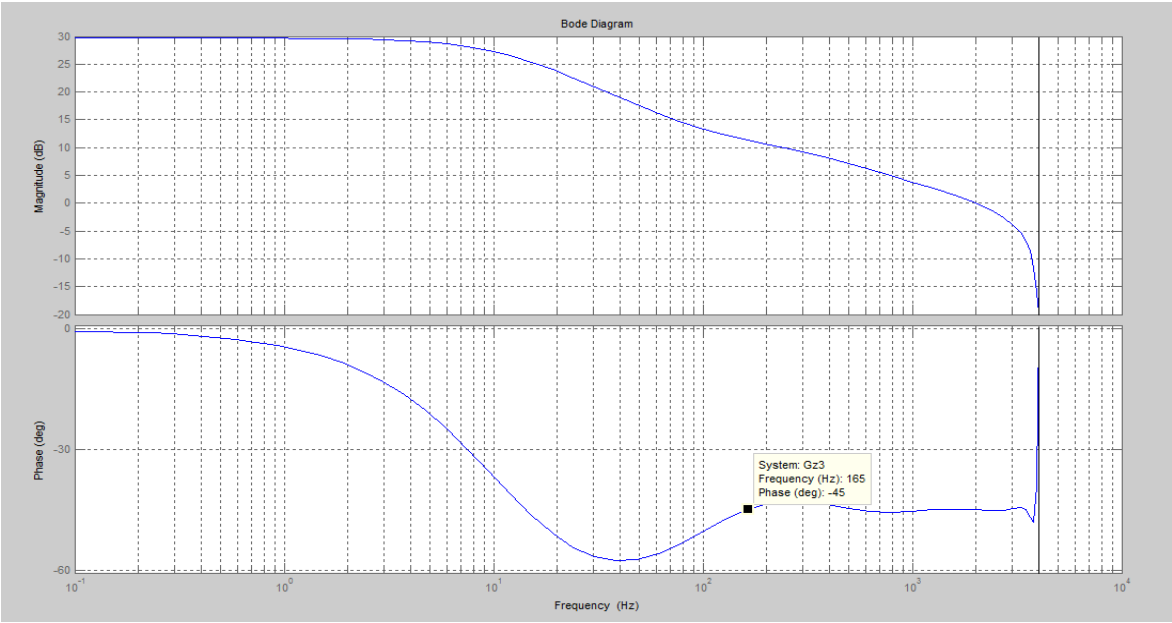

Figura 107: Diagramas de Bode de um filtro IIR com α=-0.5, T=0.000125 s, $f \cong 165$ Hz, aproximação de Tustin e estrutura cascata (2,2)x(2,2) (Integrador)

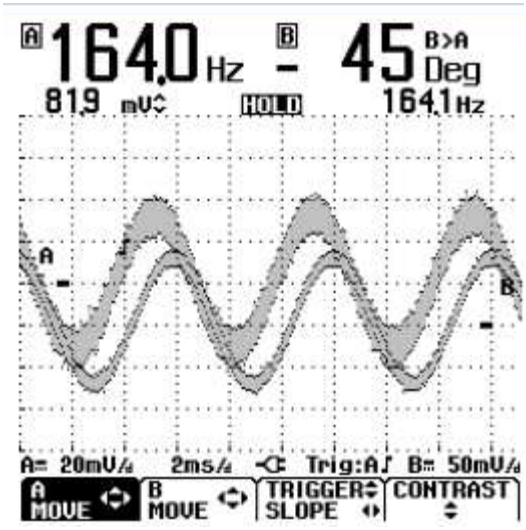

Figura 108: Resultado experimental para um filtro IIR com α=-0.5, T=0.000125 s, $f \cong 165$ Hz, aproximação de Tustin e estrutura cascata (2,2)x(2,2) (Integrador)

$$H_I(z) = \frac{1 - 0.06926z^{-1} - 1.037z^{-2} + 0.04331z^{-3} + 0.1479z^{-4}}{1 + 0.9307z^{-1} - 0.606z^{-2} + 0.5281z^{-3} + 0.01318z^{-4}} \tag{4.5}$$

$$H_{2,1}(z) = \frac{1 + 1.31z^{-1} + 0.3594z^{-2}}{1 + 1.713z^{-1} + 0.7154z^{-2}} \tag{4.6}$$

$$H_{2,2}(z) = \frac{1 - 1.38z^{-1} + 0.4114z^{-2}}{1 - 0.7822z^{-1} - 0.01842z^{-2}} \tag{4.7}$$

$$Ganho = 1 \tag{4.8}$$

As Figuras 109 e 110 ilustram a situação anterior para um filtro IIR diferenciador com α=0.5 e os mesmos parâmetros do integrador, baseadas nas equações anteriores (4.5)-(4.8). Sendo a equação (4.5) a função de transferência do filtro apresentado para a ordem do numerador e denominador igual a 4, e as restantes equações (4.6), (4.7) e (4.8) as equações obtidas através da função dir2cas desenvolvida para converter estrutura direta para estrutura em cascata. Através da observação da Figura 109 é possível analisar que para o valor da frequência 419 Hz a fase assume o valor de +45.7º.

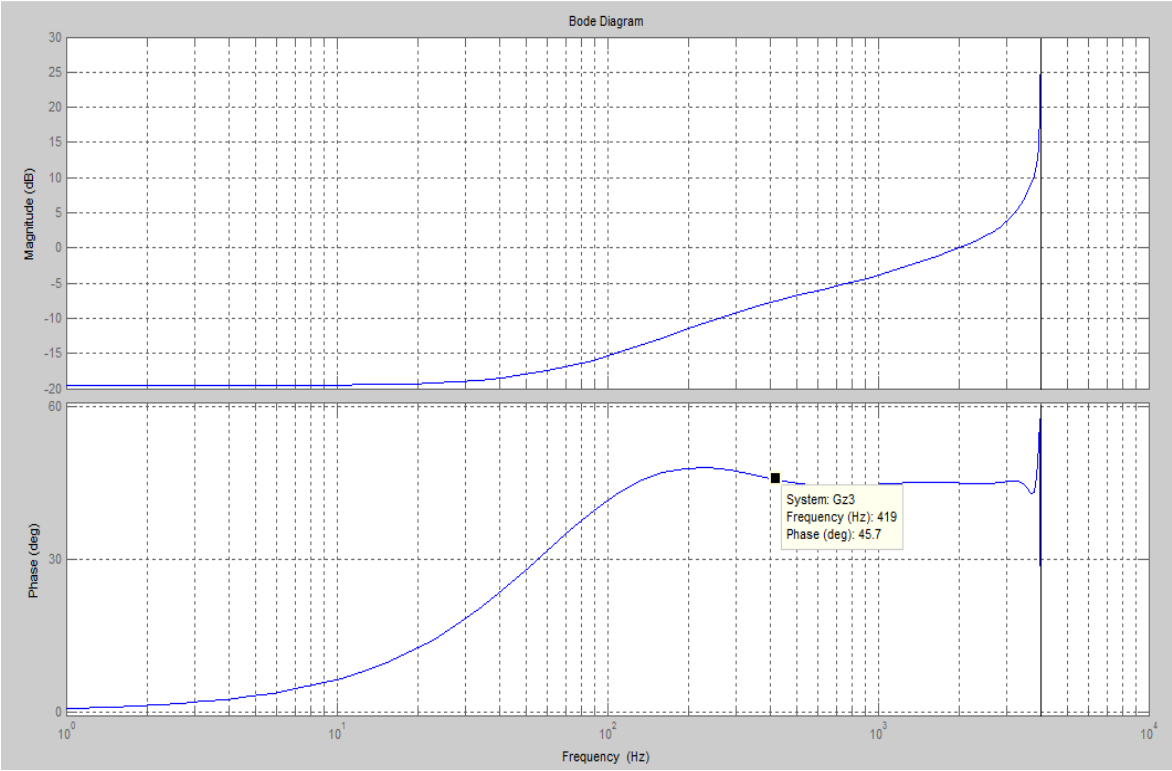

Figura 109: Diagramas de Bode de um filtro IIR com α=+0.5, T=0.000125 s, *f*≅ 419 Hz, aproximação de Tustin e estrutura cascata (2,2)x(2,2) (Diferenciador)

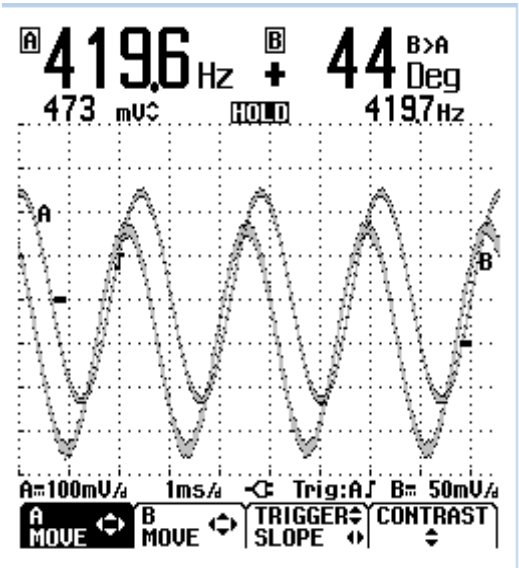

Figura 110: Resultado experimental para um filtro IIR com α=+0.5, T=0.000125 s, *f*≅ 419 Hz, aproximação de Tustin e estrutura cascata (2.2)x(2.2) (Diferenciador)

Através da análise das Figuras 107, 108, 109, 110 é possível verificar que os valores obtidos experimentalmente coincidem com os valores obtidos por simulação no MatLab.

4.3.3.1. ESTRUTURA PARALELO

Tal como a estrutura em cascata, a Figura 111 ilustra o diagrama de Bode para o filtro IIR integrador com α=-0.5, T=1/8000=0.000125 s , método de Prony para 50 amostras, ordem do numerador e denominador igual a 4 e utilizando a aproximação de Tustin para uma estrutura em paralelo.

$$H_I(z) = \frac{1 + 0.20436z^{-1} - 0.9784z^{-2} - 0.1242z^{-3} + 0.1291z^{-4}}{1 + 0.7957z^{-1} - 0.6827z^{-2} + 0.4563z^{-3} + 0.03701z^{-4}} \tag{4.9}$$

$$H_{2,1}(z) = \frac{-3.394 - 2.571z^{-1}}{1 + 0.8462z^{-1} + 0.05696z^{-2}} \tag{4.10}$$

$$H_{2,2}(z) = \frac{0.9057 - 0.7872z^{-1}}{1 - 1.642z^{-1} + 0.6497z^{-2}} \tag{4.11}$$

$$Ganho = 4 \tag{4.12}$$

A equação (4.9) é a função de transferência do filtro apresentado para a ordem do numerador e denominador igual a 4, e as restantes equações (4.10), (4.11) e (4.12) as equações obtidas através da função `dir2par` desenvolvida para converter estrutura direta para estrutura em paralelo. Para a frequência de 102 Hz o valor da fase é -45.5º. Na Figura 112 é mostrado o resultado da implementação no DSP onde se constata que o valor da fase é aproximadamente de -46º, como seria de esperar (Figura 111).

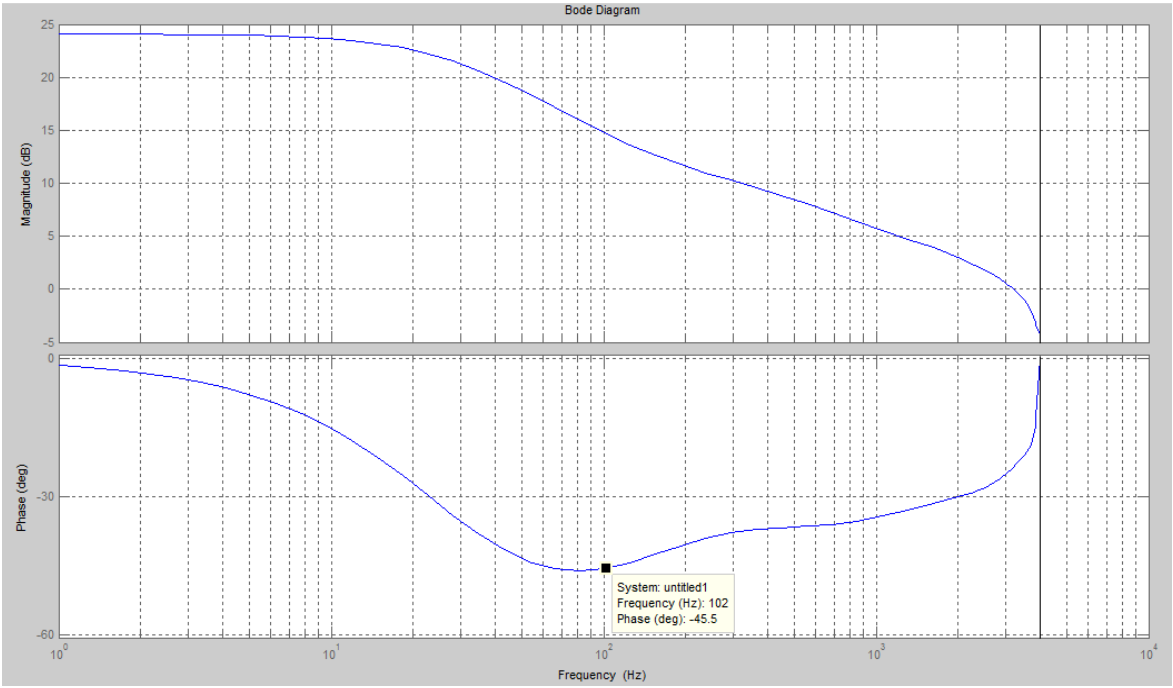

Figura 111: Diagramas de Bode de um filtro IIR com α=-0.5, T=0.000125 s, $f \cong$ 102 Hz, aproximação de Tustin e estrutura paralelo (2,2)x(2,2) (Integrador)

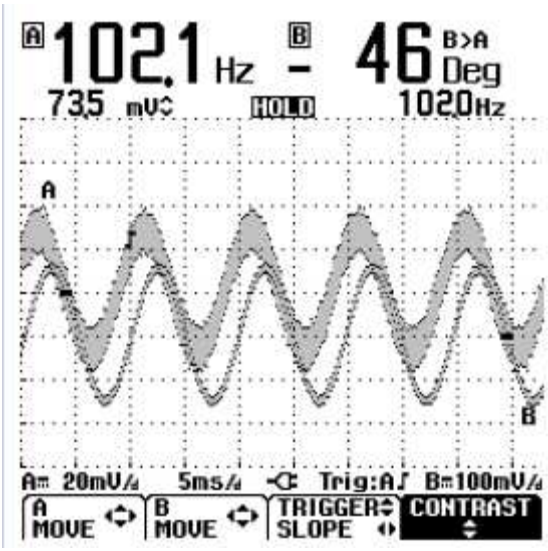

Figura 112: Resultado experimental para filtro IIR com α=-0.5, T=0.000125 s, $f \cong$ 102 Hz, aproximação de Tustin e estrutura paralelo (2.2)x(2.2) (Integrador)

As Figuras 113 e 114 ilustram a situação anterior para um filtro IIR diferenciador com α=0.5, *f*=1090 Hz, método de Prony para 65 amostras e os restantes parâmetros iguais ao integrador.

$$H_I(z) = \frac{1 - 0.1818z^{-1} - 0.9895z^{-2} + 0.1107z^{-3} + 0.1326z^{-4}}{1 + 0.8192z^{-1} - 0.6703z^{-2} - 0.4693z^{-3} + 0.03314z^{-4}} \tag{4.13}$$

$$H_{2,1}(z) = \frac{0.8934 + 0.7829z^{-1}}{1 + 1.654z^{-1} + 0.6612z^{-2}} \tag{4.14}$$

$$H_{2,2}(z) = \frac{-3.896 + 2.949z^{-1}}{1 - 8351z^{-1} + 0.05012z^{-2}} \tag{4.15}$$

$$Ganho = 4 \tag{4.16}$$

Sendo a equação (4.13) a função de transferência do filtro apresentado para a ordem do numerador e denominador igual a 4, e as restantes equações (4.14), (4.15) e (4.16) são as equações obtidas através da função `dir2par` desenvolvida para converter estrutura direta para estrutura em paralelo. Através da observação da Figura 113 é possível verificar que para o valor da frequência de 1090 Hz a fase assume o valor de +45º.

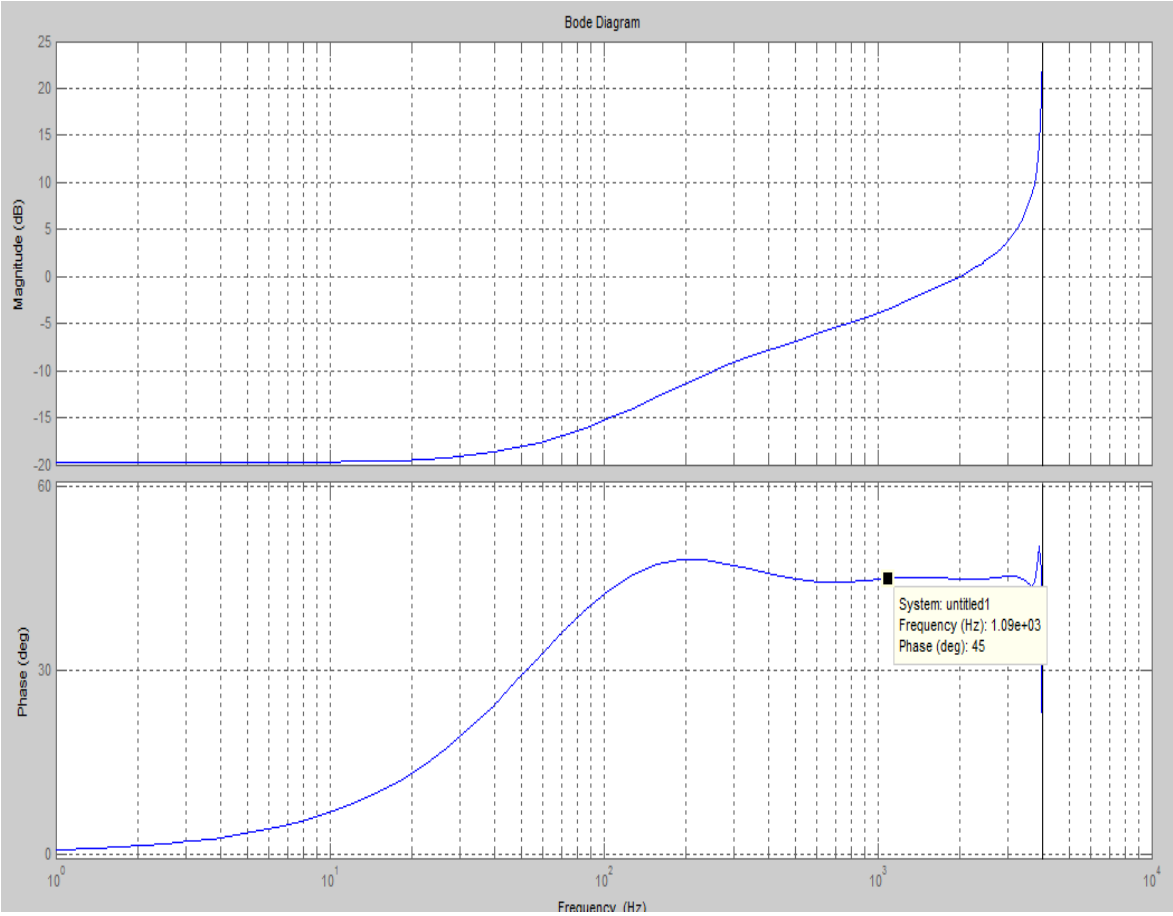

Figura 113: Diagramas de Bode de um filtro IIR com α=+0.5, T=0.000125 s, $f \cong 1090\ Hz$, aproximação de Tustin e estrutura paralelo (2,2)x(2,2)(Diferenciador)

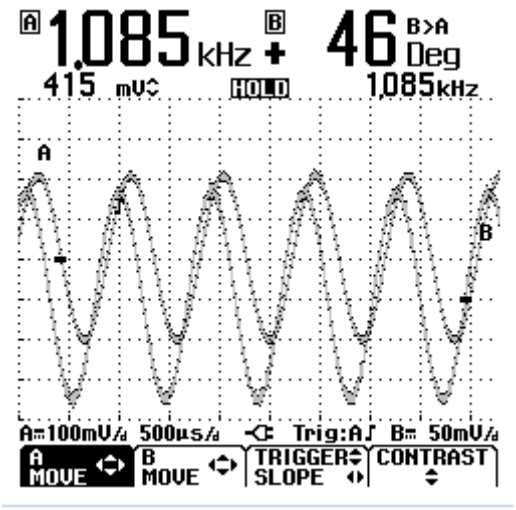

Figura 114: Resultado experimental para um filtro IIR com α=+0.5, T=0.000125 s, $f \cong 1090\ Hz$, aproximação de Tustin e estrutura paralelo (2,2)x(2,2) (Diferenciador)

Através da análise das Figuras 111, 112, 113 e 114 é possível verificar que os valores obtidos experimentalmente coincidem com os valores obtidos por simulação no MatLab.

5. CONCLUSÕES

Esta Dissertação teve como objetivo o estudo e aplicação das derivadas e integrais de ordem não inteira na área do processamento digital de sinal, nomeadamente no projeto de filtros digitais fraccionários.

Os filtros ideais fraccionários são caracterizados por terem uma amplitude com um declive dado por α20 dB/dec e por uma fase dada por $\alpha\pi/2$, em que $\alpha \in \mathcal{R}$. Os filtros reais só aproximam-se dos filtros ideais para uma faixa de frequências limitada. Mas estas características variam se o filtro digital é um IIR ou um FIR e do tipo de aproximação escolhida: Euler, Tustin ou Al-Alaoui, entre outras.

A utilização do MatLab foi um ponto essencial no desenvolvimento da Tese porque com o auxílio das suas potencialidades foi possível desenvolver o código necessário para a implementação do projeto e ainda a utilização da ferramenta GUIDE demonstrou ser de bastante utilidade para o desenvolvimento da interface gráfica. Através dessa interface gráfica foi possível efetuar a implementação de filtros digitais e a comparação/análise dos resultados obtidos experimentalmente. Esta fase foi muito desafiante mas simples devido a existir um conhecimento prévio sobre esta ferramenta do Matlab. Apesar dos conhecimentos prévios sobre essa ferramenta e do MatLab esta fase foi muito demorada

porque além da construção da interface gráfica teve-se de a interligar com a implementação dos filtros digitais fraccionários.

Após a implementação da interface foi possível analisar os resultados simulados, onde é possível evidenciar que o filtro digital FIR é sempre estável mas para a implementação do mesmo é necessário um grande número de amostras, enquanto no filtro IIR é necessário um menor número de amostras. Através da interface gráfica foi possível também a implementação de filtros digitais fraccionários (FIR e IIR) com topologias diferentes (cascata e paralelo) ao atualmente utilizado (direto).

Um outro objetivo da Dissertação, foi a comparação dos resultados simulados com os resultados práticos. Esta fase incidiu na implementação dos filtros IIR e FIR fraccionários em estudo utilizando a placa ADSP-2181-Kit Analog Devices (Ez-kit Lite). Esta fase foi a que colocou mais dificuldades, visto que não existia conhecimento prévio sobre o DSP. Mas com esforço e dedicação foi atingido o objetivo pretendido de implementar filtros digitais fraccionários em ambiente prático. Pode-se afirmar que os resultados obtidos na prática foram de acordo com o esperado (simulado) apesar do próprio DSP apresentar às vezes, em desvio no desfasamento de 1º/2º na fase.

Como futuros desenvolvimentos, deve ser feita uma análise mais pormenorizada sobre a limitação encontrada para a implementação de filtros IIR. Quando efetuado a implementação prática do filtro IIR em DSP foi encontrada a barreira de não poder ultrapassar o valor da ordem do numerador e denominador superior a cinco. Outra melhoria seria a alteração do código MatLab para as estruturas de modo que estas permitam blocos de ordem superior a dois.

Referências Documentais

[1] Hsu, Hwei P.- *Sinais e Sistemas*, Coleção Schaum, 1995. ISBN 0-07-030641-9

[2] Tan, Li- Digital *Signal Processing: Fundamentals and Applications*, Academic Press, Elsevier, 2008. ISBN 978-0-12-374090-8

[3] Porat,Boaz- *A Course in Digital Signal Processing*, John Wiley& Sons, Inc, 1997. ISBN 0-471-14961-6

[4] Diniz, Paulo; Silva, Eduardo; Netto, Sergio- *Digital Signal Processing: System Analysis and Design*, 2ª Ed, Cambridge University Press. ISBN 978-0-511-78983-0

[5] Hayes, Monson H. - *Statistical Digital Signal Processing and Modeling*, John Wiley&Sons, inc. ISBN 0-471-59431-8

[6] Ingle, Vinay K & Proakis, John G. – *Digital Signal Processing using Matlab*, Bookware Companion Series, 2ª ed International Student Edition, 2007. ISBN 0-495-24441-4

[7] Sophocles J. Orfanidis – *ADSP-2181 Experiments*, 2005.

[8] Taylor, Fred J. – *Digital Filters: Principles and Applications with MatLab*, A John Wiley & Sons, 2012. ISBN 978-1-118-14115-1

[9] Hussain,Zahir M & Sadik, Amin Z. & O'Shea, Peter – *Digital Signal Processing: An Introduction with MatLab and Application*, Springer, 2011. ISBN 978-3-642-15590-1

[10] Proakis, John G.&. Manolakis, Dimitrus G – *Digital Signal Processing, Principles, Algoritms, and Applications*. 3ºed, Prentice Hall, 1996. ISBN 0-13-394338-9

[11] Antoniou, Andreas – *Digital Signal Processing: Signals, Systems, And Filters*, McGraw-Hill, 2006. ISBN 0-07-145424-1

[12] Baese, Uwe Meyer – *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, 2007. ISBN 978-3-540-72612-8

[13] EIALI, Taan S. – *Discrete Systems and Digital Signal Processing with MatLab*, CRC Press, 2004. ISBN 0-203-48711-7

[14] Neto, Fernando Gonçalves de Almeida – *Análise de Filtros Digitais Implementados em Aritmética de Ponto Fixo usando Cadeias de Markov*, Dissertação de Mestrado Universidade de São Paulo, 2011.

[15] Pinheiro, Antonelo António Lopes – *Projeto de Filtros Digitais de Ordem Fraccionária*, Projeto de Licenciatura, Instituto Superior de Engenharia do Porto, 2007.

[16] Madisetti, Vijay K. & Williams, Douglas B. – *Digital Signal Processing: Handbook*, Chapmam & Hall, 1999.

[17] Kuc, Romam – *Introduction to Digital Signal Processing*, Bs Publications, 2008. ISBN 81-7800-168-3

[18] Leis, John W. – *Digital Signal Processing Using MatLab for Students and Researchers*, Wiley, 2011. ISBN 978-0-470-88091-3

[19] Rorabaugh, C. Britton – *Notes on Digital Signal Processing: Pratical Recipes for Design, Analysis, and Implementation*, Prentice Hall, 2011. ISBN 978-0-13-158334-4

[20] Barbosa, Ramiro Sousa – *Analise Dinâmica e Controlo de Sistemas de Ordem Faccionaria*, Dissertação de Doutoramento, Instituto Superior de Engenharia do Porto, 2005.

[21] Smith, Seten W. – *The Scientist and Engineer's Guide to Digital Signal Processing*, *California Technical*, 1999. ISBN 0-9660176-7-6

[22] Analog Devices – *Digital Signal Processing Applications using the ADSP-2100 Family*, Prentice Hall, 1990.

[23] Shenoi, B. A. – *Introduction to Digital Signal Processing and Filter Design*, Wiley-Interscience, 2006. ISBN 978-0-471-46482-2

[24] Analog Devices – *ADSP-2100 Family:Assembler Tools& Simulator Manual*, Analog Devices, 1994.

[25] Analog Devices – *ADSP-2100 Family:User's Manual*, Analog Devices, 1995.

[26] Oldham, K.B. & Spanier J. – *The Fractional Calculus*, Academic Press, New York, 1974.

[27] Podlubny, J. – *Fractional Differential Equations*, Academic Press, San Diego, 1999.

[28] Machado, J. A. T. – *Analysis and design of fractional-order digital control systems*, J. Syst. Anal, Model, Simul. 27, 1997, 107-122.

[29] Vinagre, B. M. & Podlubny, I. & Hernández, A. & Feliu, V. – *Some approximations of fractional order operators used in control theory and applications*, Frac. Calculus Appl. 49 (3), 2000, 231-248.

[30] Chen, Y. Q. & Moore, K. L. – *Discretization schemes for fractional-order differentiators and integrators*, IEEE Trans. Circuits and Systems, Fundam. Theory Appl. 49 (3), 2002, 363-367.

[31] Al-Alaoui, M. A. – *Novel digital integrator and differentiator*, Electron. Lett. 29 (4), 1993, 376-378.

[32] Vinagre, B.M. & Chen, Y. Q. & Petras, I – *Two direct Tustin discretization methods for fractional-order differentiator/intregrator*, J. Franklin Inst 340, 2003, 349-362.

[33] Chen, Y. Q. & Vinagre, B. M. & Podlubny – *Continued fraction expansion approaches to discretizing fractional order derivates-an expository review*, Nonlinear Dyn. 38, 2004, 155-170.

[34] Machado, J. A. T – *Discrete-time fractional-order controllers*, Fract, Calculus Appl. Anal. 4(1) , 2001, 47-66.

[35] Barbosa, R. S. & Machado, J. A. T. & Ferreira, I. M. – *Pole-zero approximations of digital fractional-order integrators and differentiators using signal modeling techniques,* in: 16th IFAC, World Congress, Prague, Czech Republic, July, 2005.

[36] Barbosa, Ramiro & Machado, J. A. Tenreiro & Silva, Manuel F. – *Time domain design of fractional differintegrators using least-squares*, Signal Processing, vol. 86, 2006, 2567-2581.

[37] Analog Devices – *DSP Microcomputer: ADSP-218xN Series Datasheet*, 2006.

[38] Analog Devices – *DSP Microcomputer: ADSP-2181 Datasheet*, 2012.

[39] Topward – *8110 Function Generator Operation Manual Datasheet,* 1996.

[40] Fluke Corporation – *ScopeMeter 120 and 190 Series Datasheet, 2012*.

Anexo A. Funções MatLab

Bode.m

```
function
[m1_dB,m3_dB,m5_dB,m7_dB,m9_dB,f1,f3,f5,f7,f9,w,mi_dB,fi]   =
Bode(Gz1,Gz3,Gz5,Gz7,Gz9,OrdP)
global HE H1 F1 hE m1_dB m3_dB m5_dB m7_dB m9_dB f1 f3 f5 f7
f9 w mi_dB fi Gz1 Gz3 Gz5 Gz7 Gz9
%---- Diagramas de Bode
[m9,f9,w]=bode(Gz9); m9_dB=20*log10(m9(:));
[m1,f1]=bode(Gz1,w); m1_dB=20*log10(m1(:));
[m3,f3]=bode(Gz3,w); m3_dB=20*log10(m3(:));
[m5,f5]=bode(Gz5,w); m5_dB=20*log10(m5(:));
[m7,f7]=bode(Gz7,w); m7_dB=20*log10(m7(:));
Gi=(sqrt(-1)*w).^OrdP; % Ideal: s^q
mi_dB=20*log10(abs(Gi)); fi=angle(Gi)*180/pi;
```

Convm.m

```
function X=convm(x,p)
%
%   X=convm(x,p)
%
% Esta funcao gera uma matriz de convolucao
% Livro do Monson H. Hayes, pag.572
%
N=length(x)+2*p-2;
x=x(:);
xpad=[zeros(p-1,1);x;zeros(p-1,1)];
for i=1:p
    X(:,i)=xpad(p-i+1:N-i+1);
end
```

dir2cas.m

```
function [b0,B,A] = dir2cas(b,a);
% DIRECT-form to CASCADE-form conversion (cplxpair version)
% ------------------------------------------------------
% [b0,B,A] = dir2cas(b,a)
% b0 = gain coefficient
% B = K by 3 matrix of real coefficients containing bk's
% A = K by 3 matrix of real coefficients containing ak's
% b = numerator polynomial coefficients of DIRECT form
% a = denominator polynomial coefficients of DIRECT form
% compute gain coefficient b0
b0 = b(1); b = b/b0; a0 = a(1); a = a/a0; b0 = b0/a0;
%
M = length(b); N = length(a);
if N > M
b = [b zeros(1,N-M)];
elseif M > N
a = [a zeros(1,M-N)]; N = M;
else
NM = 0;
end
%
K = floor(N/2); B = zeros(K,3); A = zeros(K,3);
if K*2 == N;
b = [b 0]; a = [a 0];
end
%
broots = cplxpair(roots(b)); aroots = cplxpair(roots(a));
for i=1:2:2*K
Brow = broots(i:1:i+1,:); Brow = real(poly(Brow));
B(fix((i+1)/2),:) = Brow;
Arow = aroots(i:1:i+1,:); Arow = real(poly(Arow));
A(fix((i+1)/2),:) = Arow;
end
```

dir2par.m

```
function [C,B,A] = dir2par(b,a);
% DIRECT-form to PARALLEL-form conversion
% ---------------------------------------
% [C,B,A] = dir2par(b,a)
%  C = Polynomial part when length(b) >= length(a)
%  B = K by 2 matrix of real coefficients containing bk's
%  A = K by 3 matrix of real coefficients containing ak's
%  b = numerator polynomial coefficients of DIRECT form
%  a = denominator polynomial coefficients of DIRECT form
%
M = length(b); N = length(a);
[r1,p1,C] = residuez(b,a);
p = cplxpair(p1,10000000*eps);
I = cplxcomp(p1,p);
r = r1(I);
K = floor(N/2); B = zeros(K,2); A = zeros(K,3);
if K*2 == N; %N even, order of A(z) odd, one factor is first
order
     for i=1:2:N-2
          Brow = r(i:1:i+1,:);
          Arow = p(i:1:i+1,:);
          [Brow,Arow] = residuez(Brow,Arow,[]);
          B(fix((i+1)/2),:) = real(Brow');
          A(fix((i+1)/2),:) = real(Arow');
     end
     [Brow,Arow] = residuez(r(N-1),p(N-1),[]);
     B(K,:) = [real(Brow') 0]; A(K,:) = [real(Arow') 0];
else
        for i=1:2:N-1
          Brow = r(i:1:i+1,:);
          Arow = p(i:1:i+1,:);
          [Brow,Arow] = residuez(Brow,Arow,[]);
          B(fix((i+1)/2),:) = real(Brow');
```

```
          A(fix((i+1)/2),:) = real(Arow');
     end
```

cplxcomp.m

```
function I = cplxcomp(p1,p2)
%  I = cplxcomp(p1,p2)
% Compares two complex pairs which contain the same scalar
elements
%  but (possibly) at differrent indices.  This routine should
be
%  used after CPLXPAIR routine for rearranging pole vector
and its
%  corresponding residue vector.
%      p2 = cplxpair(p1)
%
I=[];
for j=1:1:length(p2)
    for i=1:1:length(p1)
     if (abs(p1(i)-p2(j)) < 0.0001)
        I=[I,i];
        end
    end
end
I=I';
```

FIR_Alaoui.m

```
function [HA,H3,F3,hA] = FIR_Alaoui(OrdP,N,T)


hA=zeros(1,N);
hA(1)=1;
for k=2:N
    % Binomio C1
    C1=zeros(1,k);
    C1(1)=1;
    for i=2:k
```

```
        C1(i)=(1-(OrdP+1)/(i-1))*C1(i-1);
    end
    % Binomio C2
    C2=zeros(1,k);
    C2_aux=zeros(1,k);
    C2(k)=1;
    C2_aux(k)=1;
    for i=k-1:-1:1
        C2_aux(i)=(1/7)^(k-i);
        C2(i)=((-OrdP-k+i+1)/(k-i))*C2(i+1);
    end
    C2=C2.*C2_aux;
    hA(k)=sum(C1.*C2);
end
hA=(8/(7*T))^OrdP*hA; % multiplica constante
HA=filt(hA,1,T);
Fs=1/T;
[H3,F3]=freqz(hA,1,1024,Fs);
```

FIR_EULER.m

```
function [HE,H1,F1,hE] = FIR_EULER(OrdP,N,T)
%Resposta impulsional do operador de Euler:
%HE(z)=[(1-z^-1)/T]^q
sym hE;
hE=zeros(1,N);
hE(1)=1;
for k=2:N
    hE(k)=(1-(OrdP+1)./(k-1)).*hE(k-1);
end
hE=(1/T)^OrdP*hE; % multiplica constante
HE=filt(hE,1,T);
Fs=1/T;
[H1,F1]=freqz(hE,1,1024,Fs);
```

FIR_TUSTIN.m

```
function [HT,H2,F2,hT] = FIR_TUSTIN(OrdP,N,T)

hT=zeros(1,N);
hT(1)=1;
for k=2:N
    % Binomio C1
    C1=zeros(1,k);
    C1(1)=1;
    for i=2:k
        C1(i)=(1-(OrdP+1)/(i-1))*C1(i-1);
    end
    % Binomio C2
    C2=zeros(1,k);
    C2(k)=1;
    for i=k-1:-1:1
        C2(i)=((-OrdP-k+i+1)/(k-i))*C2(i+1);
    end
    hT(k)=sum(C1.*C2);
end
hT=(2/T)^OrdP*hT; % multiplica constante
HT=filt(hT,1,T);
Fs=1/T;
[H2,F2]=freqz(hT,1,1024,Fs);
```

GraficoBode.m

```
function
GraficoBode(m1_dB,m3_dB,m5_dB,m7_dB,m9_dB,f1,f3,f5,f7,f9,w,mi
_dB,fi)
global m1_dB m3_dB m5_dB m7_dB m9_dB f1 f3 f5 f7 f9 w mi_dB
fi
axes(handles.axes1);
semilogx(w,m1_dB,'k','linewidth',1), hold on
semilogx(w,m3_dB,'k','linewidth',1)
semilogx(w,m5_dB,'k','linewidth',1)
```

```
semilogx(w,m7_dB,'k','linewidth',1)
semilogx(w,m9_dB,'k','linewidth',1)
semilogx(w,mi_dB,'k:','linewidth',1)
Hx=xlabel('\omega (rad s^{-1})'); Hy=ylabel('Amplitude
(dB)');
set(Hx,'FontSize',12), set(Hy,'FontSize',12)
set(gca,'FontSize',12)
axes(handles.axes3);
semilogx(w,f1(:),'k','linewidth',1), hold on
semilogx(w,f3(:),'k','linewidth',1)
semilogx(w,f5(:),'k','linewidth',1)
semilogx(w,f7(:),'k','linewidth',1)
semilogx(w,f9(:),'k','linewidth',1)
semilogx(w,fi,'k:','linewidth',1)
Hx=xlabel('\omega (rad s^{-1})'); Hy=ylabel('Fase (º)');
set(Hx,'FontSize',12), set(Hy,'FontSize',12),
set(gca,'FontSize',12)
ax_Y=get(gca,'ytick'); plot([max(w) max(w)],[min(ax_Y)
max(ax_Y)],'k')
```

IIR_Alaoui.m

```
function [HA,H3,F3,hA] =
IIR_Alaoui(OrdP,OrdNumerador,OrdDenominador,T)
N=OrdNumerador+OrdDenominador+1;
hA=zeros(1,N);
hA(1)=1;
for k=2:N
    % Binomio C1
    C1=zeros(1,k);
    C1(1)=1;
    for i=2:k
        C1(i)=(1-(OrdP+1)/(i-1))*C1(i-1);
    end
    % Binomio C2
    C2=zeros(1,k);
```

```
    C2_aux=zeros(1,k);
    C2(k)=1;
    C2_aux(k)=1;
    for i=k-1:-1:1
        C2_aux(i)=(1/7)^(k-i);
        C2(i)=((-OrdP-k+i+1)/(k-i))*C2(i+1);
    end
    C2=C2.*C2_aux;
    hA(k)=sum(C1.*C2);
end
hA=(8/(7*T))^OrdP*hA; % multiplica constante
hA2=hA(1:N);
[ba,aa]=prony(hA2,OrdNumerador,OrdDenominador);
HA=filt(ba,aa,T);
Fs=1/T;
[H3,F3]=freqz(ba,aa,1024,Fs);
```

IIR_EULER.m

```
function [HE,H1,F1,hE]=
IIR_EULER(OrdP,OrdNumerador,OrdDenominador,T)
global HE H1 F1 hE m1_dB m3_dB m5_dB m7_dB m9_dB f1 f3 f5 f7
f9 w mi_dB fi Gz1 Gz3 Gz5 Gz7 Gz9
global N hE2 be ae
N=OrdNumerador+OrdDenominador+1;
hE=zeros(1,N);
hE(1)=1;
for k=2:N
    hE(k)=(1-(OrdP+1)./(k-1)).*hE(k-1);
end
hE=(1/T)^OrdP*hE; % multiplica constante
hE2=hE(1:N);
[be,ae]=prony(hE2,OrdNumerador,OrdDenominador)
HE=filt(be,ae,T);
Fs=1/T;
[H1,F1]=freqz(be,ae,1024,Fs);
```

IIR_Pade.m

```
function [Gz1,Gz3,Gz5,Gz7,Gz9,a1,a3,a5,a7,a9,b1,b3,b5,b7,b9]
= IIR_Pade(hE,T)
%---- Funcoes de transferencia
% % m=n=1
[a1,b1]=pade(hE,1,1); Gz1=tf(b1',a1',T);
% % m=n=3
[a3,b3]=pade(hE,3,3); Gz3=tf(b3',a3',T);
% % m=n=5
[a5,b5]=pade(hE,5,5); Gz5=tf(b5',a5',T);
%  % m=n=7
[a7,b7]=pade(hE,7,7); Gz7=tf(b7',a7',T);
% % m=n=9
[a9,b9]=pade(hE,9,9); Gz9=tf(b9',a9',T);
```

IIR_Prony_ls.m

```
%---- Funcoes de transferencia
% m=n=1
[b1,a1]=prony_ls(hE,1,1); Gz1=tf(b1',a1',T);
% m=n=3
[b3,a3]=prony_ls(hE,3,3); Gz3=tf(b3',a3',T);
% m=n=5
[b5,a5]=prony_ls(hE,5,5); Gz5=tf(b5',a5',T);
% m=n=7
[b7,a7]=prony_ls(hE,7,7); Gz7=tf(b7',a7',T);
% m=n=9
[b9,a9]=prony_ls(hE,9,9); Gz9=tf(b9',a9',T);
% m=n=5, Pade
[aP,bP]=pade(hE,5,5); GzP=tf(bP',aP',T);
```

IIR_Tustin.m

```
function [HT,H2,F2,hT] =
IIR_Tustin(OrdP,OrdNumerador,OrdDenominador,T)
global OrdP OrdNumerador OrdDenominador hT F2 H2 HT hT2
N=OrdNumerador+OrdDenominador+1;
```

```
hT=zeros(1,N);
hT(1)=1;
for k=2:N
    % Binomio C1
    C1=zeros(1,k);
    C1(1)=1;
    for i=2:k
        C1(i)=(1-(OrdP+1)/(i-1))*C1(i-1);
    end
    % Binomio C2
    C2=zeros(1,k);
    C2(k)=1;
    for i=k-1:-1:1
        C2(i)=((-OrdP-k+i+1)/(k-i))*C2(i+1);
    end
    hT(k)=sum(C1.*C2);
end
hT=(2/T)^OrdP*hT; % multiplica constante

hT2=hT(1:N);
[bt,at]=prony(hT2,OrdNumerador,OrdDenominador);
HT=filt(bt,at,T);
Fs=1/T;
[H2,F2]=freqz(bt,at,1024,Fs);
```

Pade.m

```
function [a,b]=pade(x,p,q)
%
%   [a,b]=pade(x,p,q)
%
% Aproximacao de Pade
% Livro do Monson H. Hayes, pag.138
%
x=x(:);
if p+q>=length(x)
```

```
%     error('Ordem do modelo demasiado elevada')
      msgbox('Ordem do modelo demasiado elevada, considerar
outros valores para a ordem do
Numerador/Denominador','Erro','error');
else
    X=convm(x,p+1);
    Xq=X(q+2:p+q+1,2:p+1);
    a=[1; -Xq\X(q+2:p+q+1,1)];
    b=X(1:q+1,1:p+1)*a;
end
```

prony_ls.m

```
function [b,a]=prony_ls(h,m,n)
%
%   [b,a]=prony_ls(h,m,n)
%
% Aproximacao de Prony
% Livro do Monson H. Hayes, Cap.4

h=h(:);
N=length(h);
% Denominador ak
x1=h(m+2:N);
X=zeros(N-m-1,n);
for k=m+2:N
    for i=1:n
        if i<k
            X(k-m-1,i)=h(k-i);
        end
    end
end
%size(X)
a=[1; -X\x1];
% Numerador bk
X0=toeplitz(h(1:m+1),[h(1) zeros(1,n)]);
```

```
b=X0*a;
```

Resposta_Impulsional.m

```
function [h1,h3,h5,h7,h9,Nk] =
Resposta_Impulsional(a1,a3,a5,a7,a9,b1,b3,b5,b7,b9)
 Nk=1000;
 h1=impz(b1,a1,Nk); % H1
 h3=impz(b3,a3,Nk);
 h5=impz(b5,a5,Nk);
 h7=impz(b7,a7,Nk);
 h9=impz(b9,a9,Nk); % H9
```

Step.m

```
function
[y1,y3,y5,y7,y9,yi,t]=Step(Gz1,Gz3,Gz5,Gz7,Gz9,OrdP,T)
t=(0:1000)*T;
y1=step(Gz1,t); % H1
y3=step(Gz3,t);
y5=step(Gz5,t);
y7=step(Gz7,t);
y9=step(Gz9,t); % H9
yi=t.^(-OrdP)./gamma(1-OrdP); % analitica
```

Anexo B. Interface Gráfica dos Filtros FIR e IIR

O código correspondente à interface gráfica de filtros FIR e IIR encontra-se em CD anexo a este relatório, com a designação de `Filtro_FIR.m` e `Filtro_IIR.m`.

Anexo C. Interface Gráfica das Estruturas

O código correspondente ao desenvolvimento das estruturas encontra-se em CD anexo a este relatório, com a designação de `Estruturas.m`, `Estruturas_FIR.m` e `Estruturas_IIR.m`.

Anexo D. Macros do DSP

O código correspondente às macros utilizadas na implementação dos filtros digitais encontra-se em CD anexo a este relatório, na pasta `macros`.