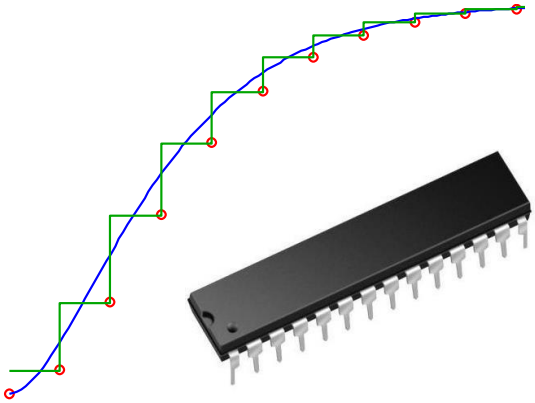
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Sistemas de Potência e Automação



Análise e Implementação de Filtros e Controladores num Sistema de Processamento Digital de Sinal

HENRIQUE JOSÉ SANTOS ANFILÓQUIO

Licenciado em Engenharia Eletrotécnica e Automação

Dissertação para a obtenção do grau de Mestre em Engenharia Eletrotécnica

Ramo de Automação e Eletrónica Industrial

Orientador:

Prof. Vasco Emanuel Anjos Soares

Júri:

Presidente: Prof. José Manuel do Valle Cardoso Igreja

Vogais: Prof. Vasco Emanuel Anjos Soares
Prof. Fernando Manuel Fernandes Melício

Dezembro de 2014

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Sistemas de Potência e Automação

Análise e Implementação de Filtros e Controladores num Sistema de Processamento Digital de Sinal

Dezembro de 2014

Resumo

Este trabalho visa a implementação prática de filtros e controladores digitais, comparando as suas respostas nos domínios contínuo e discreto.

A grande evolução tecnológica nas últimas décadas, bem como as exigências de miniaturização, foram fatores decisivos que contribuíram para a mudança do processamento e controlo analógico para a forma digital. Neste trabalho são estudadas cadeias de aquisição de sinal envolvidas em funções de filtragem ou controlo, recorrendo à utilização de um Microcontrolador (MCU) com características especificas de Processador Digital de Sinal (DSP).

Para a programação deste MCU apresentam-se diversos métodos de discretização, sendo avaliados os erros temporais associados aos mesmos e os seus limites de estabilidade. Em cada sistema a discretizar desenvolvem-se as equações às diferenças e os respetivos diagramas de fluxo de sinal, que possibilitam a programação do MCU em linguagem C.

No final deste trabalho realiza-se a simulação dos filtros e controladores em estudo, recorrendo a *software* apropriado, tanto no domínio contínuo como discreto, e comparam-se os resultados com uma implementação prática dos mesmos.

Abstract

This thesis aims to the implementation of digital filters and controllers and its comparison in continue and discrete time domains.

The great technology evolutions in the past decades, as well as the miniaturization demands, were determinative factors that contributed to the change of analog control and processing to the digital form. In this thesis, data acquisition systems are studied when used as filters or controllers, using a Microcontroller (MCU) with Digital Signal Processor (DSP) specific characteristics.

In order to program this MCU, many discretization methods are presented. Temporal error and stability issues are shown, difference equations are also derived, as well as the corresponding flow diagrams, both useful to program the device in C language.

At the end of this document, lies the simulation of the filters and controllers studied using a specific software, in both continuous and discrete domains, and a comparison with digital practical implementations.

Agradecimentos

Quero agradecer em primeiro lugar ao meu orientador, Prof. Vasco Soares, pela sugestão do tema, disponibilidade, paciência e imprescindível orientação ao longo do desenvolvimento deste trabalho.

Agradeço também aos meus colegas da empresa Alferpac, pelo apoio e acompanhamento que me deram, sem o qual o trabalho não teria a mesma diligência, em especial à direção por me ter dado a disponibilidade que necessitei ao longo do trabalho.

Aos meus colegas do ISEL, que me acompanharam e ajudaram ao longo do percurso académico.

Um agradecimento especial à minha família pela ajuda, acompanhamento e compreensão que sempre teve comigo, em especial ao meu Pai, Mãe e Irmão.

Aos meus amigos da Benedita pela presença e apoio, obrigado.

Índice

Índice de Figuras

Índice de Tabelas

Símbolos e Variáveis

$\|X^*(j\omega)\|$	-	Espectro de um sinal
γ	-	Margem de Fase
ω		Frequência angular
ω_b	-	Frequência angular associada à largura de banda
ω_c	-	Frequência angular de corte
ω_s	-	Frequência angular de amostragem
$a_0,\ldots,a_n$	-	Constantes 'a'
$b_0,\ldots,b_n$	-	Constantes 'b'
C	-	Condensador
e	-	Erro de quantificação
$\bar{e}^2$	-	Erro quadrático médio de quantificação
$f(e)$	-	Função densidade de probabilidade
f_c	-	Frequência de Corte
f_{in}	-	Frequência Interna
f_{osc}	-	Frequência do Oscilador
f_{Timer}	-	Frequência do Temporizador
$G(0)$	-	Ganho à componente contínua
$G(\omega_c)$	-	Ganho à frequência de corte
Gm	-	Margem de Ganho
g_n	-	Função do retentor de amostra
K_D	-	Ganho Derivativo
k_g	-	Margem de Ganho
K_I	-	Ganho Integral
K_P	-	Ganho Proporcional
n	-	Ordem de um sistema
Pm		Margem de Fase
P_n	-	Potência de ruído
P_s	-	Potência de um sinal
q	-	Resolução de um conversor A/D ou D/A
R	-	Resistência
R_n	-	Sinal convertido e com retenção de amostra
SNR	-	Relação Sinal-Ruído (*Signal-to-Noise Ratio*)
T_D	-	Constante de Tempo Derivativo
T_I	-	Constante de Tempo Integral
t_r	-	Tempo de retenção da amostra
T_s	-	Tempo de Amostragem

U_{erro}	-	Diferença entre Tensão de Referência e Tensão de Saída
U_i	-	Valor analógico convertido
U_{in}	-	Tensão de Entrada
U_k	-	Tensão de Entrada na amostra de índice *k*
U_{k-1}	-	Tensão de Entrada na amostra de índice *k*-1
U_{k-2}	-	Tensão de Entrada na amostra de índice *k*-2
U_n	-	Tensão de ruído
U_{out}	-	Tensão de Saída
U_{ref}	-	Tensão de referência
U_s	-	Tensão de um sinal
Y_k	-	Tensão de Saída na amostra atual
Y_{k-1}	-	Tensão de Saída na amostra de índice *k*-1
Y_{k-2}	-	Tensão de Saída na amostra de índice *k*-2

Acrónimos e Abreviaturas

A/D	-	Conversão de Analógico para Digital
ADC	-	Módulo de conversão de Analógico para Digital
ASM	-	Linguagem Assembler (*Assembly*)
D/A	-	Conversão de Digital para Analógico
DAC	-	Módulo de conversão de Digital para Analógico
DSP	-	Processador Digital de Sinal (*Digital Signal Processor*)
MCU	-	Microcontrolador
PB	-	Passa-Baixo
PI	-	Proporcional Integral
PID	-	Proporcional Integral Derivativo
S&H	-	Amostrador Retentor (*Sample-and-Hold*)
SPCE	-	Semiplano Complexo Esquerdo
ZOH	-	Retentor de ordem zero (*Zero-Order-Hold*)

Capítulo I: Introdução

1.1 – Enquadramento e Motivação

O controlo ou monitorização de sistemas é uma área vasta da engenharia já há muito estudada e utilizada. Um sistema dinâmico pode ser, como exemplo, de um dos seguintes tipos:

- Sistema de fluidos;
- Sistema térmico;
- Sistema mecânico;
- Sistema elétrico;
- Conjuntos de dois ou mais dos sistemas anteriores;

Dependendo do processo, podem existir múltiplas entradas e saídas, e todas normalmente variáveis no tempo, podendo este ser em cadeia aberta ou fechada. Por exemplo, um filtro pode ser integrado num sistema em cadeia aberta e um controlador PID é, necessariamente, integrado num sistema em cadeia fechada [1], Figura 1.1.1.

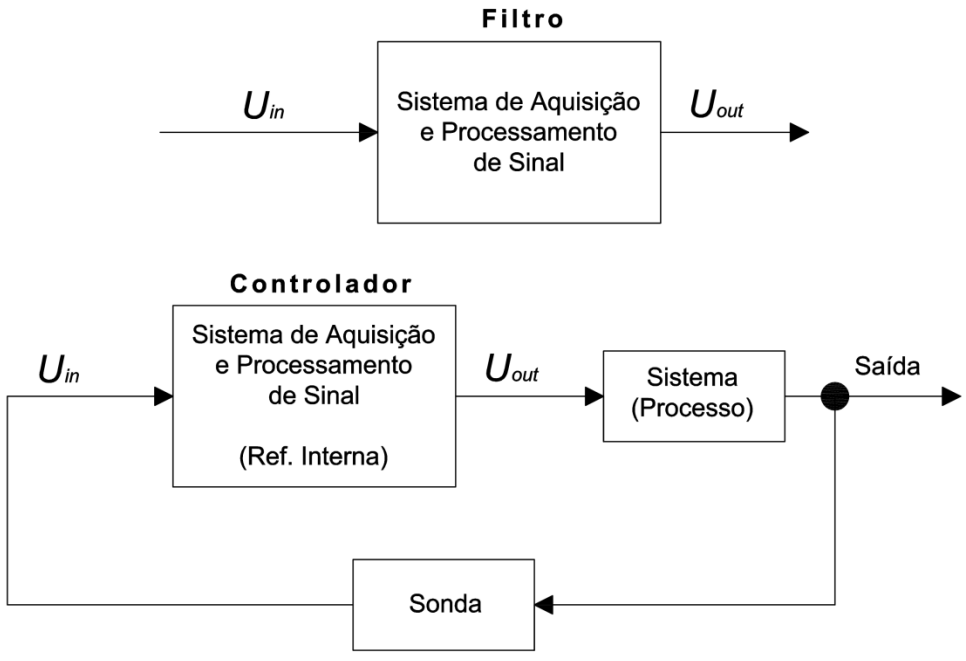


Figura 1.1.1 – Representação esquemática da aplicação prática de um filtro e controlador

O controlo ou filtragem de um sinal pode ser feito de forma analógica ou digital. Até cerca de 1970, a utilização de microcontroladores no controlo de sistemas estava comprometida devido ao elevado custo da eletrónica. As soluções existentes careciam de circuitos externos para

realizar as conversões A/D e D/A, circuitos auxiliares para a utilização das saídas e entradas e circuitos de lógica de temporização e interrupção, tornando a solução dispendiosa e apenas utilizada em processos industriais muito específicos [2]. A partir de 1970 o tamanho e custo da eletrónica diminuíram drasticamente e, com o lançamento dos novos processadores, tais como o Intel 8048 e TMS 1000, passou a ser possível utilizar este tipo de eletrónica em controlos básicos de forma económica, Figura 1.1.2. Estes chips possuem módulos integrados que simplificavam a implementação permitindo ao mesmo tempo um enorme leque de aplicações. A utilização de microcontroladores veio também revolucionar a área da aquisição de sinal.

Figura 1.1.2 – Microcontroladores Intel 8048 e TMS 1000

Sistemas de controlo, monitorização ou filtragem digital, são sistemas que podem ser parte integrante de uma cadeia de aquisição de sinal. Esta é composta, essencialmente, por:

- Transdutor;
- Condicionador de sinal;
- Placa de aquisição de sinal;

Considerando que o transdutor e o condicionador de sinal são periféricos à estrutura base da cadeia de aquisição e processamento digital de sinal, esta pode ser definida esquematicamente pela Figura 1.1.3.

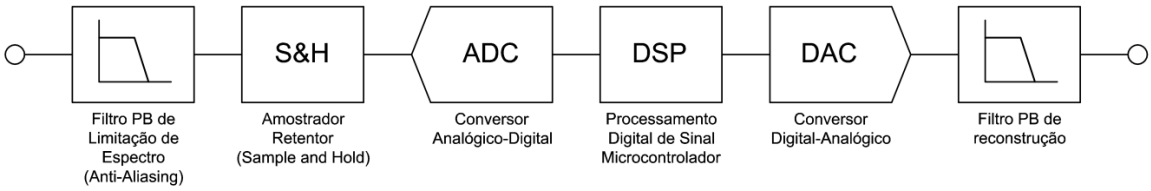


Figura 1.1.3 – Sistema de Aquisição e Processamento Digital de Sinal

É com base nesta estrutura que é desenvolvido este trabalho, no qual será criado um sistema de aquisição e processamento de sinal recorrendo à utilização de um microcontrolador que processa os sinais de forma discreta [3].

1.2 – Microcontroladores – MCU

Um microcontrolador (MCU) é um microprocessador com módulos periféricos específicos desenvolvido especialmente para aplicações de controlo com aplicação em equipamentos eletrónicos que utilizamos no dia-a-dia, tais como eletrodomésticos, eletrónica de consumo, centralinas de automóveis, etc.

Figura 1.2.1 – Centralina de carro Fiat

Estes dispositivos são, hoje em dia, económicos e possuem uma vasta gama de aplicações no controlo digital de sistemas. A maioria dos MCU's no mercado possuem circuitos integrados tais como conversores A/D e D/A, circuitos de interrupção, módulos de entradas e saídas digitais e analógicas, entre outros, permitindo a sua aplicação sem que seja necessário utilizar circuitos externos complexos.

A programação de um MCU era tradicionalmente feita em *Assembly (*ASM*)*. Hoje em dia é possível programar MCU's com linguagens de alto nível, tais como BASIC, PASCAL, C ou C++. Este tipo de linguagens oferece como vantagem ser de mais fácil de programação, teste, e compreensão do código desenvolvido, entre outras [4].

Como desvantagens da sua utilização, refira-se o espaço que o código em memória ocupa e o facto de, dependendo do compilador, a tradução para ASM não ser tão otimizada quanto a escrita direta do código nesta linguagem.

Neste trabalho é utilizado um MCU da *Microchip* (família *dsPIC*), desenvolvido especialmente para aplicações de controlo, que é programado em linguagem C. Este tipo de MCU tem características de DSP (*Digital Signal Processing*).

Um DSP é um tipo especial de MCU com uma capacidade de cálculo superior. No entanto, hoje em dia, a fronteira entre um MCU e um DSP é reduzida devido ao constante

avanço da tecnologia. As aplicações de um DSP são, principalmente, o processamento de dados representativos de sinais analógicos onde se obtêm atrasos muito inferiores aos que decorrem de um MCU normal. É possível encontrar DSP's em telemóveis, computadores, leitores de CD e MP3, sistemas de armazenamento de memória, rádios, televisões, etc.

1.3– Processamento Digital de Sinais

Um circuito típico de aquisição e processamento de sinais, apresentado na Figura 1.1.3, possui à entrada um circuito amostrador retentor e um conversor A/D, responsáveis pela discretização do sinal [5]. Para que o sinal seja processado no domínio discreto, é primeiro necessário que sejam retiradas amostras do sinal analógico original com um intervalo de tempo (T_s) entre elas - Amostragem.

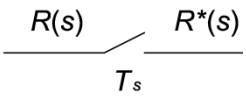


Figura 1.3.1 – Amostragem do Sinal

Na Figura 1.3.1 o sinal $R(s)$, analógico, é amostrado entre períodos de T_s (tempo de amostragem) passando a ter a denominação de $R^*(s)$, que representa $R(s)$ amostrado. O sinal é assim convertido para digital com uma frequência de amostragem, ω_s, que corresponde a:

$$\omega_s = \frac{2\pi}{T_s}$$

A amostragem de sinal carece de um circuito que permita reter amostras do sinal analógico até que seja feita a conversão para digital no conversor A/D. Após a conversão e processamento do sinal, é feita a conversão para analógico através de um conversor D/A.

Quando a frequência do sinal à entrada do sistema de aquisição (ω_b) é superior a metade da frequência de amostragem do MCU (ω_s) acontece um fenómeno de sobreposição de espectro, ou *aliasing* [5].

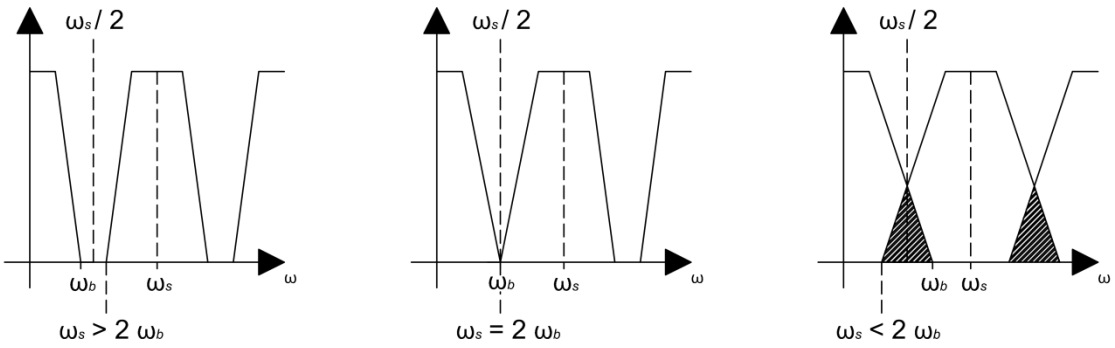


Figura 1.3.2 – Espectros resultantes para diferentes situações de frequência de amostragem

O Teorema de *Nyquist* defende que a frequência de amostragem necessita de ser, pelo menos, duas vezes superior à frequência do sinal à entrada para que o sinal não sofra uma modificação do seu espectro base (distorção do sinal). Na Figura 1.3.2 é possível observar a sobreposição dos espectros quando $2\omega_b > \omega_s$, em que $|X^*(j\omega)|$ representa o espectro de entrada. Para garantir que o sinal à entrada do controlador não possui uma frequência, tal que possa

causar distorção ao próprio, é utilizado um filtro passa-baixo analógico à entrada do sistema digital, denominado filtro limitador de espectro, ou *anti-aliasing*.

Para um tempo de amostragem de 10 µs, por exemplo, e consequentemente uma frequência de amostragem de 100 kHz, é então necessário garantir que o sinal original à entrada não possui uma frequência útil acima de 50 kHz. Para o dimensionamento do filtro limitador de espectro é necessário considerar a seguinte expressão, que relaciona o ganho do filtro com a frequência do sinal:

$$G(\omega)=\frac{G(0)}{\sqrt{1+\left(\frac{\omega}{\omega_c}\right)^{2n}}} \qquad 1.3.1$$

E onde n representa a ordem do filtro. A resposta em frequência deste filtro está apresentada na Figura 1.3.3.

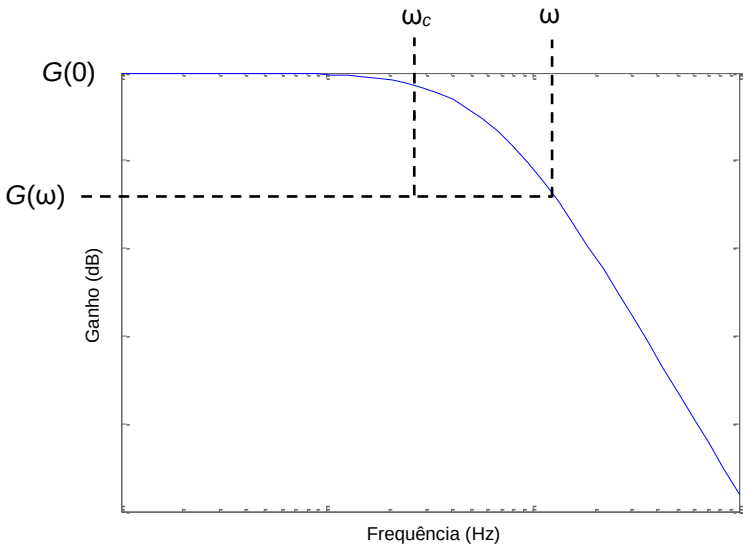


Figura 1.3.3 – Representação da Resposta em Frequência de um Filtro

É necessário calcular o valor de $G(\omega)$ como a atenuação do filtro à frequência de *Nyquist*. Este valor deve ser menor ou igual que o valor teórico da relação sinal-ruído, ou *Signal-to-Noise Ratio* (*SNR*). A base do cálculo do valor de *SNR* é o cálculo do erro de quantificação, e, que depende da largura do passo, q, denominada normalmente por resolução, existente num conversor A/D que, por sua vez, é função da tensão de referência, U_{ref}, do conversor e do número de bits, n.

$$q=\frac{U_{ref}}{2^n} \qquad 1.3.2$$

A tensão analógica à entrada do conversor, U_i, está compreendida num intervalo que depende da resolução e do valor numérico, N.

$$\left(N-\frac{1}{2}\right)q \leq U_i \leq \left(N+\frac{1}{2}\right)q \qquad 1.3.3$$

O erro relativo à conversão A/D é dado por:

$$e = Nq - U_i \qquad 1.3.4$$

E está compreendido no intervalo:

$$-\frac{q}{2} < e < \frac{q}{2} \qquad 1.3.5$$

Este erro pode ser interpretado como um ruído sobreposto à tensão de entrada. O valor deste ruído quadrático médio é determinado multiplicando o valor quadrático do erro pela sua probabilidade associada.

$$\bar{e}^2 = \int_{-\frac{q}{2}}^{\frac{q}{2}} e^2 f(e)\,de \;, \qquad \int_{-\frac{q}{2}}^{\frac{q}{2}} e^2 f(e)\,de = 1 \qquad 1.3.6$$

A função $f(e)$ representa a função densidade de probabilidade na ocorrência de um determinado valor de erro (e). Assumindo que a probabilidade da ocorrência do erro é igual independentemente do seu valor, $f(e)$ é constante com o valor de:

$$f(e) \int_{-\frac{q}{2}}^{\frac{q}{2}} de = 1 \;, \qquad f(e) = \frac{1}{q} \qquad 1.3.7$$

Dado que a variável aleatória de erro está distribuída no intervalo $]-q/2, q/2[$ de forma uniforme, em termos probabilísticos está-se perante uma distribuição uniforme, assim:

$$f(e) = \frac{1}{q} \;, \text{ com } -\frac{q}{2} < e < \frac{q}{2}$$
$$f(e) = 0 \;, \text{ caso contrário}$$

Por aplicação desta função à equação de erro quadrático médio obtém-se o valor do erro quadrático médio de quantificação.

$$\bar{e}^2 = \frac{1}{q}\int_{-\frac{q}{2}}^{\frac{q}{2}} e^2\,de \quad = \quad \frac{1}{q}\left[\frac{e^3}{3}\right]_{-\frac{q}{2}}^{\frac{q}{2}} \quad = \quad \frac{q^2}{12} \qquad 1.3.8$$

Como esta equação corresponde à definição do valor quadrático, isto é, quadrado do valor eficaz, o valor eficaz do ruído de quantificação é então:

$$e = \frac{q}{\sqrt{12}} \qquad 1.3.9$$

E da equação 1.3.2 vem:

$$e = \frac{U_{ref}}{2^n\sqrt{12}} \qquad 1.3.10$$

Por definição, a relação sinal-ruído é dada pelo quociente dos valores de potência referentes ao sinal, P_s, e ao ruído, P_n, vindo expressas em decibéis. Quando a medição das potências tem por base o mesmo valor de resistência, a relação é dada pelas tensões U_s e U_n:

$$SNR=10\log_{10}\frac{P_s}{P_n} = 10\log_{10}\frac{U_s^2/R}{U_n^2/R} = 20\log_{10}\frac{U_S}{U_n}$$

Considerando na entrada uma onda sinusoidal centrada em 0 e compreendida no intervalo [$-U_{ref}/2$; $U_{ref}/2$], o valor eficaz da tensão de sinal e ruído são:

$$U_s=\frac{U_{ref}}{2\sqrt{2}} \qquad U_n=e$$

E assim tem-se:

$$SNR(\text{dB})=20\log_{10}\frac{2^n\sqrt{12}}{2\sqrt{2}} = 20\log_{10}2^n+10\log_{10}\frac{3}{2}$$

$$SNR(\text{dB})=6{,}02n+1{,}76 \qquad 1.3.11$$

Sabendo que o conversor A/D utilizado, integrado no MCU, possui 10 bits para a conversão:

$$SNR=62\text{dB}$$

A Tabela 1.3.1 apresenta a relação entre *SNR* e o número de bits do conversor.

Tabela 1.3.1 – Relação entre *SNR* e o número de bits do conversor

n	8	10	12	14	18	20	24
***SNR* (dB)**	50	62	74	86	98	110	146

A partir da equação 1.3.1, é possível obter uma função da frequência de corte em relação à ordem do filtro a utilizar.

$$10^{-\frac{62}{20}}=\frac{10^{-\frac{0}{20}}}{\sqrt{1+\left(\frac{50.000}{f_c}\right)^{2n}}} \leftrightarrow f_c=\frac{50.000}{\sqrt[2n]{\left(\frac{1}{10^{-\frac{31}{10}}}\right)^2-1}}$$

Através desta função foi criada uma tabela que permite uma análise ao filtro a implementar:

Tabela 1.3.2 – Relação entre frequência de corte e ordem do filtro de *anti-aliasing*

Ordem	1	2	3	4	5	6	7	8
f_c (Hz)	40	1.409	4.631	8.394	11.994	15.216	18.035	20.487

Ainda que a partir do Teorema de *Nyquist* se obtenha o valor máximo da frequência de corte (ω_c) possível de definir num filtro digital, não é garantido que a resposta desse mesmo filtro a essa frequência seja idêntica à de um filtro analógico. Para isso, existem intervalos definidos por métodos empíricos que garantem a igualdade das respostas dos filtros digital e analógico. Assim, estabelecem-se os seguintes intervalos [3]:

$10\omega_c < \omega_s < 20\omega_c$, A resposta do sistema digital deve de ser comparada com a resposta do sistema analógico, introduzindo para o efeito um atraso puro médio $T_s/2$ no sistema contínuo.

$\omega_s > 20\omega_c$, O comportamento dos dois sistemas é praticamente idêntico.

Note-se que estes intervalos estão calculados de forma empírica para os filtros digitais. Na utilização de um MCU como controlador PI, por exemplo, não existindo uma frequência de corte ω_c, é necessário considerar a velocidade de resposta do sistema, assim como a velocidade da resposta e o tipo do controlador, para que possam ser definidos de forma empírica estes intervalos.

1.4– Objetivos

Com este trabalho pretende-se estudar e desenvolver uma cadeia de aquisição e processamento de sinal recorrendo a um sistema de processamento digital de sinal (DSP) para implementação de dois filtros passa-baixo (PB) de 1.ª e 2.ª ordem e dois controladores, proporcional integral (PI) e proporcional integral derivativo (PID). Para isso é necessário realizar inicialmente um estudo dos sistemas no domínio contínuo, desenvolvendo as equações dos mesmos no domínio do tempo para, posteriormente, realizar a sua discretização para o domínio discreto por três métodos de integração conhecidos (progressiva, regressiva e trapezoidal), sendo que apenas um é utilizado na simulação e implementação prática.

É apresentado o estudo da integração e critérios de estabilidade dos sistemas pelos vários métodos de integração, e o estudo da melhor estrutura a implementar (canónica direta ou não canónica direta).

Para validação e comparação dos resultados é feita a simulação dos sistemas no *software Matlab – Simulink* e implementação prática num MCU de 16 bits.

Capítulo II: Estudo dos Sistemas

Neste capítulo, irá ser abordada a resposta temporal e resposta em frequência de alguns sistemas conhecidos, nomeadamente um Filtro PB de 1.ª Ordem, Filtro PB de 2.ª Ordem com característica *Butterworth*, um controlador PI e um controlador PID. O objetivo passa pela comparação das respostas obtidas nos domínios contínuo e discreto, com as funções de transferência em função de *s* e *z* respetivamente. Os domínios contínuo e discreto podem ambos ser representados no tempo, mas apresentam respostas diferentes dependendo principalmente do tempo de amostragem. Para o estudo das respostas é necessário em primeiro lugar apresentar as equações, funções de transferência e funções de sistema em questão.

2.1 – Equações dos Sistemas

2.1.1 – Filtro PB de 1º Ordem

Um filtro passa-baixo tem como objetivo a diminuição da amplitude de um sinal com aumento da frequência. A atenuação do filtro, isto é, a capacidade de diminuir a amplitude do sinal a partir do ponto de frequência de corte (f_c), depende da sua característica e ordem. Com o incremento da ordem do um filtro, também aumenta a sua complexidade na implementação.

Um filtro pode ser constituído por componentes ativos ou passivos [5]. Quando constituído por componentes ativos, o filtro poderá ter um ganho superior à unidade, Figura 2.1.1.

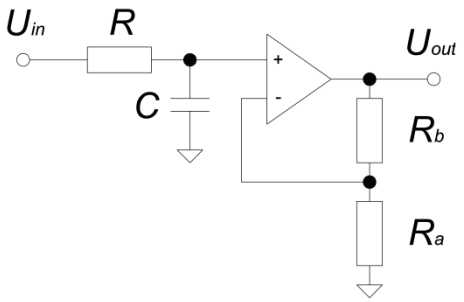


Figura 2.1.1 – Circuito equivalente de um Filtro Passa-Baixo de 1.ª Ordem com componentes ativos

Sendo G_0 o ganho estático do filtro, que depende do valor de R_a e R_b, a função de transferência deste circuito é dada por:

$$\frac{U_{out}}{U_{in}} = G_0 \frac{\frac{1}{RC}}{s+\frac{1}{RC}} \quad 2.1.1$$

Considerando:

$$\omega_c = \frac{1}{RC} \quad 2.1.2$$

A função de transferência do filtro PB de 1.ª ordem é:

$$F(s) = G_0 \frac{\omega_c}{s+\omega_c} = G_0 \frac{1}{\frac{1}{\omega_c}s+1} \quad 2.1.3$$

Como $\omega_c = 2.\pi.f_c$, a função de transferência pode ser escrita como:

$$F(s) = G_0 \frac{1}{\frac{1}{2\pi f_c}s+1} \quad 2.1.4$$

Passando para o domínio do tempo, pela tabela de transformada inversa de *Laplace* e considerando o ganho G_0 unitário tem-se:

$$f(t) = \omega_c e^{-\omega_c t} \quad 2.1.5$$

Quando constituído por componentes passivos, o circuito deste filtro toma a forma apresentada na Figura 2.1.2.

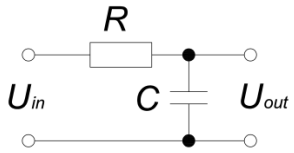


Figura 2.1.2 – Circuito equivalente de um Filtro Passa-Baixo de 1.ª Ordem com componentes passivos

A função de transferência do circuito com componentes passivos é idêntica à com componentes ativos mas com ganho unitário (equação 2.1.1).

2.1.2 – Filtro PB de 2º Ordem com característica *Butterworth*

Tal como apresentado para o filtro PB de 1.ª ordem, o circuito equivalente do filtro PB de 2.ª ordem com característica *Butterworth* [5], utilizando componentes ativos, está apresentado na Figura 2.1.3.

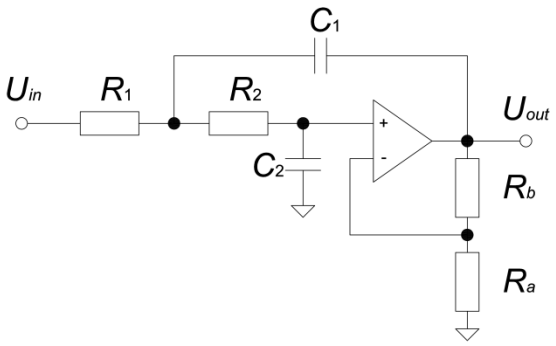


Figura 2.1.3 – Circuito Equivalente de um Filtro Passa-Baixo de 2.ª Ordem com componentes ativos

A função de transferência é obtida pela análise do circuito, dando origem a:

$$\frac{U_{out}}{U_{in}}=F_2(s)=G_0\frac{\frac{1}{R_1C_1R_2C_2}}{s^2+\left(\frac{R_1+R_2}{R_1R_2C_1}+\frac{1-G_0}{R_2C_2}\right)s+\frac{1}{R_1C_1R_2C_2}} \quad 2.1.6$$

$$F_2(s)=G_0\frac{\omega_c{}^2}{s^2+2\zeta\omega_c s+\omega_c{}^2}=G_0\frac{1}{\left(\frac{s}{\omega_c}\right)^2+2\zeta\left(\frac{s}{\omega_c}\right)+1} \quad 2.1.7$$

G_0 representa o ganho estático e ζ, denominado por coeficiente de amortecimento, é dado por:

$$\zeta=\frac{1}{2}\left(\frac{R_1+R_2}{R_1R_2C_1}+\frac{1-G_0}{R_2C_2}\right) \quad 2.1.8$$

Pela característica do filtro de *Butterworth* o valor de ζ é de $\frac{\sqrt{2}}{2}$.

O valor da frequência de corte é dado pela expressão:

$$\omega_c=\sqrt{\frac{1}{R_1C_1R_2C_2}}=2\pi f_c \quad 2.1.9$$

$$f_c=\frac{1}{2\pi}\sqrt{\frac{1}{R_1C_1R_2C_2}} \quad 2.1.10$$

Passando para o domínio do tempo, pela tabela de transformada inversa de *Laplace*, considerando o ganho estático unitário, tem-se:

$$f_2(t)=\omega_c t e^{-\omega_c t} \quad 2.1.11$$

2.1.3– Controlador PI

Um controlador PI analógico pode ser implementado com montagens série ou paralelo. A montagem em paralelo é apresentada na Figura 2.1.4:

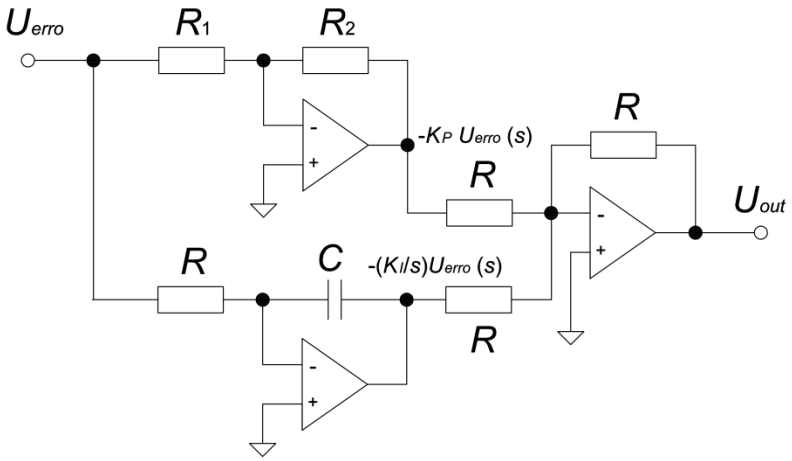


Figura 2.1.4 – Circuito Equivalente de um Controlador PI, topologia paralela

A função de transferência do controlador PI com montagem com topologia paralela é dada pela expressão:

$$\frac{U_{out}}{U_{erro}} = \frac{R_2}{R_1} + \frac{1}{sRC} \qquad 2.1.12$$

No domínio do tempo:

$$u_{out}(t) = \frac{R_2}{R_1} u_{erro}(t) + \frac{1}{RC}\int_0^t u_{erro}(t)dt \qquad 2.1.13$$

A equação típica de um controlador PI, no domínio do tempo, é dada por:

$$u_{out}(t) = K_P u_{erro}(t) + K_I \int_0^t u_{erro}(t)d(t) \qquad 2.1.14$$

Passando para o domínio de *Laplace*, a função de transferência do controlador PI é:

$$\frac{U_{out}}{U_{erro}} = C_{PI}(s) = K_p + \frac{K_I}{s} = \frac{sK_P + K_I}{s} \qquad 2.1.15$$

Assim, o ganho proporcional K_P, e o ganho integral, K_I, podem ser obtidos à custa das expressões:

$$K_P = \frac{R_2}{R_1} \qquad K_I = \frac{1}{\tau_I} = \frac{1}{RC} \qquad 2.1.16$$

O ganho integral, K_I, representa o número de vezes por minuto que se duplica a parte proporcional da ação de controlo, denominado por taxa de restabelecimento, onde τ_I é denominado por tempo integral.

A Figura 2.1.5 apresenta o circuito com a montagem com topologia série [6].

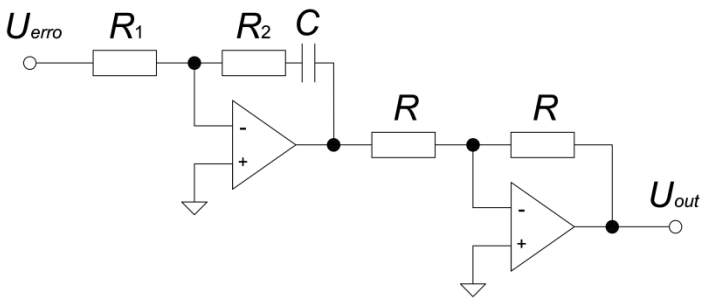


Figura 2.1.5 – Circuito Equivalente de um Controlador PI, topologia serie

A partir do circuito do controlador PI com topologia série pode escrever-se a função de transferência:

$$\frac{U_{out}}{U_{erro}} = \frac{R_2}{R_1} + \frac{1}{sR_1C} \qquad 2.1.17$$

Passando para o domínio do tempo:

$$u_{out}(t) = \frac{R_2}{R_1} U_{erro} + \frac{1}{R_1 C} \int_0^t u_{erro}(t)dt$$

Em que se tem, para ajuste dos ganhos:

$$K_P = \frac{R_2}{R_1} \qquad K_I = \frac{1}{R_1 C} \qquad 2.1.18$$

No circuito do controlador PI com montagem com topologia série, observando a equação 2.1.18 e comparando com a equação 2.1.16, é importante ter a noção que a alteração do ganho integral, quando por alteração da resistência R_1, afeta também o ganho proporcional, enquanto na montagem por topologia em paralelo é possível alterar apenas a resistência R para que o mesmo ganho seja alterado. Há assim uma relação entre a complexidade dos circuitos e a facilidade na alteração dos ganhos.

2.1.4– Controlador PID

No controlador PID também é possível realizar a montagem do circuito por várias topologias, entre elas as topologias paralela, série e com um único estágio de amplificação. A Figura 2.1.6 apresenta o circuito do controlador PID com topologia em paralelo.

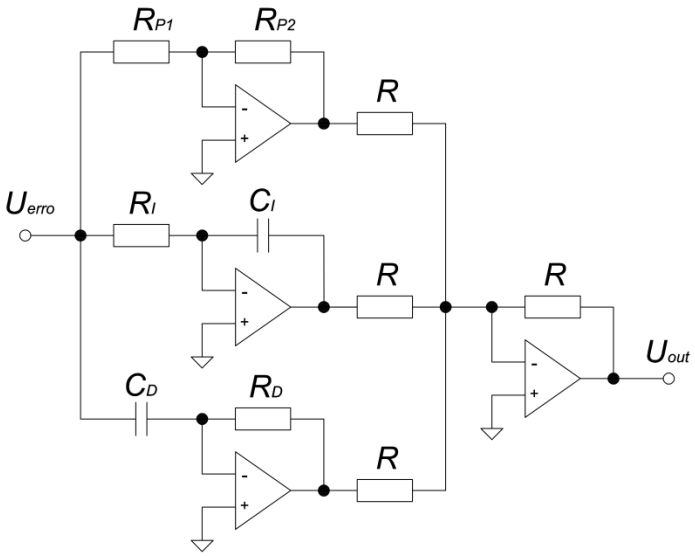


Figura 2.1.6 – Circuito Equivalente de um Controlador PID, topologia paralelo

A partir do circuito a função de transferência do controlador PID com uma topologia em paralelo é

$$\frac{U_{out}}{U_{erro}} = \frac{R_{P2}}{R_{P1}} + \frac{1}{sC_I R_I} + sR_D C_D \qquad 2.1.19$$

Passando para o domínio do tempo:

$$u_{out}(t) = \frac{R_{P2}}{R_{P1}} u_{erro}(t) + \frac{1}{C_I R_I}\int_0^t u_{erro}(t)dt + R_D C_D \frac{du_{erro}(t)}{dt} \qquad 2.1.20$$

A equação temporal do controlador PID é obtida pela combinação linear das equações temporais dos controladores P, I e D [6]. Assim:

$$u_{out}(t) = K_P u_{erro}(t) + K_P \frac{1}{T_I}\int_0^t u_{erro}(t)dt + K_P T_D \frac{du_{erro}(t)}{dt} \qquad 2.1.21$$

Em que:

$$K_P = \frac{R_{P2}}{R_{P1}} \qquad K_I = \frac{1}{sC_I R_I} \qquad K_D = K_P T_D = R_D C_D \qquad 2.1.22$$

Os componentes R_{P1} e R_{P2} ajustam a banda proporcional, enquanto R_I e C_I ajustam o ganho integral e R_D e C_D o ganho derivativo. T_D é o tempo no qual a ação diferencial se adianta do efeito da ação proporcional.

Quanto ao circuito do controlador PID com topologia serie, a Figura 2.1.7 apresenta a montagem típica.

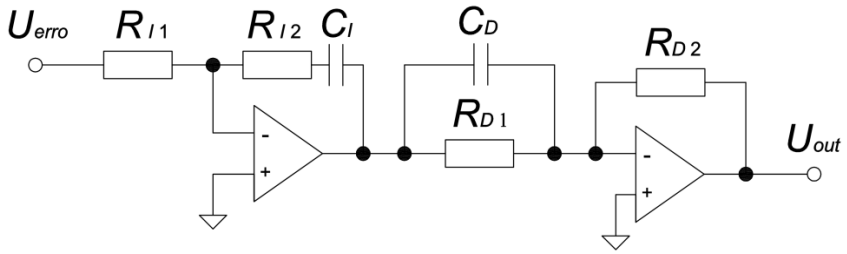


Figura 2.1.7 – Circuito Equivalente de um Controlador PID, topologia serie

A função de transferência do controlador PID pela montagem com topologia em série é dada por:

$$\frac{U_{out}}{U_{erro}}=\left(\frac{R_{I2}}{R_{I1}}+\frac{1}{sR_{I1}C_I}\right)\left(\frac{R_{D2}}{R_{D1}}+sR_{D2}C_D\right)=\frac{R_{I2}}{R_{I1}}\frac{R_{D2}}{R_{D1}}+\frac{R_{D2}C_D}{R_{I1}C_I}+s\frac{R_{I2}R_{D2}C_D}{R_{I1}}+\frac{R_{D2}}{sR_{I1}C_IR_{D1}} \qquad 2.1.23$$

No domínio do tempo:

$$u_{out}(t)=\left(\frac{R_{I2}}{R_{I1}}\frac{R_{D2}}{R_{D1}}+\frac{R_{D2}C_D}{R_{I1}C_I}\right)u_{erro}(t)+\frac{R_{D2}}{R_{I1}C_IR_{D1}}\int_0^t u_{erro}(t)dt+\frac{R_{I2}R_{D2}C_D}{R_{I1}}\frac{du_{erro}(t)}{dt} \qquad 2.1.24$$

Considerando a equação 2.1.21, o cálculo das constantes pode ser feito partindo de:

$$K_P=\frac{R_{I2}R_{D2}}{R_{I1}R_{D1}}+\frac{R_{D2}C_D}{R_{I1}C_I} \qquad K_I=\frac{R_{D2}}{R_{I1}C_IR_{D1}} \qquad K_D=\frac{R_{I2}R_{D2}C_D}{R_{I1}} \qquad 2.1.25$$

Quanto à topologia série, o circuito tem a forma apresentada na Figura 2.1.8.

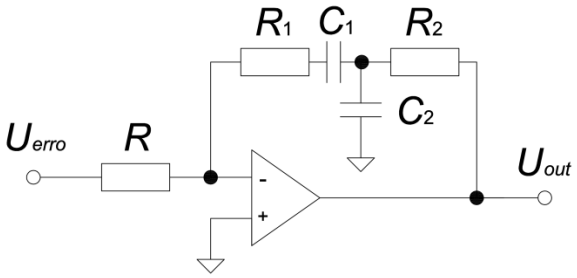


Figura 2.1.8 – Circuito Equivalente de um Controlador PID, topologia com um único estágio de amplificação

Neste tipo de montagem, a função de transferência é dada por:

$$\frac{U_{out}}{U_{erro}}=\frac{R_1C_1+R_2C_1+R_2C_2}{RC_1}+\frac{1}{sRC_1}+\frac{sR_1R_2C_2}{R} \qquad 2.1.26$$

No domínio do tempo:

$$u_{out}(t)=\frac{R_1C_1+R_2C_1+R_2C_2}{RC_1}u_{erro}(t)+\frac{1}{RC_1}\int_0^t u_{erro}(t)dt+\frac{R_1R_2C_2}{R}\frac{du_{erro}(t)}{dt}$$

Considerando novamente a equação 2.1.21 o cálculo das constantes pode ser feito partindo de:

$$K_P=\frac{R_1C_1+R_2C_1+R_2C_2}{RC_1} \qquad K_I=\frac{1}{RC_1} \qquad K_D=\frac{R_1R_2C_2}{R} \qquad 2.1.27$$

Ainda que seja um circuito mais simples, a topologia com um único estágio de amplificação apresenta dificuldades na alteração dos ganhos, consequência da alteração de qualquer componente alterar pelo menos dois ganhos, enquanto na montagem em paralelo a alteração de qualquer ganho só depende da alteração dos componentes associados ao mesmo.

A partir da equação 2.1.21, passando para o domínio de *Laplace* tem-se:

$$C_{PID}(s)=K_P+\frac{K_I}{s}+sK_D=\frac{sK_P+K_I+s^2K_D}{s} \quad 2.1.28$$

$$C_{PID}(s)=\frac{s^2K_D+sK_P+K_I}{s} \quad 2.1.29$$

Nos controladores PI e PID existe um fenómeno denominado de *windup*. Este ocorre nos controladores quando o valor à saída dos mesmos se encontra saturado no máximo, ou mínimo, e este continua a acumular continuamente um valor de erro.

Observando a equação 2.1.21, percebe-se que ao aumentar o tempo, para um erro u_{erro} constante, o valor do integral é continuamente crescente à medida que t aumenta. Se se controlar o valor da componente integral, isto é, o valor de $\frac{1}{T_I}K_P\int_0^t u_{erro}dt$, pode-se garantir que o valor de u_{out} não será superior ao admitido à saída do controlador. Outra forma, é controlar diretamente o valor de u_{out} utilizando uma comparação com o valor máximo admitido à saída do sistema, no entanto, após uma alteração do valor de u_{erro}, é necessário esperar pela diminuição ou aumento do valor da componente integral acumulado (quando existe).

2.2– Respostas dos Sistemas no Domínio Contínuo

Neste ponto do trabalho são obtidas as respostas temporais dos filtros de 1.ª e 2.ª ordem e controladores PI e PID. Aqui é possível observar o atraso nas respostas dos filtros e a resposta dos controladores face a uma entrada variável.

Estas respostas podem ser obtidas à custa de entradas do tipo degrau unitário (*step*) ou rampa. Para observar o atraso na resposta dos filtros é dada uma entrada em degrau, enquanto para os controladores é utilizado um degrau unitário para o controlador PI e uma rampa para o PID. Através do programa *Matlab* é possível efetuar a simulação, através de um ou mais ficheiros onde são introduzidos os dados e as equações (.m *files*).

2.2.1– Resposta do Filtro PB de 1º Ordem

Para obter a resposta do Filtro Passa-Baixo de 1.ª ordem a uma entrada do tipo degrau unitário é utilizado em *Matlab* o seguinte código, baseado na equação 2.1.3 e considerando um ganho G_0 unitário:

```
fc=4000;
wc=2*pi*fc;

Filtro = tf([1], [1/wc 1])
figure (1)
step(Filtro)
xlabel('Tempo (s)');
ylabel('Saída');
title('Resp. Temporal - Filtro PB de 1.ª Ordem')
```

A Figura 2.2.1 apresenta a imagem devolvida pelo código apresentado, para uma frequência de corte de 4 kHz.

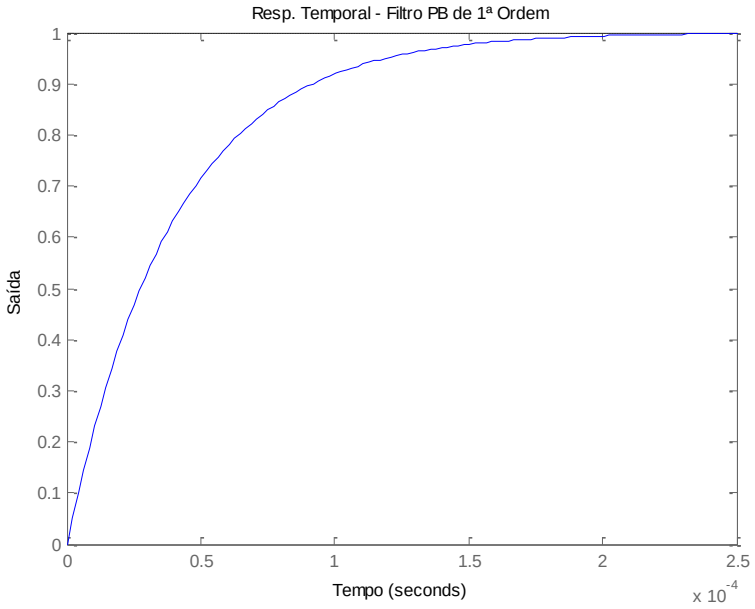


Figura 2.2.1 – Resposta Temporal de um Filtro PB de 1.ª ordem com f_c=4 kHz no domínio contínuo

É possível observar na Figura 2.2.1 que existe um tempo de estabelecimento, relativo ao atraso existente na resposta de um filtro. O tempo de estabelecimento, t_s, é o tempo ao fim do qual a resposta do sistema cresce desde 0 até se aproximar do regime estacionário com uma determinada margem de erro. Para este tempo, considera-se neste trabalho que a zona de estabelecimento da resposta se encontra na faixa onde a amplitude da resposta possui um erro de 5% face à amplitude final [6].

Para um sistema de primeira ordem, o cálculo do tempo de estabelecimento é dado pela expressão:

$$t_s(5\%) = 3\tau$$

Onde τ, constante de tempo do sistema de 1.ª ordem, no caso do filtro de primeira ordem, tem o valor de $\tau = \frac{1}{\omega_c}$. Então, para f_c=4 kHz:

$$t_s(5\%) = \frac{3}{2\pi f_c} \approx 119\ \mu s$$

A resposta em frequência deste filtro é obtida na Figura 2.2.2.

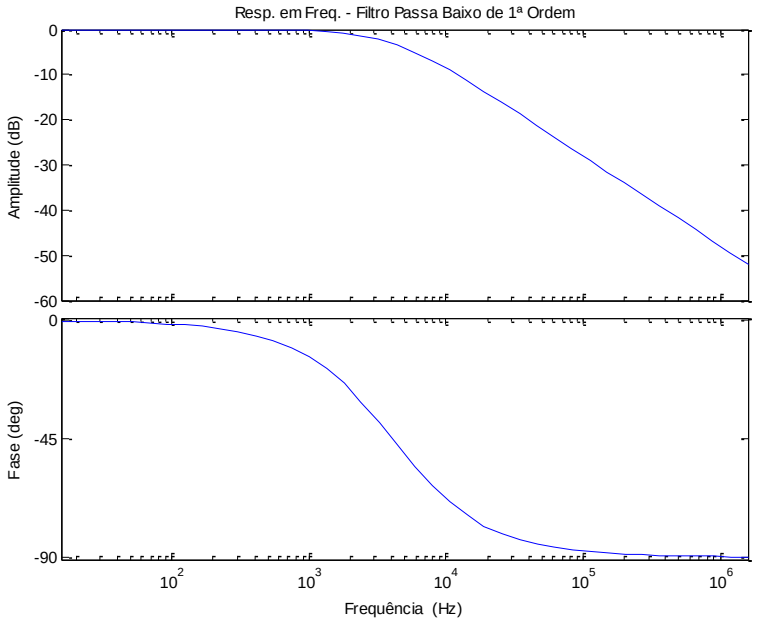


Figura 2.2.2 – Resposta em Frequência de um Filtro PB de 1.ª ordem com f_c=4 kHz no domínio contínuo

2.2.2 – Resposta do Filtro PB de 2º Ordem (*Butterworth*)

Para a resposta do Filtro Passa-Baixo de 2.ª ordem a uma entrada do tipo degrau unitária utiliza-se o seguinte código, baseado na equação 2.1.7 e considerando G_0 unitário e característica *Butterworth*:

```
fc=4000;
wc=2*pi*fc;

figure (2)
Filtro2 = tf([1], [1/wc^2 sqrt(2)/wc 1])
step (Filtro2)
xlabel('Tempo (s)');
ylabel('Saída');
title('Resp. Temporal - Filtro PB de 2.ª Ordem');
```

Que devolve, para uma frequência de corte de 4 kHz, a resposta apresentada na Figura 2.2.3.

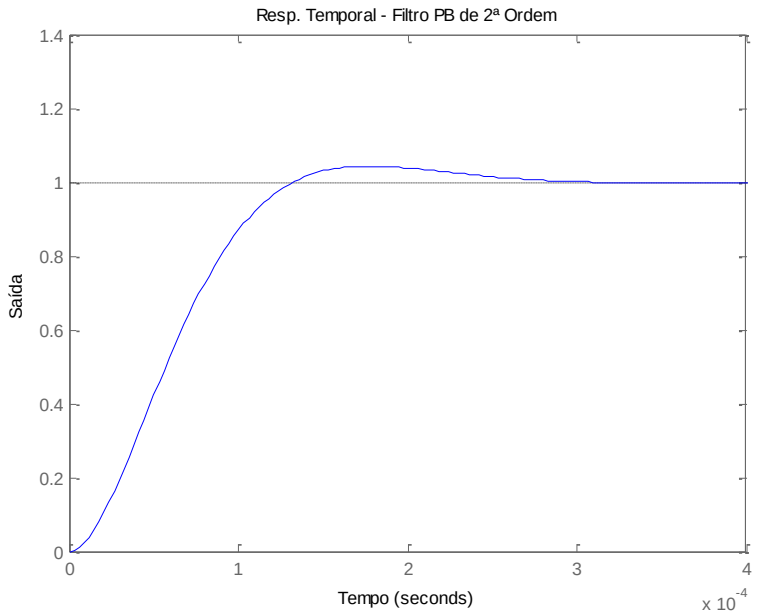


Figura 2.2.3 – Resposta Temporal de um Filtro PB de 2.ª ordem com f_c=4 kHz no domínio contínuo

Num sistema de 2.ª ordem, considera-se que o tempo de estabelecimento é dado pela expressão, obtida por métodos empíricos [6]:

$$t_s(5\%) = \frac{3}{\zeta\omega_c}\ \text{s}$$

Para f_c=4 kHz, e no caso do filtro com característica *Butterworth*, em que $\zeta=\frac{\sqrt{2}}{2}$:

$$t_s(5\%) = \frac{3}{2\zeta\pi f_c} = 167\ \mu\text{s}$$

Assim, comparando as respostas dos filtros de primeira e de segunda ordem, é possível afirmar que o tempo de estabelecimento de um filtro de 2.ª ordem é maior.

A resposta em frequência do filtro de 2.ª ordem com característica *Butterworth* e f_c=4 kHz está apresentada na Figura 2.2.4.

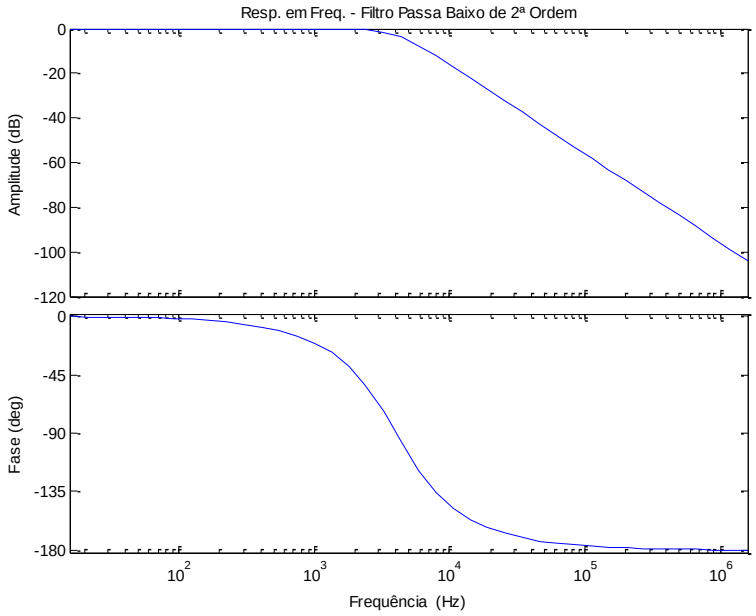


Figura 2.2.4 – Resposta em Frequência de um Filtro PB de 2.ª ordem com f_c=4 kHz no domínio contínuo

Como se pode observar, o filtro de 2.ª ordem possui uma maior atenuação a partir da frequência de corte.

Para ζ=1, ou seja, sem característica *Butterworth*, o tempo de estabelecimento é maior, como mostra a Figura 2.2.5.

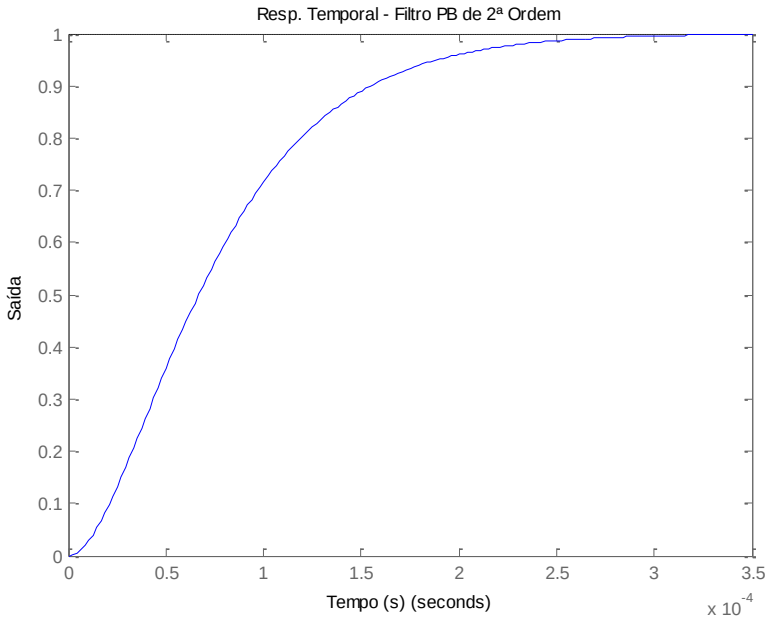


Figura 2.2.5 – Resposta Temporal de um Filtro PB de 2.ª ordem com f_c=4 kHz e coeficiente de amortecimento unitário

2.2.3 – Resposta do Controlador PI

Para a obtenção da resposta de um sistema a uma entrada do tipo degrau, utilizando o tipo de código em *Matlab* que foi utilizado nos pontos anteriores, é necessário que o sistema possua mais polos que zeros. Como na função de transferência de um controlador PI existe igual número de zeros e polos, não é possível utilizar o mesmo código. No entanto se se considerar a equação em ordem ao tempo do controlador PI (equação 2.1.14), e se se admitir que u_{erro} é a entrada do controlador, é possível colocar nessa entrada o valor 1 e observar o valor da saída do controlador ao longo do tempo, ou seja, a resposta a uma entrada do tipo degrau unitário.

Assim, para uma entrada u_{erro} =1 V:

$$u_{out}=K_P+K_I t \qquad 2.2.1$$

O código em *Matlab* utilizado é:

```
KP=1;
KI=1;

syms t;
Uout_PI = KP+KI*t       %Para e(t)=1
figure (3);
ezplot(Uout_PI, [0, 100]);
xlabel('Tempo (s)');
ylabel('Saída');
title('Resp. Temporal - Controlador PI');
axis([0 5 0 5]);
```

A resposta do controlador, para K_P=K_I=1, é apresentada na Figura 2.2.6.

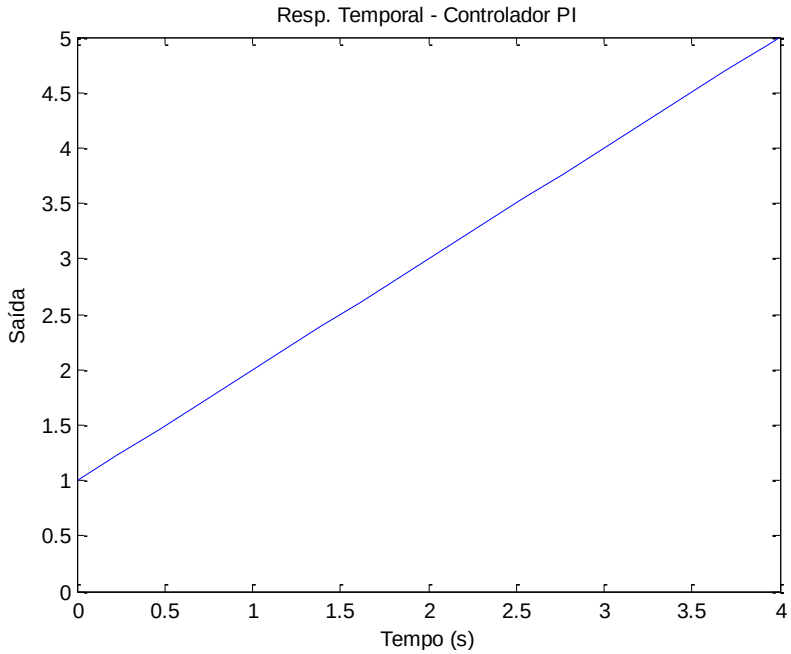


Figura 2.2.6 – Resposta Temporal do Controlador PI a uma entrada do tipo degrau unitário

Observando a Figura 2.2.6 percebe-se que a saída do controlador PI é uma reta com declive positivo. Este facto compreende-se dado que o PI dará como referência ao sistema a controlar um valor que aumenta constantemente na tentativa de baixar o erro que, neste caso, tem constantemente o valor de 1. É também possível observar nesta resposta a componente proporcional, que representa o valor inicial, 1, à saída do controlador.

2.2.4– Resposta do Controlador PID

No controlador PID, sabendo que a sua função de transferência possui um número de zeros superior ao número de polos, não é possível obter a sua resposta pelos comandos mais simples do *Matlab*. No entanto, conhecendo a equação do filtro em ordem ao tempo, é possível substituir o valor da entrada (erro) u_{erro} por 1 para obter a resposta a uma entrada do tipo degrau unitário. Como essa resposta seria idêntica à resposta obtida no ponto anterior para o controlador PI, consequência de no termo derivativo a derivada de u_{erro} =1 V ser 0, neste ponto será obtida a resposta do controlador PID a uma entrada do tipo rampa, substituindo u_{erro} =t.

$$u_{out}=K_P t+K_I t^2+K_D \qquad 2.2.2$$

Para obter esta resposta é utilizado o código:

```
KP=1;
KI=1;
KD=1;

syms t
Uout_PID = KP*t+ KI*t^2+ KD    %Para e(t)=t
figure (4);
ezplot(Uout_PID, [0, 100]);
xlabel('Tempo (s)');
ylabel('Saída');
title('Resp. Temporal - Controlador PID');
axis([0 3 0 5]);
```

Que devolve a imagem apresentada na Figura 2.2.7.

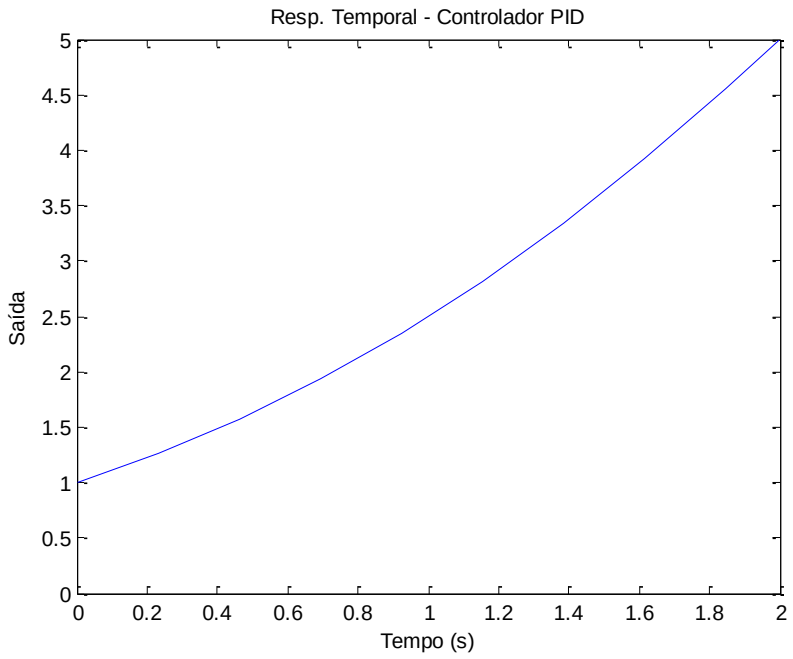


Figura 2.2.7 – Resposta Temporal do Controlador PID a uma entrada do tipo degrau unitário

Pela observação da Figura 2.2.7 é possível concluir que a resposta do controlador PID a um erro que aumenta constantemente com o aumento do tempo é uma saída que aumenta também constantemente de forma quadrática.

O Anexo 1 apresenta todo o código utilizado neste ponto do trabalho.

2.3– Métodos de Discretização

A transformação de um sinal contínuo em discreto pode ser feita segundo vários métodos de transformação, sendo sempre necessário, em qualquer sistema considerar a necessidade de, na conversão A/D, existir um processo de *Zero-Order-Hold* (ZOH) (ver Figura 1.1.3). A discretização de um sinal cria um outro composto por impulsos com um determinado tempo de amostra entre eles. Ao adicionar um ZOH, a amplitude dos impulsos é mantida constante durante o tempo de retenção da amostra.

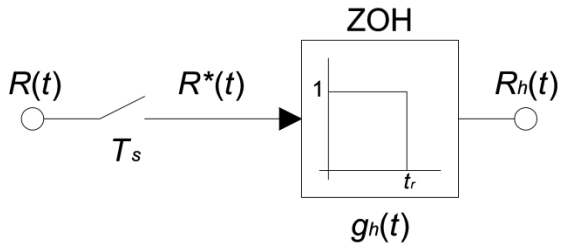


Figura 2.3.1– Retenção de amostra com ZOH

A retenção de amostra é dada pela função $g_h(t)$, onde t_r representa o tempo de retenção de amostra e $t_r \leq T_S$ [5].

Os modelos equivalentes discretos podem ser obtidos partindo dos modelos no domínio contínuo através de vários métodos. Entre eles tem-se:

- Integração numérica

  Consiste na transformação do domínio de *s* para o domínio de *z* através da substituição da variável *s* na função de transferência por uma expressão correspondente no domínio de *z*. Existem vários métodos de integração conhecidos, entre eles:

  - Integração retangular progressiva (*Forward Euler Method*)
  - Integração retangular regressiva (*Backward Euler Method*)
  - Integração trapezoidal ou bilinear (*Trapezoidal Euler Method*)
  - Transformação bilinear com deformação da escala de frequências (*warped method*)

- Mapeamento de polos e zeros

  Este método consiste na transformação do domínio contínuo para o domínio discreto através das tabelas de transformadas de *z*, partindo de $z=e^{sT_s}$.

- Modelo equivalente por retentores

  Neste método é assumido que o sistema se encontra num *loop* digital em que à entrada se têm amostras de forma contínua assim como à saída. Assim, recorrendo à utilização de séries e da tabela de transformadas de z obtém-se a função de transferência no domínio discreto.

- Conservação da resposta impulsiva (*impulse invariance transformation*)
- Invariância da resposta ao escalão (*step invariance transformation*)
- Transformada de *z* adaptada (*matched z transformation*)

O método de transformação mais utilizado é a integração numérica, mais especificamente, o método de integração trapezoidal. Para a discretização de um sistema, é utilizada uma integração por forma a criar um retângulo com largura T_s e área próxima à da área correspondente no sinal contínuo neste intervalo. Quanto mais próxima for esta área, menor é o erro associado ao método de integração.

Olhando para a Figura 2.3.2 é possível observar o princípio básico da discretização por integração numérica [3].

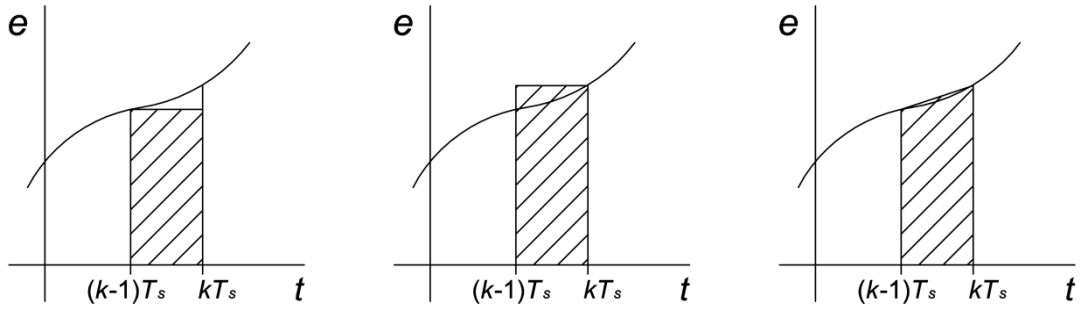


Figura 2.3.2 – Integração Progressiva, Regressiva e Trapezoidal respetivamente

Considerando um sistema com entrada $e(t)$ e saída $u(t)$ e, considerado que, no domínio de Laplace, se tem [5]:

$$\frac{U(s)}{E(s)}=\frac{1}{s} \quad \rightarrow \quad U(s)=\frac{1}{s}E(s)$$

No domínio do tempo:

$$u(t)=\int e(t)\mathrm{d}t$$

Observando a Figura 2.3.2, é possível representar $u(t)$ discretizado através da equação:

$$u(kT_s)=\int_0^{kT_s} e(t)\mathrm{d}t=\int_0^{(k\text{-}1)T_s} e(t)\mathrm{d}t+\int_{(k\text{-}1)T_s}^{kT_s} e(t)\mathrm{d}t \qquad 2.3.1$$

Outra forma de representar a equação é:

$$u(k)=T_s\sum_{j=0}^{k} \mathrm{e}(j)$$

Para a integração progressiva, considerando a Figura 2.3.2, tem-se:

$$u(k)=T_s\sum_{j=0}^{k\text{-}1} e(j)+T_s e(j\text{-}1)$$

Verifica-se que o histórico da integração é dado por:

$$u(k\text{-}1)=T_s\sum_{j=0}^{k\text{-}1}e(j)$$

Então, transformando para o domínio de *z*:

$$U(z)=U(z)z^{-1}+T_sE(z)z^{-1}$$

$$U(z)(1\text{-}z^{-1})=T_sE(z)z^{-1}$$

De onde se obtém a relação $E(z)/\ U(z)$:

$$\frac{E(z)}{U(z)}=\frac{1\text{-}z^{-1}}{T_sz^{-1}}=\frac{z\text{-}1}{T_s}$$

Do inverso da função de sistema equivalente à integração progressiva dada na função de transferência inicial obtém-se a transformação:

$$s\rightarrow\frac{z\text{-}1}{T_s} \qquad 2.3.2$$

Na integração regressiva, considera-se:

$$U(z)=U(z)z^{-1}+T_sE(z)$$

$$\frac{E(z)}{U(z)}=\frac{1\text{-}z^{-1}}{T_s}$$

Assim, na integração regressiva tem-se:

$$s\rightarrow\frac{z\text{-}1}{T_sz} \qquad 2.3.3$$

Para a integração trapezoidal, há uma divisão da área do trapézio em duas, por forma a obter-se o valor da mesma com largura T_s. Assim, na primeira metade utiliza-se o valor de $e(k\text{-}1)$ e na segunda metade o valor de $e(k)$ e, após a transformação para o domínio de *z*, obtém-se:

$$U(z)=U(z)z^{-1}+\frac{T_s}{2}\left(E(z)+E(z)z^{-1}\right)$$

Onde:

$$\frac{E(z)}{U(z)}=\frac{2}{T_s}\frac{1\text{-}z^{-1}}{1+z^{-1}}$$

E assim, para a integração trapezoidal a partir das equações no domínio de *Laplace*:

$$s \rightarrow \frac{2}{T_s}\frac{z\text{-}1}{z+1} \qquad 2.3.4$$

A Tabela 2.3.1 apresenta, de forma simplificada, as variáveis de substituição para a transformação direta das equações no domínio de *Laplace* para domínio discreto.

Tabela 2.3.1 – Variáveis de Substituição pelas diferentes Integrações

Integração	Domínio de *s*	Aproximação no domínio de *z*
Progressiva	s	$\frac{z\text{-}1}{T_s}$
Regressiva	s	$\frac{z\text{-}1}{T_s z}$
Trapezoidal	s	$\frac{2}{T_s} \times \frac{z\text{-}1}{z+1}$

É possível discretizar as funções de transferência dos Filtros e os Controladores pelos diferentes métodos de Integração. A obtenção da resposta do controlador PID a uma entrada do tipo degrau está comprometida pelas razões já apresentadas nos pontos 2.2.3 e 2.2.4 deste trabalho. Ainda assim, neste capítulo são apresentadas as equações deste controlador no domínio de *z*, pelos vários métodos de integração.

Uma vez no domínio discreto, é possível realizar uma transformação para o domínio do tempo, segundo determinadas regras que nos permitem obter as chamadas equações às diferenças $y(kT_s)$.

Existem três métodos para esta transformação:

- Séries de potência (divisão longa): este método permite obter o valor da resposta em função de kT_s, com k um número inteiro positivo.
- Transformadas inversas de *z* pela tabela: este método permite obter a equação às diferenças correspondente através da partição da equação no domínio de *z* e recorrendo ao uso de tabelas de transformadas.
- Inversão de integral: através do integral inverso é possível obter a transformada inversa de *z*, construindo a equação às diferenças.

À exceção do primeiro, os métodos apresentados são utilizados, neste trabalho, para a obtenção das equações no domínio do tempo e posterior comparação dos resultados no domínio *z*. Para a comparação das respostas, também com as equações às diferenças, é necessário dar uma perturbação ao sistema. Assim sendo, a comparação das respostas no domínio contínuo com as respostas na forma de equação às diferenças, é feita também considerando uma entrada no sistema do tipo degrau unitário.

2.3.1 – Integração no Filtro PB de 1.ª Ordem

2.3.1.1 – Integração Progressiva no Filtro PB de 1.ª Ordem

Com $s \rightarrow \frac{z-1}{T_s}$, partindo da equação 2.1.3 tem-se a seguinte transformação:

$$F(s)=G_0\frac{1}{\frac{1}{\omega_c}s+1} \rightarrow F(z)=G_0\frac{1}{\frac{1}{\omega_c}\frac{z-1}{T_s}+1}$$

$$F_P(z)=G_0\frac{\omega_c T_s}{z+\omega_c T_s-1} \qquad 2.3.5$$

Para obter a resposta temporal a uma entrada do tipo degrau, é necessária a multiplicação da equação obtida pela *FT* (Função de Transferência) de um degrau de valor 1. Como no domínio de *Laplace* a *FT* de um degrau unitário é $\frac{1}{s}$, no domínio de *z*, com $s \rightarrow \frac{z-1}{T_s}$, a *FS* (Função de Sistema) de um degrau unitário é então $\frac{1}{z-1}$. O termo T_s não se aplica no degrau unitário devido à sua continuidade no tempo. Assim sendo, a *FS* da resposta a uma entrada do tipo degrau é:

$$F_{P\,Step}(z)=G_0\frac{\omega_c T_s}{z+\omega_c T_s-1}\times\frac{1}{z-1}=G_0\frac{\omega_c T_s}{(z+\omega_c T_s-1)(z-1)} \qquad 2.3.6$$

Para f_c = 10 Hz, $\omega_c = 20\pi$ rad.s^{-1}, T_s = 0,01 s e G_0=1 V/V:

$$F_{P\,Step}(z)=\frac{0{,}6283}{(z-0{,}3717)(z-1)} \qquad 2.3.7$$

Neste caso, o melhor método para obter a transformada inversa de *z* é pelas tabelas de transformadas inversas recorrendo à partição da *FS*.

$$\frac{F_{P\,Step}(z)}{z}=\frac{A}{z}+\frac{B}{z-0{,}3717}+\frac{C}{z-1}$$

Em que

$$A=\left[\frac{0{,}6283}{(z-0{,}3717)(z-1)}\right]_{z=0}=1{,}6903$$

$$B=\left[\frac{0{,}6283}{z(z-1)}\right]_{z=0{,}3717}=-2{,}6903$$

$$C=\left[\frac{0{,}6283}{z(z-0{,}3717)}\right]_{z=1}=1$$

E, assim sendo, há agora condições de aplicar as transformadas inversas de *z*:

$$F_{P\,Step}(z)=\frac{1{,}6903z}{z}-\frac{2{,}6903z}{z-0{,}3717}+\frac{1}{z-1}$$

$$F_{P\,Step}(kT_s)=a-2{,}6903\times0{,}3717^{k}+1\times1^{k}$$

$$F_{P\,Step}(kT_s)=a+1-2{,}6903\times0{,}3717^{k} \qquad 2.3.8$$

com $a=\begin{bmatrix}1{,}6903, & k=0\\ 0, & k\neq0\end{bmatrix}$

O código utilizado para obter em *Matlab* as respostas no domínio contínuo, domínio discreto pela integração progressiva e equação às diferenças é:

```
%Continue Domain
    F = tf([wc], [1 wc])
%Forward Rule
    F_f=tf([wc*Ts], [1 wc*Ts-1], Ts)
% Equação às Diferenças
y_P=            1-2.6903.*0.3717.^(n/Ts)
y_P(1)=1.6903 + 1-2.6903.*0.3717.^(0/Ts)

figure (1)
hold on
plot(n,y_P,'ro')
axis([0 0.1 0 1])
step(F,F_f)
axis([0 0.1 0 1])
legend('Representação da Equação às Diferenças', 'Dimínio
Contínuo', ' Domínio Discreto')
xlabel('Tempo')
ylabel('Resposta')
title('Respostas nos Diferentes Domínios- Int. Prog.');
```

E devolve a imagem que se encontra na Figura 2.3.3.

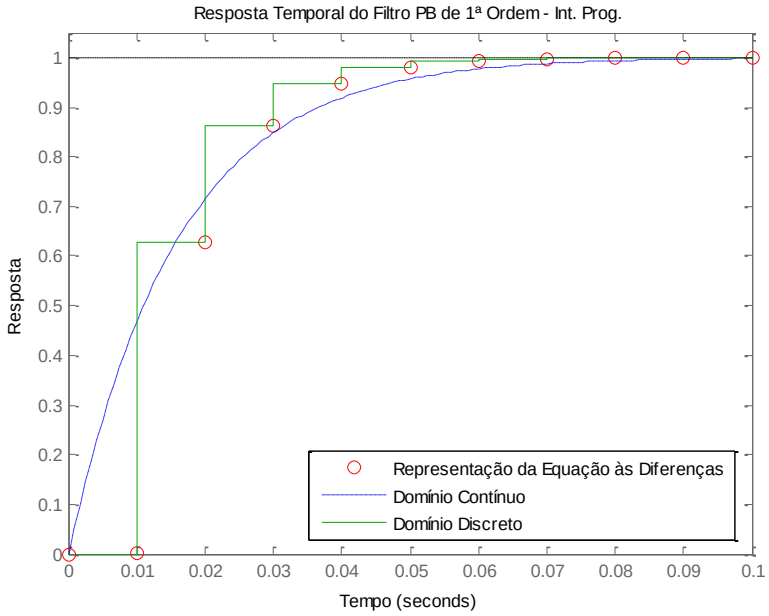


Figura 2.3.3 – Resposta do Filtro PB 1.ª Ordem pela equação às diferenças integração progressiva com T_s=0,01 s e f_c=10 Hz

Comparando a Figura 2.3.3 com a Figura 2.1.1, é possível observar que ao utilizar uma frequência de corte menor, utilizando 10 Hz em vez de 4 kHz, o atraso na resposta do filtro a uma entrada do tipo degrau é muito maior.

2.3.1.2 – Integração Regressiva no Filtro PB de 1.ª Ordem

Com $s \rightarrow \frac{z\text{-}1}{T_s z}$, partindo da equação 2.1.3, tem-se a seguinte transformação:

$$F_R(z)=G_0\frac{1}{\frac{1}{\omega_c}s+1} \rightarrow F_R(z)=G_0\frac{1}{\frac{1}{\omega_c}\frac{z\text{-}1}{T_s z}+1}$$

$$F_R(z)=G_0\frac{\omega_c T_s z}{(1+\omega_c T_s)z\text{-}1} \qquad 2.3.9$$

Substituindo as constantes (f_c = 10 Hz, ω_c = 20π rad.s^{-1}, T_s = 0,01 s e G_0=1 V/V) tem-se:

$$F_R(z)=\frac{0{,}3859}{z\text{-}0{,}6143} \qquad 2.3.10$$

E aplicando o degrau unitário:

$$F_{R\,Step}(z)=\frac{0{,}3859}{z\text{-}0{,}6143}\frac{z}{z\text{-}1}=\frac{0{,}3859z}{(z\text{-}0{,}6143)(z\text{-}1)} \qquad 2.3.11$$

Neste caso, o melhor método para obter a transformada inversa de z é pelo método do integral inverso:

$$F_{R\,Step}(kT_s)=0{,}3859\left(\left[\frac{z^k}{z\text{-}0{,}6143}\right]_{z=1}+\left[\frac{z^k}{z\text{-}1}\right]_{z=0{,}6143}\right)$$

$$F_{R\,Step}(kT_s)=0{,}3859\left(\frac{0{,}6143^k}{\text{-}0{,}3857}+\frac{1}{0{,}3857}\right)$$

$$F_{R\,Step}(kT_s)=1\text{-}0{,}6143^k \qquad 2.3.12$$

O código utilizado para a obtenção das respostas encontra-se no Anexo 2.

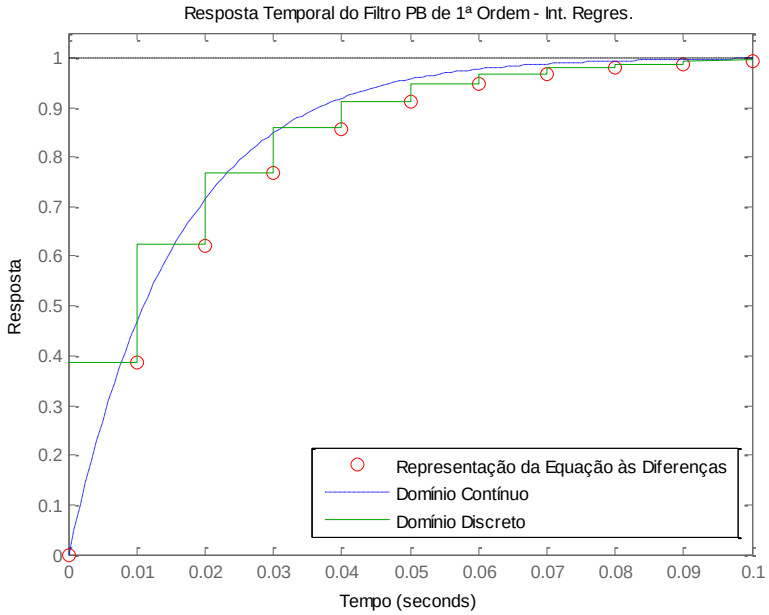


Figura 2.3.4 – Resposta do Filtro PB 1.ª Ordem pela equação às diferenças integração regressiva com T_s=0,01 s e f_c=10 Hz

2.3.1.3– Integração Trapezoidal no Filtro PB de 1.ª Ordem

Com $s \rightarrow \frac{2}{T_s}\frac{z-1}{z+1}$, partindo da equação 2.1.3, tem-se a seguinte transformação:

$$F(s)=G_0\frac{1}{\frac{1}{\omega_c}s+1} \rightarrow F_T(z)=G_0\frac{1}{\frac{1}{\omega_c}\frac{2}{T_s}\frac{z-1}{z+1}+1}$$

$$F_T(z)=G_0\frac{\omega_c z+\omega_c T_s}{(2+\omega_c T_s)z-2+\omega_c T_s} \qquad 2.3.13$$

Substituindo as constantes (f_c = 10 Hz, $\omega_c = 20\pi$ rad.s^{-1}, T_s = 0,01 s e G_0=1 V/V) tem-se:

$$F_T(z)=\frac{0,6283z+0,6283}{2,6280z-1,3720} \qquad 2.3.14$$

Tendo em conta que este método de integração é feito tomando dois pontos para o traçado de uma diagonal (forma de trapézio), não é possível aplicar a conversão se *s* para *z* na equação do degrau unitário. Assim sendo, utilizou-se a integração progressiva que se fixa num só ponto. Deste modo, tem-se:

$$F_{T\,Step}(z)=\frac{0,6283z+0,6283}{2,6280z-1,3720}\frac{1}{z-1}$$

$$F_{T\,Step}(z)=\frac{0,2391z+0,2391}{(z-0,5221)(z-1)} \qquad 2.3.15$$

Desta forma, o melhor método para obter a transformada inversa de *z* é, novamente, pelas tabelas de transformadas inversas recorrendo à partição da *FS*.

$$\frac{F_{T\,Step}(z)}{z}=\frac{A}{z}+\frac{B}{z\text{-}0{,}5221}+\frac{C}{(z\text{-}1)}$$

$$A=0{,}4580 \qquad B=\text{-}1{,}4590 \qquad C=1$$

$$F_{T\,Step}(z)=0{,}4580-\frac{1{,}4590z}{z\text{-}0{,}5221}+\frac{1.z}{(z\text{-}1)}$$

$$F_{T\,Step}(kT_s)=a\text{-}1{,}4590\times0{,}5221^{k}+1^{k}$$

$$F_{T\,Step}(kT_s)=a+1\text{-}1{,}4590\times0{,}5221^{k} \qquad 2.3.16$$

com $a=\begin{bmatrix}0{,}4580, & k=0\\ 0, & k\neq0\end{bmatrix}$

O código utilizado para a obtenção das respostas encontra-se no Anexo 2.

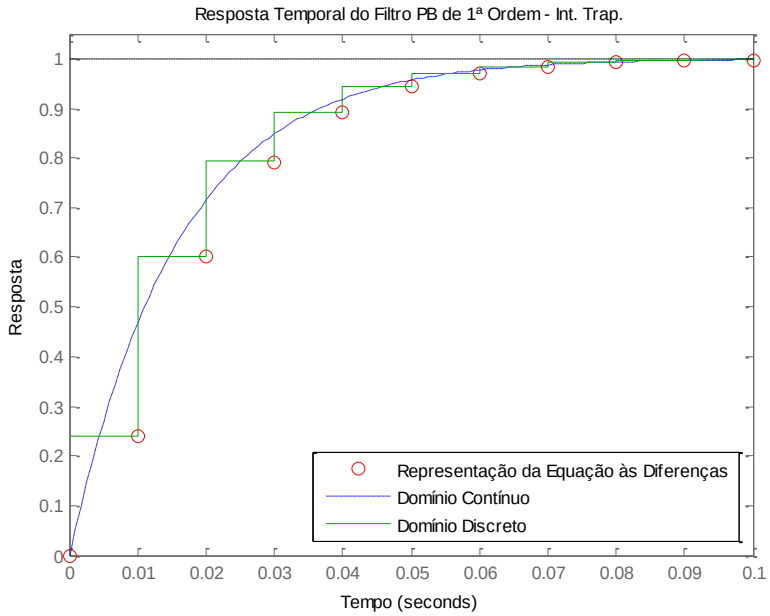


Figura 2.3.5 – Resposta do Filtro PB 1.ª Ordem pela equação às diferenças integração trapezoidal com T_s=0,01 s e f_c=10 Hz

Comparando com os outros métodos de integração, observa-se que a resposta no domínio discreto possui um valor mais próximo da resposta no domínio contínuo.

2.3.1.4 – Resposta do Filtro PB de 1.ª Ordem pelas Diferentes Integrações

A Figura 2.3.6 apresenta no mesmo gráfico as respostas do filtro PB de 1.ª ordem a uma entrada do tipo degrau, obtidas pelos diferentes métodos de integração. O código utilizado para a obtenção das respostas encontra-se no Anexo 2.

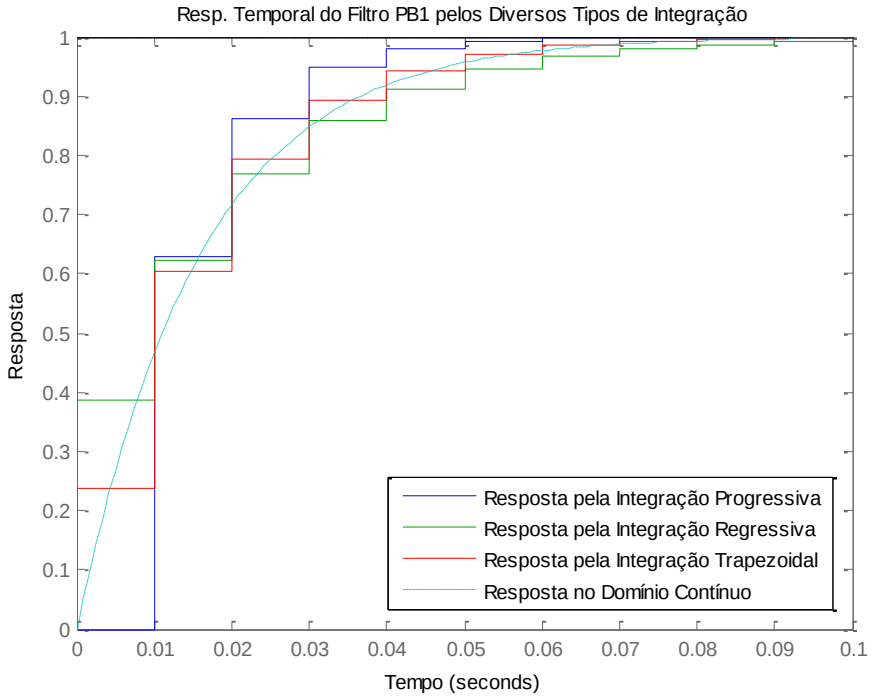


Figura 2.3.6 – Resposta temporal a uma entrada do tipo degrau do Filtro PB 1.ª Ordem pelas equações às diferenças várias integrações com T_S=0,01 s e f_C=10 Hz

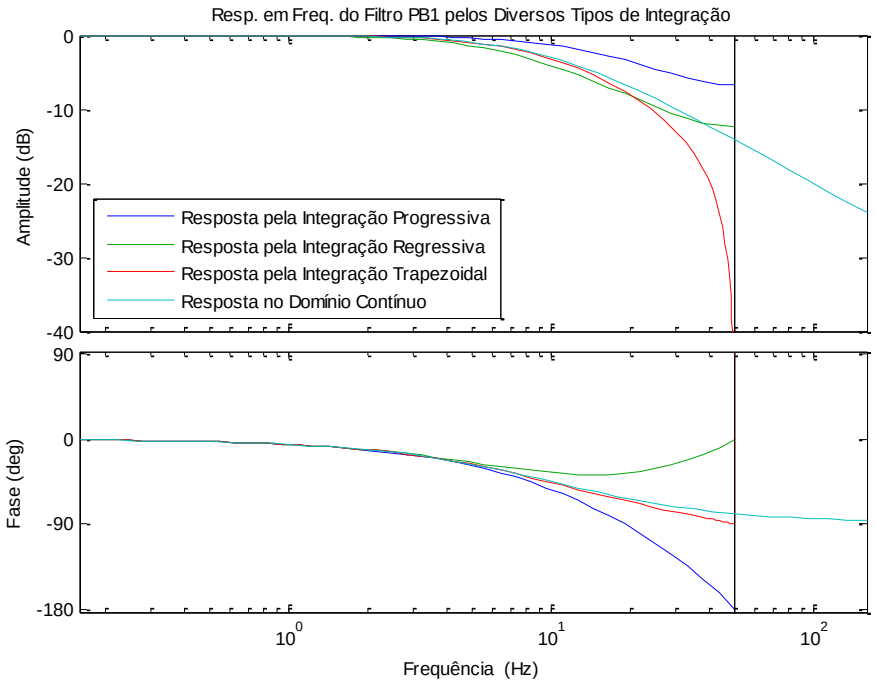


Figura 2.3.7 – Resposta em frequência do Filtro PB 1.ª Ordem pelas equações às diferenças várias integrações com T_S=0,01 s e f_C=10 Hz

A diferença entre as resposta no domínio contínuo e domínio discreto pela integração trapezoidal está relacionadas com uma deformação da escala de frequências (*frequency warping effect*) e da característica da fase [3], causada por este método de integração. Também nos controladores o sinal sofre uma pequena distorção, resultando numa diferença das respostas nos diferentes domínios, no entanto é nos filtros que este efeito toma maior importância.

Assumindo uma diferença entre a frequência do sinal contínuo (ω) e a frequência do sinal discreto (ω_d):

$$\omega \neq \omega_d$$

Considerando o método de integração trapezoidal, partindo de:

$$H(z)=[H(s)]_{s=\frac{2}{T_s}\frac{z-1}{z+1}}$$

Sabendo que $z=e^{j\omega_d T_s}$ e $s=j\omega$:

$$H\left(e^{j\omega_d T_s}\right)=[H(j\omega)]_{j\omega=\frac{2}{T_s}\frac{e^{j\omega_d T_s}-1}{e^{j\omega_d T_s}+1}}$$

Considerando:

$$j\omega=\frac{2}{T_s}\frac{e^{j\omega_d T_s}-1}{e^{j\omega_d T_s}+1}$$

Simplificando, obtém-se:

$$\omega=\frac{2}{T_s}tan\left(\frac{\omega_d T_s}{2}\right) \qquad 2.3.17$$

Que pode ser escrito também da forma:

$$\omega_d=\frac{2}{T_s}tan^{-1}\left(\frac{\omega T_s}{2}\right) \qquad 2.3.18$$

Com esta relação, é possível utilizar o método de integração trapezoidal definindo um valor de frequência ω_p onde a resposta do sistema discreto tem o mesmo valor que a resposta do sistema no domínio contínuo. Este método de integração é conhecido como transformação bilinear com deformação na escala de frequência (ou *warped*).

$$H(z)=[H(s)]_{s\approx\frac{\omega_p}{tan\left(\frac{\omega_p T_s}{2}\right)}\frac{z-1}{z+1}} \qquad 2.3.19$$

Ao observar a Figura 2.3.7 conclui-se que o método da integração trapezoidal permite obter um erro muito menor em relação à curva original no domínio contínuo. Este erro na resposta temporal pode ser calculado se forem criados, através da resposta a uma entrada do tipo degrau, vetores com os valores das respostas ao longo do tempo, e se se calcular em cada ponto o erro absoluto e relativo.

Uma vez tendo definido no programa as funções de transferência 'F' no domínio do tempo, 'F_f' discretizada pelo método de integração progressiva, 'F_b', integração regressiva e 'F_t' integração trapezoidal, os vetores são criado no *Matlab* através do código:

```
t=0:0.001:0.1;
t2=0:Ts:Ts*10;
Sr=step(F,t);
Fr=step(F_f,t2);
Br=step(F_b,t2);
Tr=step(F_t,t2);
```

Este código devolve quatro vetores. O primeiro vetor, correspondente à resposta no domínio contínuo, possui 101 valores de 0 a 0,1 com 0,001 de intervalo. Os restantes vetores, correspondentes às respostas no domínio discreto, possuem apenas 11 valores, e necessitam portanto de ser "expandidos", repetindo os valores por forma a criar um vetor com a mesma dimensão que o vetor da resposta no domínio contínuo.

Com ajuda do programa *Excel* são colocados os valores numa tabela e é calculado o valor do erro relativo e absoluto pelas expressões:

$$Erro\ Absoluto = Resp.\ no\ Dom.\ Contínuo - Resp.\ no\ Dom.\ Discreto$$

$$Erro\ Relativo = \frac{Resp.\ no\ Dom.\ Contínuo - Resp.\ no\ Dom.\ Discreto}{Resp.\ no\ Dom.\ Contínuo}$$

E assim obtêm-se a Figura 2.3.8:

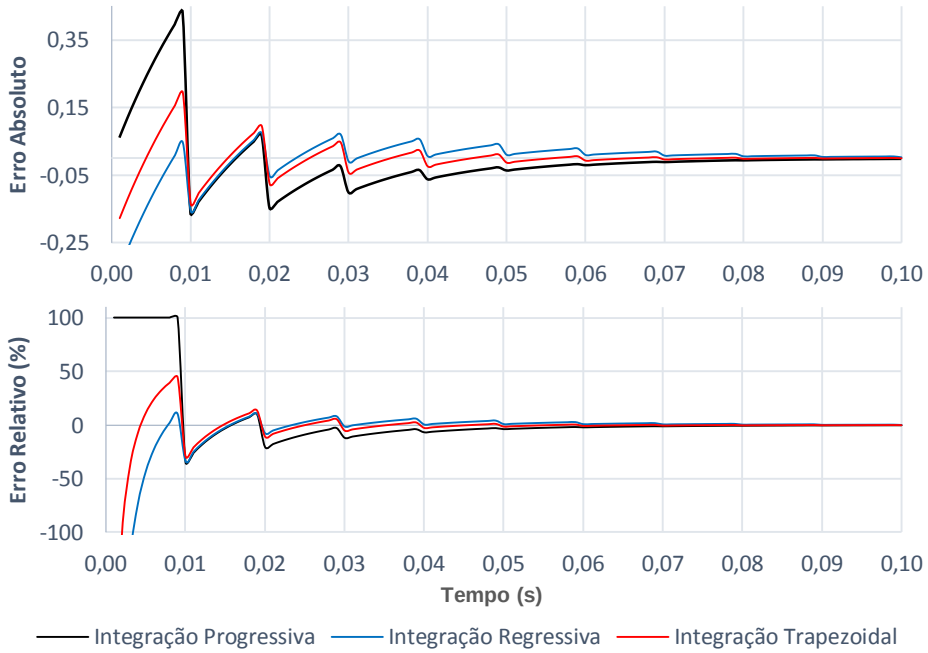


Figura 2.3.8 – Erro absoluto e relativo das respostas no domínio discreto do Filtro PB 1 em relação à resposta no domínio contínuo para uma entrada do tipo degrau unitário com T_s=0,01 s e f_c=10 Hz

É possível observar que a resposta que apresenta menor erro ao longo do tempo é a resposta do sistema no domínio discreto, discretizado pelo método de integração trapezoidal.

2.3.2 – Integração no Filtro PB de 2.ª Ordem

2.3.2.1– Integração Progressiva no Filtro PB de 2.ª Ordem

Com $s \rightarrow \frac{z-1}{T_s}$, a partir da equação 2.1.7, tem-se a seguinte transformação:

$$F_2(s)=G_0\frac{\omega_c^2}{s^2+2\zeta\omega_c s+\omega_c^2} \rightarrow F_{2P}(z)=G_0\frac{\omega_c}{\left(\frac{z-1}{T_s}\right)^2+2\zeta\omega_c\frac{z-1}{T_s}+\omega_c^2}$$

$$F_{2P}(z)=G_0\frac{\omega_c^2 T_s^{\,2}}{z^2+(2\zeta\omega_c T_s-2)z+1-2\zeta\omega_c T_s+\omega_c^2 T_s^{\,2}} \qquad 2.3.20$$

Substituindo, T_s=0,01 s, ω_c =62,8319 rad.s^{-1}, $\zeta=\frac{\sqrt{2}}{2}$ e G_0=1 V/V, tal como no filtro PB de 1.ª ordem, e aplicando a fórmula resolvente para obter os polos da função:

$$F_{2P}(z)=\frac{0{,}3948}{z^2-1{,}1110z+0{,}5062}=\frac{0{,}3948}{(z-(0{,}5555-0{,}4445j))(z-(0{,}5555+0{,}4445j))} \qquad 2.3.21$$

Tal como no filtro PB de 1.ª ordem, para obter a resposta a uma entrada do tipo degrau unitário, é necessária a multiplicação por $s \rightarrow \frac{z-1}{T_s}$. Desta forma, tem-se:

$$F_{2P\,Step}(z)=\frac{0{,}3948}{(z-(0{,}5555-0{,}4445j))(z-(0{,}5555+0{,}4445j))}\times\frac{1}{z-1}=$$
$$=\frac{0{,}3948}{(z-(0{,}5555-0{,}4445j))(z-(0{,}5555+0{,}4445j))(z-1)} \qquad 2.3.22$$

Neste caso, o melhor método para obter a transformada inversa de z é pelas tabelas de transformadas inversas recorrendo à partição da *FS*.

$$\frac{F_{2P\,Step}(z)}{z}=\frac{A}{z}+\frac{B}{z-1}+\frac{C}{z-(0{,}5555-0{,}4445j)}+\frac{D}{z-(0{,}5555+0{,}4445j)}$$

Em que

A=-0,7800 B=0,9991 C=-0,1095-0,9869j D=-0,1095+0,9869j

E assim sendo há agora condições de aplicar as transformadas inversas de z:

$$F_{2P\,Step}(z)=-\frac{0{,}7800z}{z}+\frac{0{,}9991z}{z-1}+\frac{(-0{,}1095-0{,}9869)z}{z-(0{,}5555-0{,}4445j)}-\frac{(-0{,}1095+0{,}9869)z}{z-(0{,}5555+0{,}4445j)}$$

$$F_{2P\,Step}(kT_s)=a+0{,}9991+(-0{,}1095-0{,}9869)\times(0{,}5555-0{,}4445j)^k$$
$$+(-0{,}1095+0{,}9869)\times(0{,}5555+0{,}4445j)^k \qquad 2.3.23$$

com $a=\begin{bmatrix}-0{,}7800,\ k=0\\ 0,\ k\neq 0\end{bmatrix}$

O código utilizado para a obtenção das respostas encontra-se no Anexo 2.

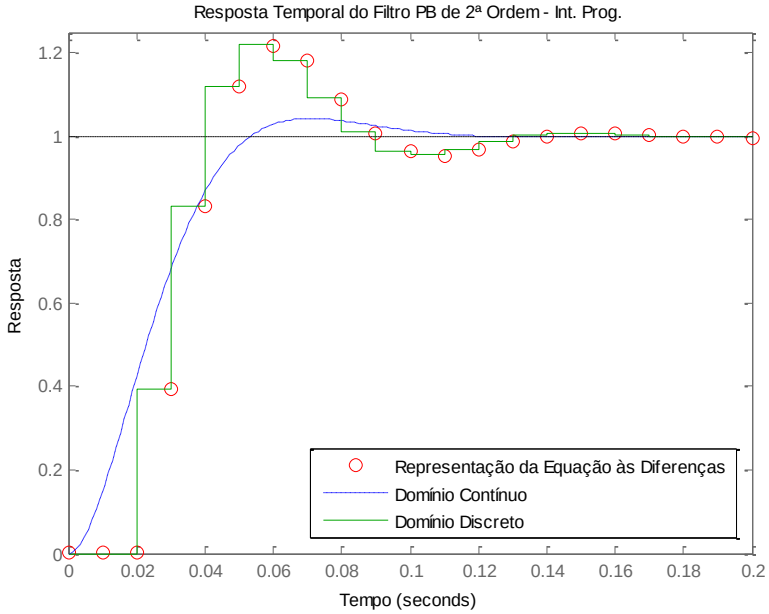


Figura 2.3.9 – Resposta do Filtro PB 2.ª Ordem pela equação às diferenças integração progressiva com T_s=0,01 s e f_c=10 Hz

2.3.2.2– Integração Regressiva no Filtro PB de 2.ª Ordem

Com $s \rightarrow \frac{z-1}{T_s z}$, a partir da equação 2.1.7, tem-se a seguinte transformação:

$$F_2(s)=G_0\frac{\omega_c^2}{s^2+2\zeta\omega_c s+\omega_c^2} \rightarrow F_{2R}(z)=G_0\frac{\omega_c^2}{\left(\frac{z-1}{T_s z}\right)^2+2\zeta\omega_c\frac{z-1}{T_s z}+\omega_c^2}$$

$$F_{2R}(z)=G_0\frac{\omega_c^2 T_s^2 z^2}{\left(1+2\zeta\omega_c T_s+\omega_c^2 T_s^2\right)z^2+(-2-2\zeta\omega_c T_s)z+1} \qquad 2.3.24$$

Substituindo, T_s=0,01 s, ω_c=62,8319 rad.s^{-1}, $\zeta=\frac{\sqrt{2}}{2}$ e G_0=1 V/V tal como no filtro PB de 1.ª ordem, e aplicando a fórmula resolvente para obter os polos da função:

$$F_{2R}(z)=\frac{0{,}3948z^2}{2{,}2830z^2-2{,}8890z+1}=\frac{0{,}1729z^2}{(z-(0{,}6327+0{,}1941j))(z-(0{,}6327-0{,}1941j))} \qquad 2.3.25$$

Tal como no filtro PB de 1.ª ordem, para obter a resposta a uma entrada do tipo degrau unitário, é necessária a multiplicação por $s \rightarrow \frac{z-1}{z}$. Desta forma, tem-se:

$$F_{2\,R\,Step}(z)=\frac{0{,}1489z^3}{(z-(0{,}6327+0{,}1941j))(z-(0{,}6327-0{,}1941j))(z-1)} \qquad 2.3.26$$

Mais uma vez, o melhor método para obter a transformada inversa de z é pelas tabelas de transformadas inversas recorrendo à partição da *FS*.

$$\frac{F_{2\,R\,Step}(z)}{z}=\frac{A}{z}+\frac{B}{z-1}+\frac{C}{z-(0{,}6327+0{,}1941j)}+\frac{D}{z-(0{,}6327-0{,}1941j)}$$

Em que

A=0 B=1,0019 C=-0,4145+0,2206j D=-0,4145-0,2206j

E, assim sendo, há agora condições de aplicar as transformadas inversas de z:

$$F_{2\,R\,Step}(z)=\frac{0.z}{z}+\frac{1{,}0019z}{z-1}+\frac{(-0{,}4145+0{,}2206j)z}{z-(0{,}6327+0{,}1941j)}-\frac{(-0{,}4145-0{,}2206j)z}{z-(0{,}6327-0{,}1941j)}$$

$$F_{2\,R\,Step}(kT_s)=1{,}0019+(-0{,}4145+0{,}2206j)\times(0{,}6327+0{,}1941j)^k \qquad 2.3.27$$
$$+(-0{,}4145-0{,}2206j)\times(0{,}6327-0{,}1941j)^k$$

O código utilizado para a obtenção das respostas encontra-se no Anexo 2.

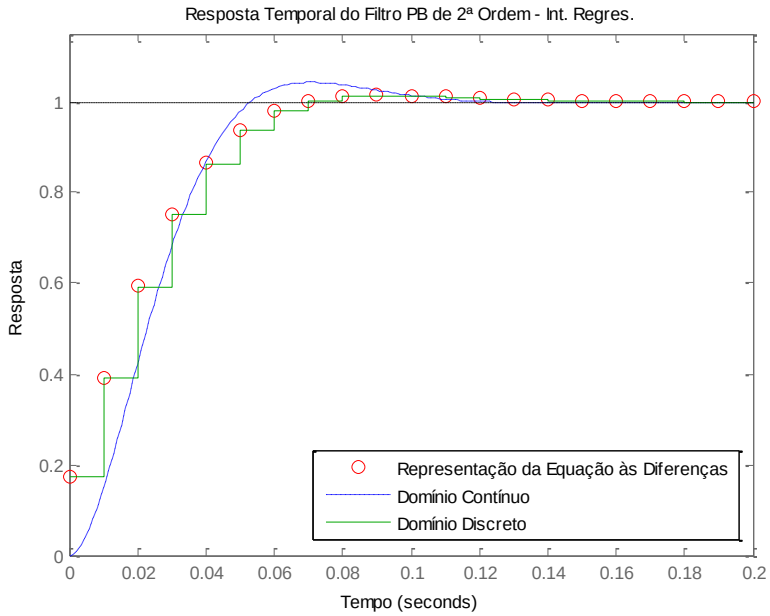


Figura 2.3.10 – Resposta do Filtro PB 2.ª Ordem pela equação às diferenças integração regressiva com T_S=0,01 s e f_C=10 Hz

2.3.2.3– Integração Trapezoidal no Filtro PB de 2.ª Ordem

Com $s \rightarrow \frac{2}{T_s}\frac{z-1}{z+1}$, a partir da equação 2.1.7, tem-se a seguinte transformação:

$$F_2(s)=G_0\frac{\omega_c^2}{s^2+2\zeta\omega_c s+\omega_c^2} \rightarrow F_{2\,T}(z)=G_0\frac{\omega_c^2}{\left(\frac{2}{T_s}\frac{z-1}{z+1}\right)^2+2\zeta\omega_c\frac{2}{T_s}\frac{z-1}{z+1}+\omega_c^2}$$

$$F_{2\,T}(z)=G_0\frac{\omega_c^2T_s^2z^2+2\omega_c^2T_s^2z+\omega_c^2T_s^2}{(4+4\zeta\omega_cT_s+\omega_c^2T_s^2)z^2+(-8+2\omega_c^2T_s^2)z+4-4\zeta\omega_cT_s+\omega_c^2T_s^2} \qquad 2.3.28$$

Substituindo, T_s=0,01 s, ω_c=62,8319 rad.s^{-1}, $\zeta=\frac{\sqrt{2}}{2}$ e G_0=1 V/V tal como no filtro PB de 1.ª ordem, e aplicando a fórmula resolvente para obter os polos da função:

$$F_{2\,T}(z)=\frac{0{,}0640z^2+0{,}1279z+0{,}0640}{(z-(0{,}5841+0{,}2881j))(z-(0{,}5841-0{,}2881j))} \qquad 2.3.29$$

Tal como no filtro PB de 1.ª ordem, para obter a resposta a uma entrada do tipo degrau unitário, é necessária a multiplicação por $s \rightarrow \frac{z-1}{T_s}$ (degrau com integração progressiva). Desta forma, tem-se:

$$F_{2\,T\,Step}(z)=\frac{0{,}0640z^2+0{,}1279z+0{,}0640}{(z-(0{,}5841+0{,}2881j))(z-(0{,}5841-0{,}2881j))(z-1)} \qquad 2.3.30$$

Mais uma vez o melhor método para obter a transformada inversa de z é pelas tabelas de transformadas inversas recorrendo à partição da *FS*.

$$\frac{F_{2\,T\,Step}(z)}{z}=\frac{A}{z}+\frac{B}{z-1}+\frac{C}{z-(0{,}5841+0{,}2881j)}+\frac{D}{z-(0{,}5841-0{,}2881j)}$$

Em que

A=-0,1509 B=0,9996 C=-0,4244+0,7634j D=-0,4244-0,7634j

E, assim sendo, há agora condições de aplicar as transformadas inversas de z:

$$F_{2\,T\,Step}(z)=-\frac{0{,}1509z}{z}+\frac{0{,}9996z}{z-1}+\frac{(-0{,}4244+0{,}7634j\,)z}{z-(0{,}5841+0{,}2881j)}-\frac{(-0{,}4244-0{,}7634j)z}{z-(0{,}5841-0{,}2881j)}$$

$$F_{2\,T\,Step}(kT_s)=a+0{,}9996+(-0{,}4244+0{,}7634j\,)(0{,}5841+0{,}2881j)^k +(-0{,}4244-0{,}7634j\,)(0{,}5841-0{,}2881j)^k \qquad 2.3.31$$

com $a=\begin{bmatrix}-0{,}1509, & k=0\\ 0, & k\neq 0\end{bmatrix}$

O código utilizado para a obtenção das respostas encontra-se no Anexo 2.

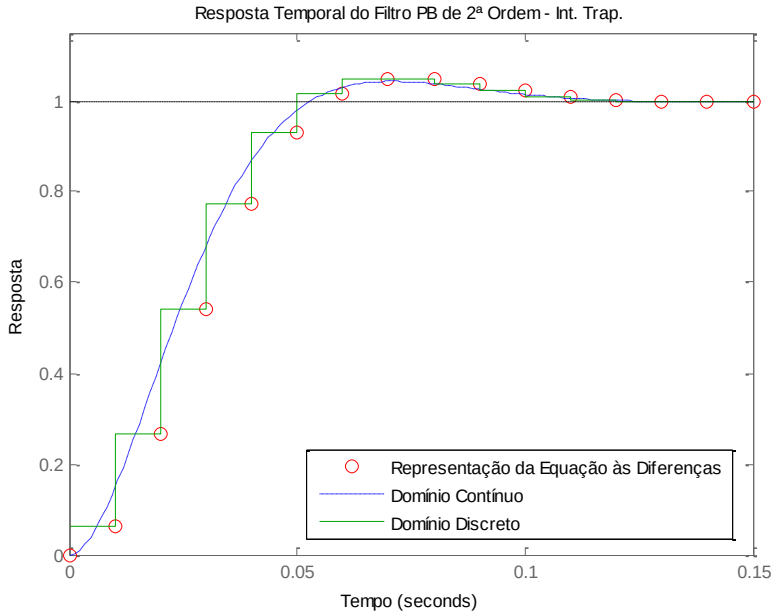


Figura 2.3.11 – Resposta do Filtro PB 2.ª Ordem pela equação às diferenças integração trapezoidal com T_s=0,01 s e f_c=10 Hz

2.3.2.4– Resposta do Filtro PB de 2.ª Ordem pelas Diferentes Integrações

Tal como apresentado para o filtro PB de 1.ª ordem, as respostas pelas diferentes integrações no filtro PB de 2.ª ordem e o erro associado a cada uma delas é apresentado neste ponto. O código utilizado para a obtenção das respostas encontra-se no Anexo 3.

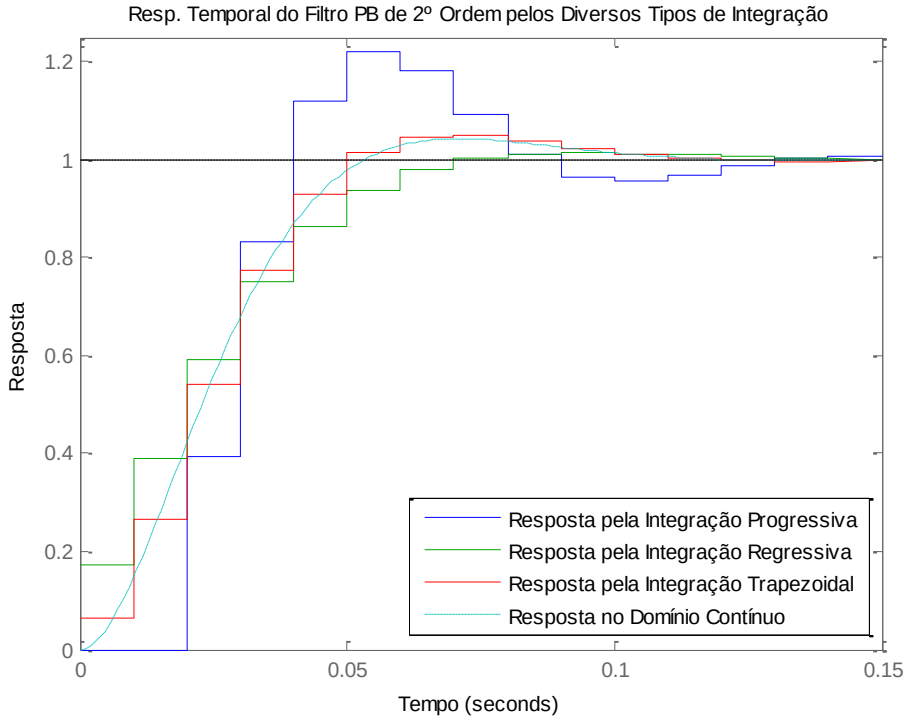


Figura 2.3.12 – Resposta temporal a uma entrada do tipo degrau do Filtro PB 2.ª Ordem pelas equações às diferenças várias integrações com T_s=0,01 s e f_c=10 Hz

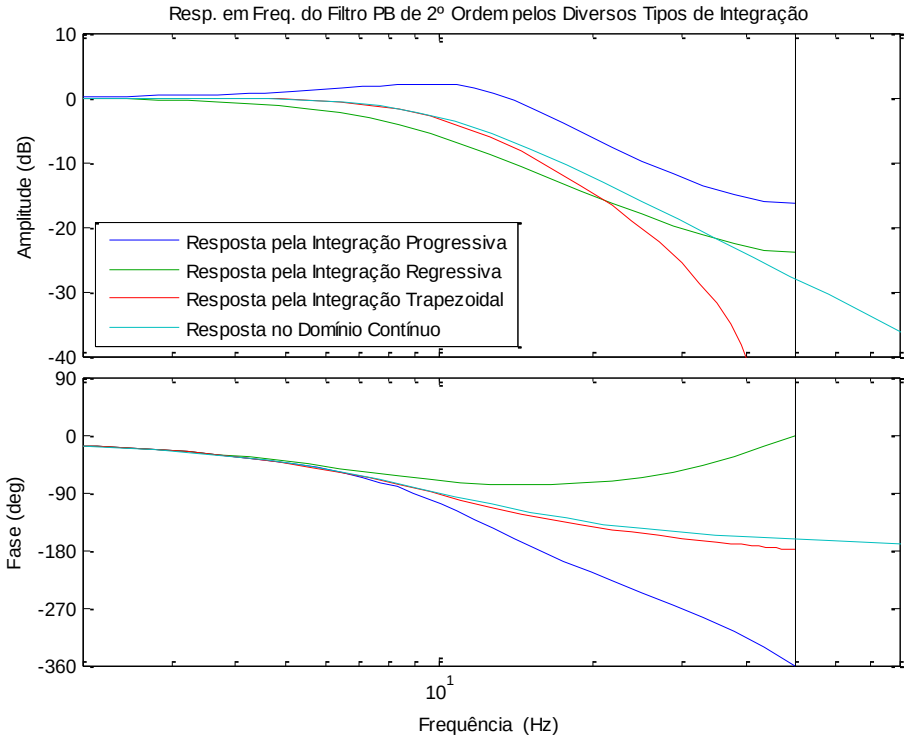


Figura 2.3.13 – Resposta em frequência do Filtro PB 2.ª Ordem pelas equações às diferenças várias integrações com T_s=0,01 s e f_c=10 Hz

Analisando a resposta em frequência conclui-se que a resposta obtida pelo sistema discretizado, pelo método de integração trapezoidal, consegue um valor mais próximo do valor obtido na resposta no domínio contínuo. Em todas as respostas pelos diferentes métodos de integração, com o aumento da frequência aumenta também a diferença entre as respostas no domínio discreto e contínuo.

A resposta em frequência sofre uma alteração em função da frequência de amostragem do sinal. A Figura 2.3.14 apresenta a resposta em frequência para 3 frequências de amostragem distintas, onde se observa que quanto maior for a frequência de amostragem, e quanto maior a diferença entre esta e a frequência de corte, melhor é a resposta do filtro.

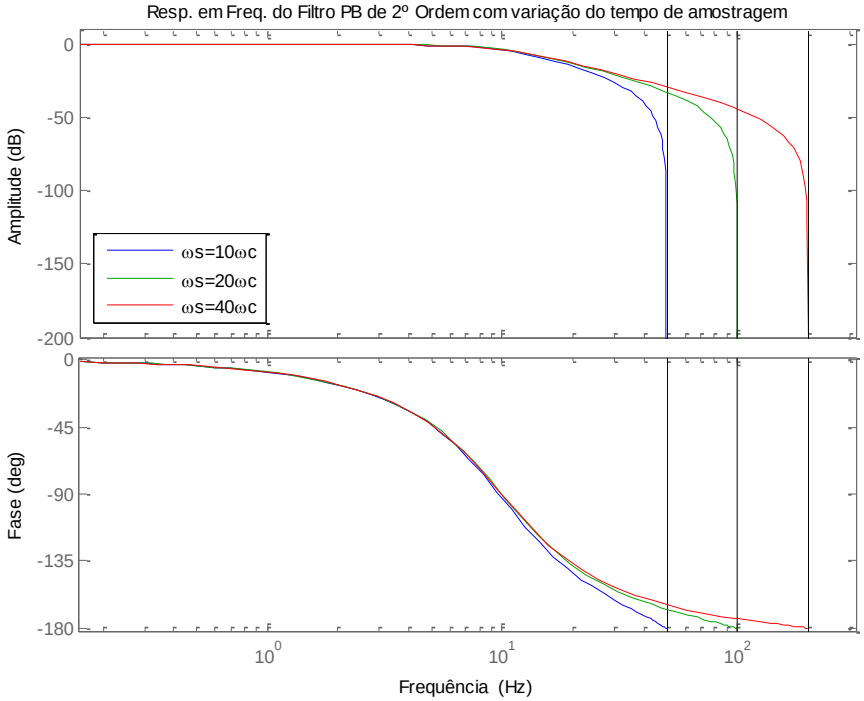


Figura 2.3.14 – Resposta do Filtro PB 2.ª Ordem a vários períodos de amostragem com T_s=0,01 s e f_c=10 Hz

Tal como feito para o filtro PB de 1.ª ordem, foi feita uma operação para calcular erro absoluto e relativo ao longo do tempo, para uma resposta a uma entrada do tipo degrau unitário, apresentada na Figura 2.3.15.

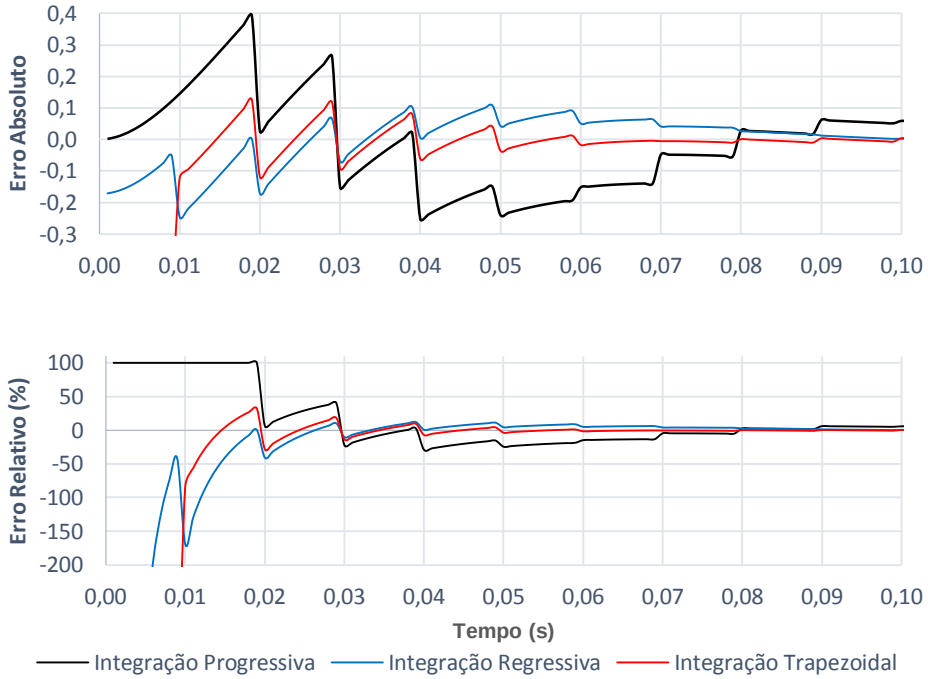


Figura 2.3.15 – Erro absoluto e relativo das respostas no domínio discreto do Filtro PB 2 em relação à resposta no domínio contínuo para uma entrada do tipo degrau unitário com T_s=0,01 s e f_c=10 Hz

Observando a figura conclui-se que o método que garante o menor erro é o método de integração trapezoidal.

2.3.3 – Integração no Controlador PI

Os controladores são utilizados em cadeia fechada, sendo a sua entrada a diferença entre o valor de referência e o valor à saída do sistema. Como não é estudado neste trabalho nenhum sistema em específico, não é apresentada neste ponto a resposta do controlador na sua função de controlo mas sim como sistema em cadeia aberta. Assim, as respostas observadas são a resposta a uma entrada de erro unitário constante, fazendo a saída do controlador aumentar infinitamente.

2.3.3.1 – Integração Progressiva no Controlador PI

A transformação do domínio contínuo para o discreto pela integração progressiva é feita da mesma maneira que para os filtros. Assim, partindo da equação 2.1.15:

$$C_{PI}(s)=\frac{sK_P+K_I}{s} \rightarrow C_{PI\,P}(z)=\frac{\frac{z-1}{T_s}K_P+K_I}{\frac{z-1}{T_s}}$$

$$C_{PI\,P}(z)=\frac{\frac{K_P}{T_s}z-\frac{K_P}{T_s}+K_I}{\frac{1}{T_s}z-\frac{1}{T_s}}=\frac{K_Pz-K_P+K_IT_s}{z-1} \qquad 2.3.32$$

Conhecendo as constantes, K_P=0,1 e K_I=1:

$$C_{PI\,P}(z)=\frac{0,1z-0,09}{z-1} \qquad 2.3.33$$

A resposta a uma entrada do tipo degrau é dada por:

$$C_{PI\,P\,Step}(z)=\frac{0,1z-0,09}{z-1}\frac{1}{z-1}=\frac{0,1z-0,09}{(z-1)^2} \qquad 2.3.34$$

Neste caso utiliza-se também a transformada inversa pelas tabelas de transformadas, com especial atenção à presença de raízes de ordem múltipla em que a transformação se processa de forma diferente, ficando a expansão parcial das frações:

$$\frac{C_{PI\,P}(z)}{z}=\frac{0,1z-0,09}{z(z-1)^2}=\frac{A}{z}+\frac{B}{z-1}+\frac{C}{(z-1)^2}$$

Em que os termos A, B e C são respetivamente:

$$A=0,09 \qquad B=\frac{d}{dz}\left[\frac{0,1z-0,09}{z}\right]_{z=1}=0,09 \qquad C=0,01$$

$$C_{PI\,P}(z)=\frac{0,09z}{z}+\frac{0,09}{z-1}+\frac{0,01z}{(z-1)^2}$$

$$C_{PI\,P}(kT_s)=a+0{,}09+0{,}01k \qquad 2.3.35$$

com $a=\begin{bmatrix}0{,}09, & k=0\\ 0, & k\neq 0\end{bmatrix}$

O código utilizado para a obtenção das respostas encontra-se no Anexo 2.

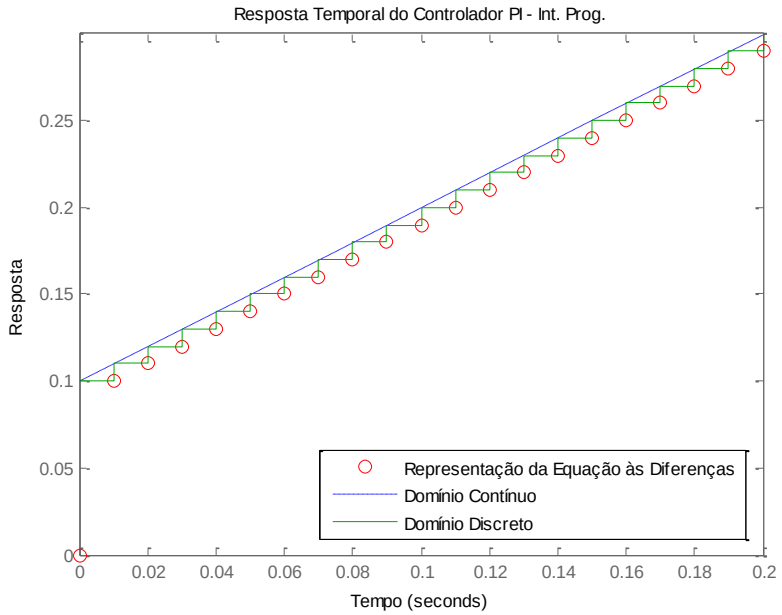


Figura 2.3.16 – Resposta do Controlador PI pela equação às diferenças integração progressiva com T_S=0,01 s, K_P=0,1 e K_I=1

2.3.3.2– Integração Regressiva no Controlador PI

Para a integração progressiva no controlador PI é feita a passagem, partindo da equação 2.1.15:

$$C_{PI}(s)=\frac{sK_P+K_I}{s} \rightarrow C_{PI\,B}(z)=\frac{\frac{z-1}{T_s z}K_P+K_I}{\frac{z-1}{T_s z}}$$

$$C_{PI\,B}(z)=\frac{\left(\frac{K_P}{T_s}+K_I\right)z-\frac{K_P}{T_s}}{\frac{1}{T_s}z-\frac{1}{T_s}}=\frac{(K_P+T_sK_I)z-K_P}{z-1} \qquad 2.3.36$$

Substituindo as constantes, K_P=0,1, K_I=1, e aplicando o degrau unitário, tem-se:

$$C_{PI\,B}(z)=\frac{0{,}11z-0{,}1}{z-1} \qquad 2.3.37$$

$$C_{PI\,B\,Step}(z)=\frac{0{,}11z-0{,}1}{z-1}\frac{z}{z-1}=\frac{0{,}11z^2-0{,}1z}{(z-1)^2} \qquad 2.3.38$$

Utilizando também o método das transformadas inversas de z pelas tabelas de transformadas obtém-se:

$$\frac{C_{PI\,B}(z)}{z}=\frac{0{,}11z^2-0{,}1z}{z(z-1)^2}$$

$$\frac{C_{PI\,B}(z)}{z}=\frac{A}{s}+\frac{B}{z-1}+\frac{C}{(z-1)^2}$$

$$A=0 \qquad B=0{,}11 \qquad C=0{,}01$$

$$C_{PI\,B}(z)=\frac{0z}{z}+\frac{0{,}11}{z-1}+\frac{0{,}01z}{(z-1)^2}$$

$$C_{PI\,B}(kT_s)=0{,}11+0{,}01k \qquad 2.3.39$$

O código utilizado para a obtenção das respostas encontra-se no Anexo 2.

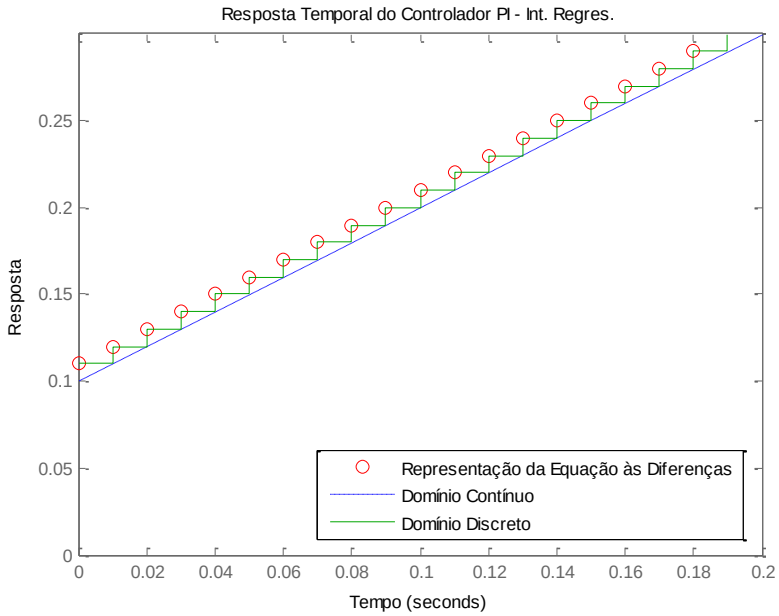


Figura 2.3.17 – Resposta do Controlador PI pela equação às diferenças integração regressiva com T_S=0,01 s, K_P=0,1 e K_I=1

2.3.3.3– Integração Trapezoidal no Controlador PI

Pela integração trapezoidal, partindo da equação 2.1.15, substituindo s pela expressão equivalente de z, tem-se:

$$C_{PI}(s)=\frac{sK_P+K_I}{s} \rightarrow C_{PI\,T}(z)=\frac{\frac{2}{T_s}\frac{z-1}{z+1}K_P+K_I}{\frac{2}{T_s}\frac{z-1}{z+1}}$$

$$C_{PI\,T}(z)=\frac{(2K_P+K_IT_s)z-2K_P+K_IT_s}{2z-2} \qquad 2.3.40$$

Substituindo as constantes obtém-se, para K_P=0,1, K_I=1:

$$C_{PI\,T}(z)=\frac{0{,}21z-0{,}19}{2z-2}=\frac{0{,}105z-0{,}095}{z-1} \qquad 2.3.41$$

E utilizando mais uma vez mais o degrau unitário com integração progressiva, tem-se:

$$C_{PI\,T\,Step}(z)=\frac{0{,}105z-0{,}095}{z-1}\frac{1}{z-1}=\frac{0{,}105z-0{,}095}{(z-1)^2} \qquad 2.3.42$$

E através do mesmo método de transformadas inversas de z utilizado anteriormente:

$$\frac{C_{PI\,T\,Step}(z)}{z}=\frac{0{,}105z-0{,}095}{z(z-1)^2}$$

$$\frac{C_{PI\,T\,Step}(z)}{z}=\frac{A}{z}+\frac{B}{z-1}+\frac{C}{(z-1)^2}$$

A=0,095 B=0,095 C=0,01

$$C_{PI\,T\,Step}(z)=-\frac{0{,}095z}{z}+\frac{0{,}095}{z-1}+\frac{0{,}01z}{(z-1)^2}$$

$$C_{PI\,T\,Step}(nT_s)=a+0{,}095+0{,}01k \qquad 2.3.43$$

com $a=\begin{bmatrix}0{,}095,\ k=0\\ 0,\ k\neq 0\end{bmatrix}$

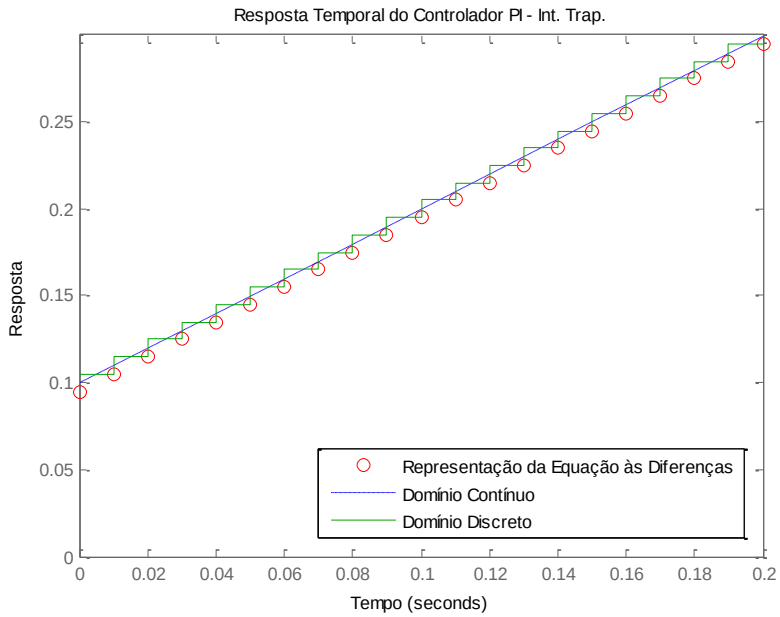


Figura 2.3.18 – Resposta do Controlador PI pela equação às diferenças integração trapezoidal com T_s=0,01 s, K_P=0,1 e K_I=1

2.3.3.4– Resposta do Controlador PI pelas Diferentes Integrações

Utilizando um código semelhante ao utilizado nos filtros, obtêm-se os seguintes resultados:

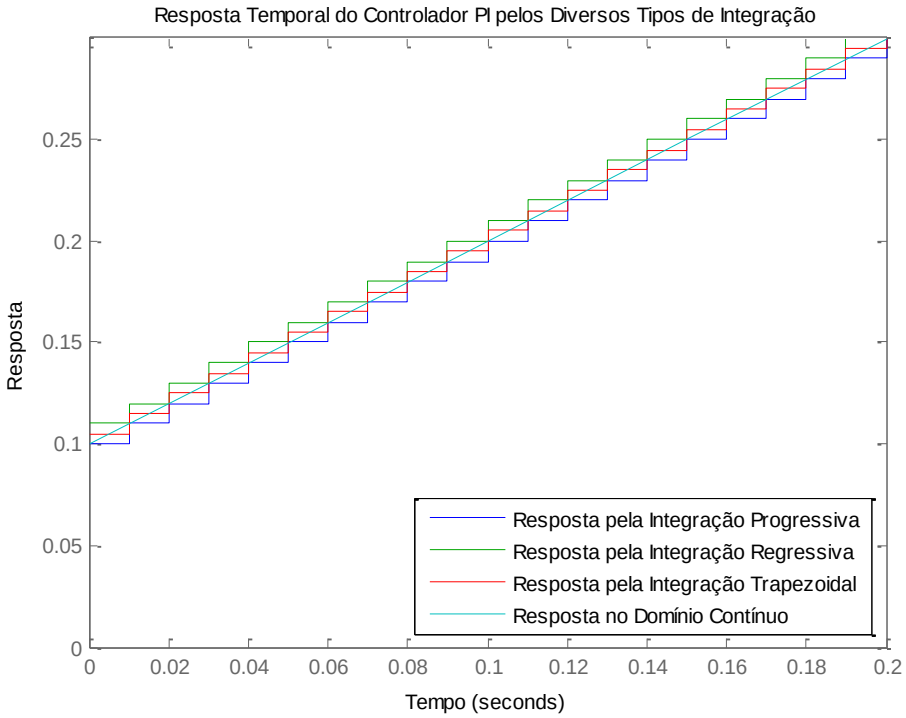


Figura 2.3.19 – Resposta temporal a uma entrada do tipo degrau do Controlador PI pelas equações às diferenças várias integrações com T_S=0,01 s, K_P=0,1 e K_I=1

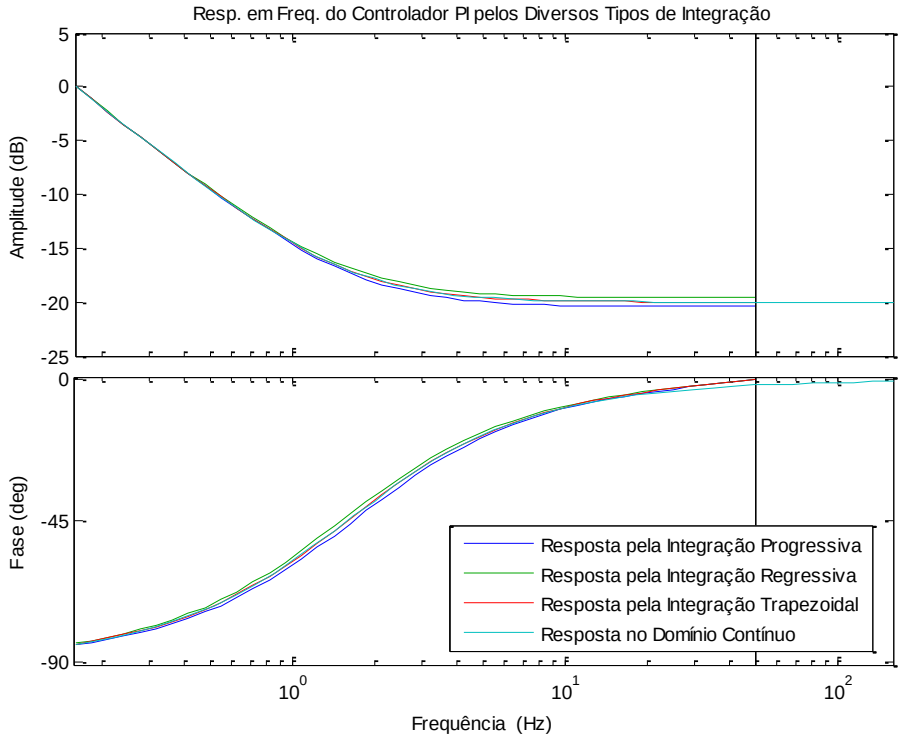


Figura 2.3.20 – Resposta em frequência do Controlador PI pelas equações às diferenças várias integrações com T_S=0,01 s, K_P=0,1 e K_I=1

O código utilizado para a obtenção das respostas encontra-se no Anexo 3.

No caso do controlador PI, a resposta em frequência leva a concluir que, com o aumento da frequência, a resposta do controlador não é alterada no domínio discreto em nenhum método de integração. Quanto ao erro da resposta, considerando as respostas do controlador apresentadas nos pontos anteriores, percebe-se que o cálculo do erro resultaria numa onda com evolução temporal em forma de dente de serra, e constante, não havendo necessidade de apresentar graficamente.

2.3.4 – Integração no Controlador PID

A função de transferência do controlador PI permite esboçar uma resposta temporal e em frequência nos diferentes domínios, devido ao número de zeros e polos ser idêntico. No entanto no controlador PID não é possível a mesma representação, a menos que se considere o controlador em cadeia fechada, ainda que sem nenhum sistema a controlar (igual a 1). Assim sendo, neste ponto não são apresentadas quaisquer respostas, apenas as funções de transferência pelos vários métodos de integração.

2.3.4.1– Integração Progressiva no Controlador PID

A substituição de *s* pela expressão equivalente de *z* é, no controlador PID, idêntica às restantes aplicações. Assim, partindo da equação 2.1.29:

$$C_{PID}(s)=\frac{s^2K_D+sK_P+K_I}{s} \rightarrow C_{PID\,P}(z)=\frac{\left(\frac{z-1}{T_s}\right)^2 K_D+\left(\frac{z-1}{T_s}\right)K_P+K_I}{\frac{z-1}{T_s}}$$

$$C_{PID\,P}(z)=\frac{K_Dz^2+(-2K_D+T_sK_P)z+K_D-T_sK_P+T_s^2K_I}{T_sz-T_s} \qquad 2.3.44$$

Como a função de transferência no domínio discreto para integração progressiva possui mais polos que zeros, a representação da resposta temporal a uma entrada do tipo degrau unitário não é possível, tal como explicado anteriormente neste trabalho. Deste modo, não é desenvolvida a função *y*(*k*) para o controlador PID com integração progressiva.

2.3.4.2 – Integração Regressiva no Controlador PID

Pela integração regressiva, as equações são, partindo da equação 2.1.29:

$$C_{PID}(s)=\frac{s^2K_D+sK_P+K_I}{s} \rightarrow C_{PID\,B}(z)=\frac{\left(\frac{z-1}{T_sz}\right)^2 K_D+\left(\frac{z-1}{T_sz}\right)K_P+K_I}{\frac{z-1}{T_sz}}$$

$$C_{PID\,B}(z)=\frac{(K_D+T_sK_P+T_s^2K_I)z^2+(-2K_D.-T_sK_P)z+K_D}{T_sz^2-T_sz} \qquad 2.3.45$$

Neste caso, o número de polos e zeros é igual. No entanto, como os polos existentes são em zero, a representação temporal da resposta a uma entrada do tipo degrau não é possível de ser representada. Além disso, não sendo possível representar a resposta temporal no domínio contínuo não seria possível a comparação de resultados.

2.3.4.3 – Integração Trapezoidal no Controlador PID

Na integração trapezoidal, substituído *s* pela expressão equivalente de *z* tem-se, partindo da equação 2.1.29:

$$C_{PID}(s)=\frac{s^2K_D+sK_P+K_I}{s} \rightarrow C_{PID\ T}(z)=\frac{\left(\frac{2}{T_s}\frac{z-1}{z+1}\right)^2 K_D+\left(\frac{2}{T_s}\frac{z-1}{z+1}\right)K_P+K_I}{\frac{2}{T_s}\frac{z-1}{z+1}}$$

$$C_{PID\ T}(z)=\frac{\left(4K_D+2T_sK_P+T_s^2K_I\right)z^2+\left(-8K_D+2T_s^2K_I\right)z+4K_D-2T_sK_P+T_s^2K_I}{2T_sz^2-2T_s} \qquad 2.3.46$$

O cálculo da função de *y*(*k*) a uma entrada do tipo degrau está comprometido pelas razões já descritas nas integrações do controlador PID pelos restantes métodos, progressivo e regressivo.

2.4– Estabilidade

A estabilidade avalia o comportamento dos sistemas dinâmicos face a variações. A capacidade de um sistema se equilibrar quando sofre uma perturbação descreve o comportamento de um sistema estável. Um sistema diz-se instável quando o valor da sua saída não tende para o valor de entrada ou quando, dependendo do sistema, não o faz num determinado tempo.

Neste capítulo é feito um estudo sobre a estabilidade dos sistemas apresentados dividindo-se em estabilidade absoluta (qualitativa) e estabilidade relativa (quantitativa). Olhando para a Figura 2.4.1, observa-se a aplicação de um filtro e de um controlador, em que no controlador existe uma cadeia fechada com o objetivo de controlar um sistema (processo). A estabilidade de um sistema pode ser analisada através da equação característica $1+G(s)H(s)=0$, que representa o denominador de uma função de transferência de um sistema em malha fechada, que pode ser proveniente do fecho de um sistema em malha aberta (introdução de realimentação no sistema) [6].

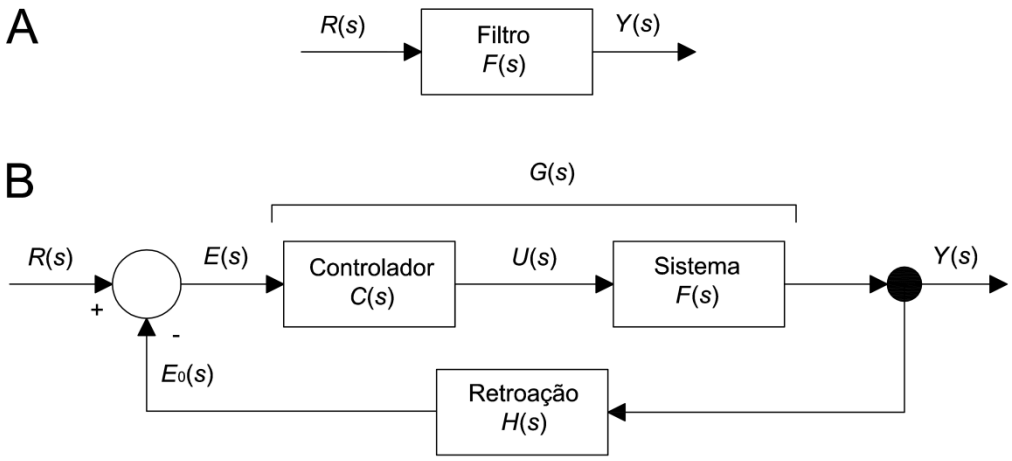


Figura 2.4.1 – Diagrama de Blocos de sistema com filtro (A) e com controlador (B)

Observando a Figura 2.4.1, a Função de Transferência do sistema com controlador é dada por:

$$\frac{Y(s)}{R(s)}=\frac{C(s)F(s)}{1+C(s)F(s)H(s)} \qquad 2.4.1$$

Onde a equação característica é dada por:

$$1+C(s)F(s)H(s)=0 \qquad 2.4.2$$

2.4.1– Mapeamento do SPCE do plano *s* para o plano *z*

Num sistema contínuo no tempo e pertencente a uma cadeia fechada, a localização dos polos determina a sua estabilidade e comportamento dinâmico. Por forma a poder prever também no domínio discreto o comportamento dos sistemas e estudar os limites de estabilidade dos mesmos, é importante conseguir representar o plano de *z* no plano *s* e estudar o sistema pela localização dos seus polos.

Seja $s=\sigma+j\omega$ um ponto no plano de *s*, ao longo do eixo de $j\omega$ tem-se:

$$z=e^{st}=e^{\sigma T_s}e^{j\omega T_s} \qquad 2.4.3$$

Como neste eixo $\sigma=0$, tem-se:

$$z=e^{j\omega T_s}=\cos\omega T_s+j\times sen\ \omega T_s \qquad 2.4.4$$

No plano de *z*, esta função representa um círculo unitário. No domínio contínuo, o sistema será estável se todos os polos se encontrarem no Semiplano Complexo Esquerdo (SPCE) do plano *s*. Representando esta condição no plano de *z*, um sistema discreto será estável se todos os polos se encontrarem dentro da circunferência de raio 1 centrada na origem do plano *z*.

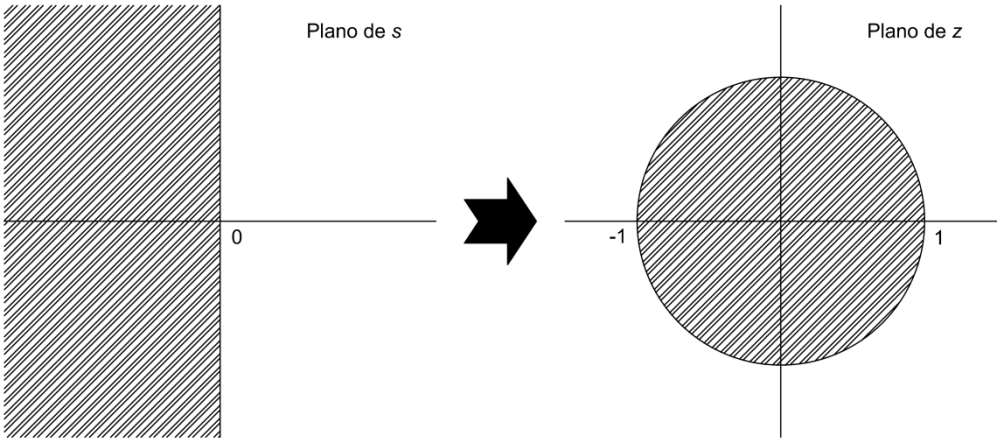


Figura 2.4.2 – Limites de estabilidade nos planos de *s* e *z*

Um sistema discreto pode ser instável devido às características de estabilidade idênticas às dos sistemas contínuos. Por outro lado, se a frequência à entrada do sistema discreto for superior à frequência de *Nyquist* ocorre distorção do sinal pelo facto de se verificar uma sobreposição espectral. Além da frequência de entrada, também o método de integração utilizado resulta em diferentes limites de estabilidades do sistema. Este capítulo tem por objetivo apresentar os limites de estabilidade face à discretização de sistemas pelos diferentes métodos de integração e, para isso, é feito o mapeamento do semiplano complexo esquerdo do plano *s* (onde o sistema é estável no domínio contínuo) no plano *z* segundo os métodos de integração estudados neste trabalho (progressiva, regressiva e trapezoidal).

2.4.1.1 – Mapeamento de *s* em *z* – Método de integração progressiva

A partir da relação de transformação de *s* para *z* do método de integração progressiva:

$$s=\frac{z-1}{T_s}$$

$$z=1+T_s s$$

O limite de estabilidade para $s=j\omega$ é:

$$z=1+T_s j\omega \qquad 2.4.5$$

Esta função pode ser representada no plano de *z*, como toda a parte à esquerda de 1. Assim, conclui-se que sistemas contínuos discretizado pelo método de integração progressiva podem ser instáveis no domínio discreto [7].

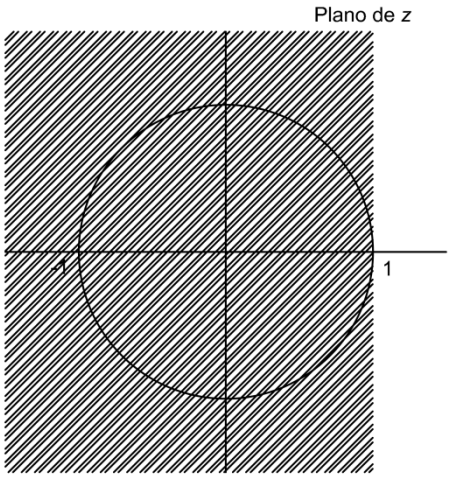


Figura 2.4.3 – Limites de estabilidade no plano de *z* utilizando integração progressiva

2.4.1.2– Mapeamento de *s* em *z* – Método de integração regressiva

A partir da relação de transformação de *s* para *z* pelo método de integração regressiva [8]:

$$s=\frac{z-1}{T_s z} \qquad z=\frac{1}{1-T_s s}$$

Para que seja garantida a condição de estabilidade, com R_e a parte real:

$$R_e(s)<0 \qquad 2.4.6$$

Ou seja, que os polos se encontrem no SPCE. Então:

$$R_e\left(\frac{z-1}{T_s z}\right)<0 \qquad 2.4.7$$

Como T_s é sempre positivo, a equação 2.4.7 pode escrever-se:

$$R_e\left(\frac{z\text{-}1}{z}\right)<0 \qquad 2.4.8$$

Escrevendo z na forma algébrica, $z=\alpha+j\beta$:

$$R_e\left(\frac{\alpha\text{-}1+j\beta}{\alpha+j\beta}\right)<0 \rightarrow R_e\left(\frac{(\alpha\text{-}1+j\beta)(\alpha-j\beta)}{(\alpha+j\beta)(\alpha-j\beta)}\right)<0 \rightarrow R_e\left(\frac{\alpha^2\text{-}\alpha+\beta^2+j\beta}{\alpha^2+\beta^2}\right)<0$$

$$\alpha^2\text{-}\alpha+\beta^2<0$$

Sabendo que a equação genérica de uma circunferência no plano x,y é dada por:

$$(x\text{-}a)^2+(y\text{-}b)^2=r^2$$

Onde a e b são os deslocamentos do centro da circunferência no eixo x e y respetivamente, e r é o raio da circunferência, é possível uma transformação de forma a obter a equação 2.4.9:

$$\left(\alpha\text{-}\frac{1}{2}\right)^2+\beta^2<\left(\frac{1}{2}\right)^2 \qquad 2.4.9$$

Que representa no plano de z uma circunferência de raio ½ centrada em ½.

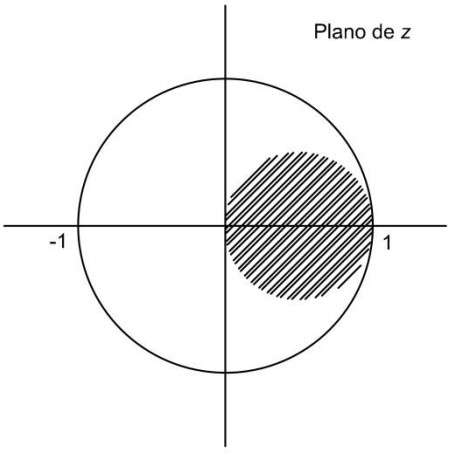


Figura 2.4.4 – Limites de estabilidade no plano z utilizando integração regressiva

Ainda que um sistema discretizado pelo método de integração regressiva seja estável, por os polos se encontrarem na zona de estabilidade do plano z, o projeto de um controlador, por exemplo, torna-se mais complexo e o desempenho é limitado, consequência da localização dos polos do sistema no plano z ser restrita à circunferência de raio 0,5 apresentada na Figura 2.4.4.

Este método de integração apresenta ainda deformações de fase e amplitude na característica de resposta em frequência. O mesmo acontece no método de integração progressiva, com a desvantagem ainda de se poderem originar sistemas instáveis.

2.4.1.3 – Mapeamento de *s* em *z* – Método de integração trapezoidal

A partir da relação de transformação de *s* para *z* do método de integração regressiva [8]:

$$s=\frac{2}{T_s}\frac{z-1}{z+1} \qquad z=\frac{1+\frac{T_s}{2}s}{1-\frac{T_s}{2}s}$$

Como

$$R_e\left(\frac{2}{T_s}\frac{z-1}{z+1}\right)<0 \rightarrow R_e\left(\frac{z-1}{z+1}\right)<0 \qquad 2.4.10$$

$$R_e\left(\frac{\alpha-1+j\beta}{\alpha+1+j\beta}\right)<0 \rightarrow R_e\left(\frac{(\alpha-1+j\beta)(\alpha+1-j\beta)}{(\alpha+1+j\beta)(\alpha+1-j\beta)}\right)<0 \rightarrow R_e\left(\frac{\alpha^2-1+\beta^2+j2\beta}{(\alpha-1)^2+\beta^2}\right)<0$$

Simplificando tem-se:

$$\alpha^2+\beta^2<1 \qquad 2.4.11$$

Então o mapeamento do semiplano complexo esquerdo em *z* é todo o círculo de raio 1 centrado em 0.

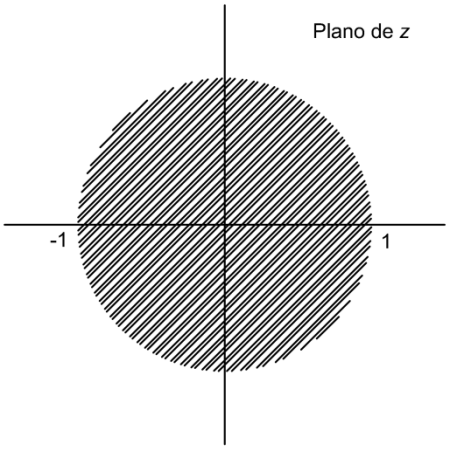


Figura 2.4.5 – Limites de estabilidade no plano *z* utilizando integração trapezoidal

Neste método de integração, é garantida a manutenção da estabilidade no filtro. No entanto, não são preservadas as larguras de banda, havendo uma deformação na escala de frequências e na característica da fase [8].

2.4.2 – Estabilidade Absoluta

A estabilidade absoluta trata-se do critério qualitativo que define se um sistema é estável, ou não, sem quantificar esta estabilidade. Para isso recorre-se, normalmente, ao cálculo dos polos da equação do filtro ou, no caso de um controlador, da equação característica do sistema e, dependendo do seu domínio, consegue-se saber se o sistema é efetivamente estável.

Para sistemas com ordem superior à segunda, recorrem-se a outros métodos, tais como o critério de estabilidade de *Routh-Hurwitz*, no domínio contínuo, e teste de estabilidade de *Jury*, no domínio discreto, que, ainda que funcionem também em sistemas de 2.ª ou menor ordem, apresentam maior dificuldade face ao simples cálculo dos polos [1].

2.4.2.1 – Critério de Estabilidade de Estabilidade de Routh-Hurwitz

Este critério, puramente matemático, pretende avaliar a estabilidade absoluta partindo da equação característica do conjunto sistema-filtro/controlador.

Considerando uma equação característica de n-ésima ordem do tipo:

$$a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0 = 0 \qquad 2.4.12$$

Para que o sistema seja estável é necessário que todos os coeficientes *a* desta equação existam e que tenham todos o mesmo sinal. A tabela de *Routh* tem a seguinte forma:

Tabela 2.4.1 – Tabela de *Routh*, domínio contínuo

s^0	q_1	q_2	q_3	…	0
s^1	p_1	p_2	p_3	…	0
…	…	…	…	…	…
s^{n-3}	c_1	c_2	c_3	…	0
s^{n-2}	b_1	b_2	b_3	…	0
s^{n-1}	a_{n-1}	a_{n-3}	a_{n-5}	…	0
s^n	a_n	a_{n-2}	a_{n-4}	…	0

O sistema é estável se todos os coeficientes da primeira coluna existirem e tiverem o mesmo sinal. Isto é, todos diferentes de 0 e de ∞ e todos negativos ou positivos. A necessidade dos coeficientes terem todos o mesmo sinal provém do número de mudanças de sinal nos coeficientes ocorridas ao percorrer essa coluna ser igual ao número de raízes da equação, isto é, se houver, por exemplo, apenas um coeficiente de sinal negativo, significa que existem duas mudanças de sinal ao percorrer a coluna, o que significa que existem dois polos no SPCD e o sistema é, assim, instável.

Para o preenchimento desta tabela é necessário ter em conta um conjunto de considerações:

- Numa linha vertical marcam-se de baixo para cima as direções das potências decrescentes de s da equação característica;
- Os coeficientes a_i da equação característica colocam-se apenas nas direções de s^n e s^{n-1}, como se indica na tabela acima;
- Os coeficientes b_i, c_i, …, p_i, q_i calculam-se a partir dos a_i;
- A tabela continua na vertical e na horizontal até que se obtenham zeros;
- Pode multiplicar-se ou dividir-se qualquer linha por uma constante diferente de zero antes que a linha seguinte seja calculada.

Os valores dos coeficientes b_i serão calculados da seguinte forma:

$$b_1=\frac{1}{a_{n-1}}\begin{vmatrix} a_{n-1} & a_{n-3} \\ a_n & a_{n-2} \end{vmatrix}=\frac{a_{n-1}a_{n-2}-a_n a_{n-3}}{a_{n-1}} \qquad 2.4.13$$

$$b_2=\frac{1}{a_{n-1}}\begin{vmatrix} a_{n-1} & a_{n-5} \\ a_n & a_{n-4} \end{vmatrix}=\frac{a_{n-1}a_{n-4}-a_n a_{n-5}}{a_{n-1}} \qquad 2.4.14$$

$$c_1=\frac{1}{b_1}\begin{vmatrix} b_1 & b_2 \\ a_{n-1} & a_{n-3} \end{vmatrix}=\frac{b_1 a_{n-3}-a_{n-1}b_2}{b_1} \qquad 2.4.15$$

Para um sistema no domínio discreto, o critério de *Routh* pode também ser aplicado se for considerada uma transformação bilinear para transformar o SPCE do plano *s* no interior do círculo unitário no plano *z*. Nesta transformação, substitui-se *z*:

$$z=\frac{1+w}{1-w}$$

Dada a equação característica em *w*:

$$F(w)=b_n w^n+b_{n-1}w^{n-1}+\ldots b_1 w+b_0=0 \qquad 2.4.16$$

E a tabela de *Routh* é formada da seguinte forma [1]:

Tabela 2.4.2 – Tabela de *Routh*, domínio discreto

w^n	b_n	b_{n-2}	b_{n-4}	…
w^{n-1}	b_{n-1}	b_{n-3}	b_{n-5}	…
w^{n-2}	c_1	c_2	c_3	…
…	…	…	…	…
w^1	j_1			
w^0	k_1			

Onde as primeiras duas linhas são obtidas diretamente da equação característica e as restantes são à custa das expressões:

$$c_1=\frac{b_{n-1}b_{n-2}-b_nb_{n-3}}{b_{n-1}},\quad c_2=\frac{b_{n-1}b_{n-4}-b_nb_{n-5}}{b_{n-1}},\qquad c_3=\frac{b_{n-1}b_{n-6}-b_nb_{n-7}}{b_{n-1}},$$

$$d_1=\frac{c_1b_{n-3}-b_{n-1}c_2}{c_1},\qquad \dots\ ,\qquad \dots\ ,$$

Aplicando o critério de *Routh* a um sistema discreto, o critério de estabilidade é o mesmo para o domínio contínuo. Assim, todos os elementos da coluna da esquerda têm de ter o mesmo sinal, pois o número de polos no SPCD do plano *s* é igual ao número de mudanças de sinal encontradas ao percorrer esta mesma coluna e, havendo polos no SPCD, o sistema é instável.

2.4.2.2 – Teste de Estabilidade de Jury

O teste de estabilidade de *Jury* é semelhante ao critério de estabilidade de *Routh* no domínio contínuo. Este teste é apenas para o domínio discreto e pode ser aplicado a equações características de qualquer ordem ainda que aumente a dificuldade com o aumento do grau.

Para o teste *Jury*, a equação característica expressa-se:

$$F(z)=a_nz^n+a_{n-1}z^{n-1}+\dots+a_1z+a_0=0 \qquad 2.4.17$$

Em que $a_n>0$. Para aplicar o critério de estabilidade é necessário preencher a seguinte tabela [1]:

z^0	z^1	z^2	…	z^{n-k}	…	z^{n-1}	z^n
a_0	a_1	a_2	…	a_{n-k}	…	a_{n-1}	a_n
a_n	a_{n-1}	a_{n-2}	…	a_k	…	a_1	a_0
b_0	b_1	b_2	…	b_{n-k}	…	b_{n-1}	
b_{n-1}	b_{n-2}	b_{n-3}	…	b_{k-1}	…	b_0	
c_0	c_1	c_2	…	c_{n-k}	…		
c_{n-2}	c_{n-3}	c_{n-4}	…	c_{k-2}	…		
…	…	…	…	…			
…	…	…	…	…			
l_0	l_1	l_2	l_3				
l_3	l_2	l_1	l_0				
m_0	m_1	m_2					

Figura 2.4.6 – Tabela de *Jury*, domínio discreto

Onde os termos são definidos pelas seguintes equações:

$$b_k=\begin{vmatrix}a_0 & a_{n-k}\\ a_n & a_k\end{vmatrix},\quad c_k=\begin{vmatrix}b_0 & b_{n-k-1}\\ n_{n-1} & b_k\end{vmatrix},\quad d_k=\begin{vmatrix}c_0 & c_{n-2-k}\\ c_{n-2} & c_k\end{vmatrix},\quad \dots$$

Deste modo, as condições necessárias e suficientes para provar a estabilidade do sistema são:

- $F(1)>0$
- $(-1)^n F(-1)>0$
- $|a_0|<a_n$

Se uma destas condições não se verificar, o sistema é instável. Se todas elas forem verdadeiras, é necessário ainda que:

- $|b_0|>b_{n-1}$
- $|c_0|>c_{n-2}$
- $|d_0|>d_{n-3}$
-
- $|m_0|>m_2$

O teste de estabilidade de *Jury* para sistemas de 2.ª e 3.ª ordem pode ser simplificado. Assim, em sistemas de segunda ordem, em que a equação característica é:

$$F(z)=a_2z^2+a_1z+a_0 \ , \qquad \text{com } a_2>0$$

É garantido que os polos do sistemas estarão dentro do circulo unitário se:

$$F(1)>0, \quad F(-1)>0, \quad |a_0|<a_2$$

Num sistema de terceira ordem, com a equação:

$$F(z)=a_3z^3+a_2z^2+a_1z+a_0 \ , \qquad \text{com } a_3>0$$

O sistema é estável se:

$$F(1)>0, \quad F(-1)>0, \quad |a_0|<a_3$$

$$\left|det\begin{bmatrix} a_0 & a_3 \\ a_3 & a_0 \end{bmatrix}\right| > \left|det\begin{bmatrix} a_0 & a_1 \\ a_3 & a_2 \end{bmatrix}\right|$$

2.4.3– Equações dos Sistemas a Estudar

A estabilidade pode ser estudada tanto no domínio contínuo como no domínio discreto. O interessante neste trabalho é a comparação entre a análise nos dois domínios e o estudo dos limites da estabilidade nos mesmos.

Tendo em conta que, conforme apresentado no ponto anterior, um sistema integrado pelo método trapezoidal ou regressivo é sempre estável se também o for no domínio contínuo, apenas serão calculados os limites de estabilidade para um sistema quando integrado pelo método progressivo, considerando que o mesmo sistema no domínio contínuo é estável.

Para isso é necessário ter em consideração as seguintes equações, provenientes dos pontos de 2.3:

Tabela 2.4.3 – Equações de Transferência dos filtros e controladores pelo método de integração progressivo

Sistema	Equação no Domínio Contínuo	Equação no Domínio Discreto (Integração Progressiva)
Filtro PB de 1.ª Ordem	$F(s)=G_0\frac{\omega_c}{s+\omega_c}$	$F_P(z)=G_0\frac{\omega_c T_s}{z+\omega_c T_s-1}$
Filtro PB de 2.ª Ordem	$F_2(s)=G_0\frac{\omega_c^2}{s^2+2\zeta\omega_c s+\omega_c^2}$	$F_{2\,P}(z)$
Controlador PI	$C_{PI}(s)=\frac{sK_P+K_I}{s}$	$C_{PI\,P}(z)=\frac{\frac{K_P}{T_s}z-\frac{K_P}{T_s}+K_I}{\frac{1}{T_s}z-\frac{1}{T_s}}$
Controlador PID	$C_{PID}(s)=\frac{s^2K_D+sK_P+K_I}{s}$	$C_{PID\,P}(z)$

Com:

$$F_{2\,P}(z)=G_0\frac{\omega_c^2T^2}{z^2+(2\zeta\omega_c T_s-2)z+1-2\zeta\omega_c T_s+\omega_c^2T_s^2}$$

$$C_{PID\,P}(z)=\frac{K_Dz^2+(-2K_D+T_sK_P)z+K_D-T_sK_P+T_s^2K_I}{T_sz-T_s}$$

2.4.3.1 – Estabilidade Absoluta de um Filtro PB de 1.ª Ordem

No domínio contínuo tem-se:

$$F(s)=G_0\frac{\omega_c}{s+\omega_c}$$

Para um ganho G_0 unitário, o polo do sistema é:

$$s+\omega_c=0$$

$$s=-\omega_c$$

Para qualquer valor de $\omega_c>0$ o sistema no domínio contínuo é estável pois encontra-se no SPCE.

No domínio discreto, pela integração progressiva:

$$F_P(z)=G_0\frac{\omega_c T_s}{z+\omega_c T_s-1}$$

Onde os polos, para um ganho G_0 unitário, é:

$$z+\omega_c T_s-1=0$$

$$z=1-\omega_c T_s$$

Para que o polo esteja dentro da circunferência de raio 1 centrada na origem o valor absoluto do polo tem de ser inferior a 1:

$$|1-\omega_c T_s|<1$$

Em que

$0<\omega_c T_s$ para que o sistema seja estável.

Como ω_c e T_s são números positivos, o filtro passa-baixo de 1.ª ordem no domínio discreto é sempre estável pela integração progressiva, assim como o será nas integrações regressiva e trapezoidal.

2.4.3.2 – Estabilidade Absoluta de um Filtro PB de 2.ª Ordem (Butterworth)

No domínio contínuo tem-se:

$$F_2(s)=G_0\frac{\omega_c^2}{s^2+2\zeta\omega_c s+\omega_c^2}$$

Para um ganho G_0 unitário, os polos são calculados a partir de:

$$s^2+2\zeta\omega_c s+\omega_c^2=0$$

$$s=-\frac{2\zeta\omega_c}{2}+\frac{2\zeta\omega_c}{2}j \qquad s=-\frac{2\zeta\omega_c}{2}-\frac{2\zeta\omega_c}{2}j$$

Como ambos os polos se encontram no SPCE, para qualquer $\omega_c>0$ e $2\zeta>0$, o sistema é estável

No domínio discreto, pela integração progressiva:

$$F_{2\,P}(z)=G_0\frac{\omega_c^2T^2}{z^2+(2\zeta\omega_c T_s-2)z+1-2\zeta\omega_c T_s+\omega_c^2 {T_s}^2}$$

Considerando um ganho G_0 unitário, os polos são:

$$z^2+(2\zeta\omega_c T_s-2)z+1-2\zeta\omega_c T_s+\omega_c^2 {T_s}^2=0$$

Em que, recorrendo à fórmula resolvente e desenvolvendo, se tem que para o sistema ser estável é necessário verificar as condições:

$$2\zeta\omega_c T_s-\omega_c^2 {T_s}^2<1 \qquad \omega_c T_s>0$$

2.4.3.3– Estabilidade Absoluta de um Controlador PI

Para o estudo da estabilidade do controlador PID, toma-se como exemplo o sistema correspondente ao anel interior da máquina DC [9]. É controlada a corrente neste anel através de um controlador PI ou PID. O diagrama de blocos da máquina encontra-se na Figura 2.4.7.

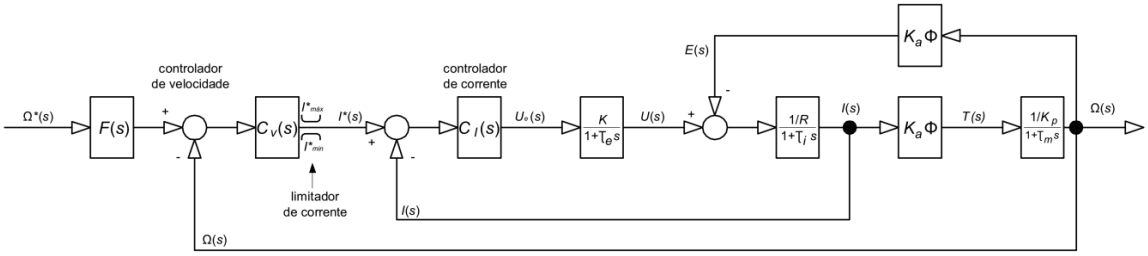


Figura 2.4.7 – Diagrama de blocos do controlo de Velocidade com cadeia subordinada de corrente

Função de transferência do controlador PI:

$$C_{PI}(s)=K_P+\frac{K_I}{s}=\frac{K_I\left(\frac{K_P}{K_I}s+1\right)}{s}=\frac{K_I(\tau_C s+1)}{s} \qquad 2.4.18$$

Função de transferência do anel interior da máquina de corrente contínua:

$$G(s)=\frac{\frac{K_e}{R_a}}{(\tau_e s+1)(\tau_a s+1)} \qquad 2.4.19$$

Função de transferência do sistema em malha fechada com controlador:

$$\frac{I(s)}{I^*(s)}=\frac{\frac{K_I(\tau_C s+1)}{s}\frac{\frac{K_e}{R_a}}{(\tau_e s+1)(\tau_a s+1)}}{1+\frac{K_I(\tau_C s+1)}{s}\frac{\frac{K_e}{R_a}}{(\tau_e s+1)(\tau_a s+1)}}$$

Como o polo da componente elétrica está mais próximo de 0, é o polo que o controlador procura anular ($\tau_c=\tau_a$). Assim, simplificando obtém-se:

$$\frac{I(s)}{I^*(s)}=\frac{\frac{K_I K_e}{R_a}}{\tau_e s^2+s+\frac{K_I K_e}{R_a}}$$

Onde a equação característica é dada por:

$$\tau_e s^2+s+\frac{K_I K_e}{R_a}=0$$

Assim, pelo critério de *Routh*, preenche-se a tabela com:

s^{n-2}	b_1	0
s^{n-1}	1	0
s^n	τ_e	$\frac{K_I K_e}{R_a}$

Onde b_1 é calculado da seguinte forma:

$$b_1=\frac{1}{1}\begin{vmatrix}1 & 0\\ \tau_e & \frac{K_I K_e}{R_a}\end{vmatrix}=\frac{K_I K_e}{R_a}$$

Como todas as constantes são maiores que zero, todos os elementos da primeira coluna possuem o mesmo sinal (positivo). Assim, o sistema é estável pelo critério de *Routh*.

No domínio discreto, pela integração progressiva, a equação característica é dada por:

$$\tau_e\left(\frac{z-1}{T_s}\right)^2+\frac{z-1}{T_s}+\frac{K_I K_e}{R_a}=0$$

$$\tau_e z^2+(-2\tau_e+T_s)z+\tau_e-T_s+\frac{K_I K_e T_s^2}{R_a}=0$$

Para que o sistema seja estável, pelo critério de *Jury*, é necessário verificar a condição:

$$F(1)>0 \quad \rightarrow \quad \frac{K_I K_e T_s^2}{R_a}>0$$

Assim, a primeira condição é verificada pois todas as constantes têm valor positivo. É também necessário verificar a condição:

$$F(-1)>0 \quad \rightarrow \quad 4\tau_e-2T_s+\frac{K_I K_e T_s^2}{R_a}>0$$

É necessário ainda que:

$$|a_0|<a_n \quad \rightarrow \quad \left|\tau_e-T_s+\frac{K_I K_e T_s^2}{R_a}\right|<\tau_e$$

Simplificando, para que o sistema seja estável é necessário que:

$$4\tau_e-2T_s+\frac{K_I K_e T_s^2}{R_a}>0>-T_s+\frac{K_I K_e T_s^2}{R_a} \qquad 2.4.20$$

2.4.3.4– Estabilidade Absoluta de um Controlador PID

É também possível a utilização de um controlador PID na cadeia de corrente. Assim, a função de transferência do controlador PID:

$$C_{PI}(s)=K_P+\frac{K_I}{s}+K_D s=\frac{K_D s^2+K_P s+K_I}{s}=\frac{(\tau_1 s+1)(\tau_2 s+1)}{s} \qquad 2.4.21$$

Considerando a equação 2.4.19, a função de transferência do sistema em malha fechada com controlador:

$$\frac{I(s)}{I^*(s)}=\frac{\frac{(\tau_1 s+1)(\tau_2 s+1)}{s}\frac{\frac{K_e}{R_a}}{(\tau_e s+1)(\tau_a s+1)}}{1+\frac{(\tau_1 s+1)(\tau_2 s+1)}{s}\frac{\frac{K_e}{R_a}}{(\tau_e s+1)(\tau_a s+1)}}$$

O objetivo do controlador PID é anular os dois polos do sistema. Assim, tem-se:

$$\tau_1=\tau_e \quad \text{e} \quad \tau_2=\tau_a$$

E a função de transferência é:

$$\frac{I(s)}{I^*(s)}=\frac{\frac{\frac{K_e}{R_a}}{s}}{1+\frac{\frac{K_e}{R_a}}{s}}=\frac{K_e}{R_a s+K_e}$$

A equação característica é assim:

$$R_a s+K_e=0$$

Onde $s=-\frac{K_e}{R_a}$, logo o sistema é estável pois só possui um polo no SPCE.

No domínio discreto, utilizando a integração progressiva, a função de sistema é dada por:

$$\frac{I(z)}{I^*(z)}=\frac{K_e}{R_a\frac{2}{T_s}\frac{z-1}{T_s}+K_e}=\frac{K_e T_s^2}{2R_a z-1+K_e T_s^2}$$

A equação característica é dada por:

$$2R_a z-1+K_e T_s^2=0$$

$$z=\frac{K_e T_s^2-1}{2R_a}$$

Assim, para que o sistema seja estável é necessário que:

$$\frac{K_e T_s^2-1}{2R_a}<1$$

2.4.4– Estabilidade Relativa

A estabilidade relativa quantifica a estabilidade do sistema. O método mais utilizado passa pela obtenção das margens de ganho e margem de fase do sistema e avaliá-lo considerando intervalos de estabilidade pré-definidos. Esta estabilidade avalia o comportamento do sistema em malha aberta, para isso, considerando a Figura 2.4.1, é estudada a função de transferência [6]:

$$C(s)F(s)H(s)$$

A margem de ganho k_g é definida como sendo o inverso do módulo da função de transferência em cadeia aberta, à frequência para a qual o ângulo de fase da referida função é de -180º. Para um sistema ser estável é necessário que k_g seja superior a 1. Um sistema é tanto mais estável quanto maior for k_g.

A margem de fase γ é definida como sendo 180º mais o ângulo de fase da função de transferência em cadeia aberta para o qual o módulo da função de transferência do sistema $FT(j\omega)$ é unitário. Um sistema diz-se então estável quando γ>0º.

Para que um sistema seja efetivamente estável é então necessário verificarem-se estas duas condições k_g>1 e γ>0º. Para que um sistema seja considerado muito estável é necessário verificar:

- k_g > 6 dB
- 30º < γ < 60º

A margem de ganho e margem de fase podem ser obtidas pela análise do diagrama de bode das funções. O *Matlab* permite através do comando '*margin*' obter diretamente as margens e daí fazer a análise da estabilidade. Na Figura 2.4.8 é apresentada uma representação da análise a fazer ao diagrama de bode.

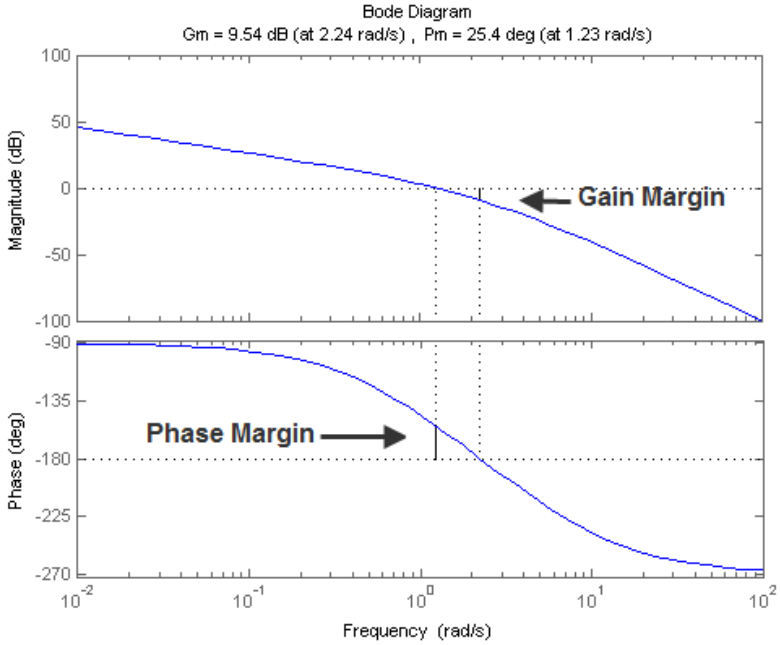


Figura 2.4.8 – Representação das margens de ganho e de fase

Se tivermos um sistema como o apresentado na Figura 2.4.1, em que *C*(*s*) corresponde à equação de um filtro ou controlador e *G*(*s*) ao sistema, a análise das margem de ganho e margem de fase é feita à função de transferência do sistema global.

2.4.4.1 – Estabilidade Relativa de um Controlador PI

Utilizando o sistema apresentado no ponto 2.4.3.3, para o valor das constantes:

K_I=5.000 K_e=6 R_a=3Ω τ_e=5×10^{-5} s

O conjunto do sistema e controlador tem a função de transferência:

$$\frac{I(s)}{I^*(s)}=\frac{\frac{K_IK_e}{R_a}}{\tau_e s^2+s+\frac{K_PK_IK_e}{R_a}}=\frac{2\text{x}10^8}{s^2+20.000s+2\text{x}10^8}$$

Utilizando a integração progressiva, a função de sistema é:

$$\frac{I(z)}{I^*(z)}=\frac{\frac{K_IK_e}{R_a}}{\tau_e\left(\frac{z-1}{T_s}\right)^2+\left(\frac{z-1}{T_s}\right)+\frac{K_IK_e}{R_a}}=\frac{\frac{K_IK_eT_s^2}{R_a}}{\tau_e z^2+(-2\tau_e+T_s)z+\tau_e-T_s+\frac{K_IK_eT_s^2}{R_a}}$$

Em que o limite de estabilidade, apresentado na equação 2.4.20, toma os valores:

$$0{,}0002-2T_s+10.000T_s^2>0>-T_s+10.000T_s^2 \qquad 2.4.22$$

Utilizando a função '*margin*' do programa *Matlab* obtém-se a Figura 2.4.9 onde se encontra o valor da margem de ganho e margem de fase, para um T_s de 10 µs.

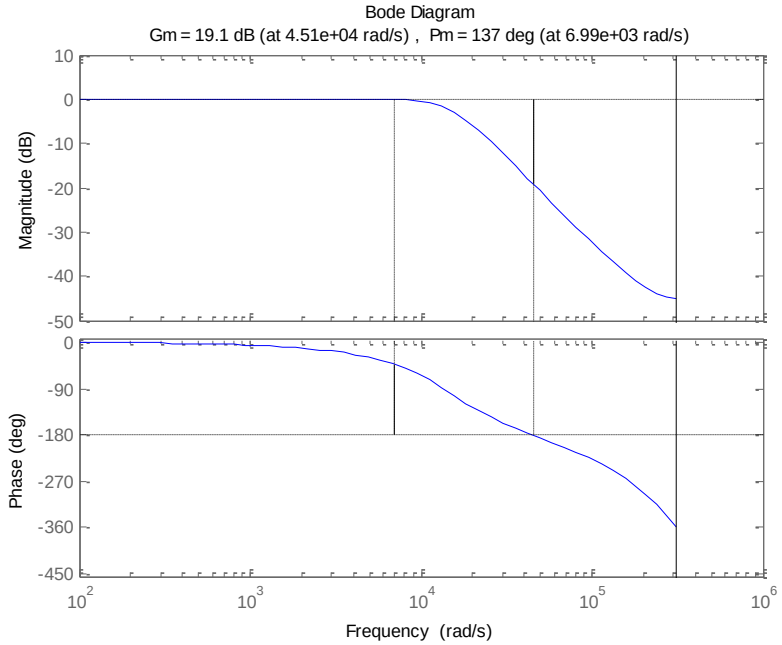


Figura 2.4.9 – Margem de ganho e de fase de um sistema com controlador PI para T_s=10 µs

O limite de estabilidade, substituindo as constantes na equação 2.4.22 é:

$$1{,}81\times10^{-4}>0>-9\times10^{-6}$$

Este limite mostra que o sistema é estável. Este facto é comprovado através da margem de ganho e margem de fase onde:

- k_g = 19,1dB, como k_g >6 ➔ Muito Estável
- ɣ = 137, como ɣ > 0º ➔ Estável

Se o tempo de amostragem for um valor tal, que não se verifique a condição de estabilidade, por exemplo T_s=100 µs, substituindo T_s na equação 2.4.22:

$$1\times10^{-4}>\mathbf{0>0}$$

Como a condição não se verifica, o sistema não é estável. Este facto pode ser verificado através da margem de ganho e margem de fase.

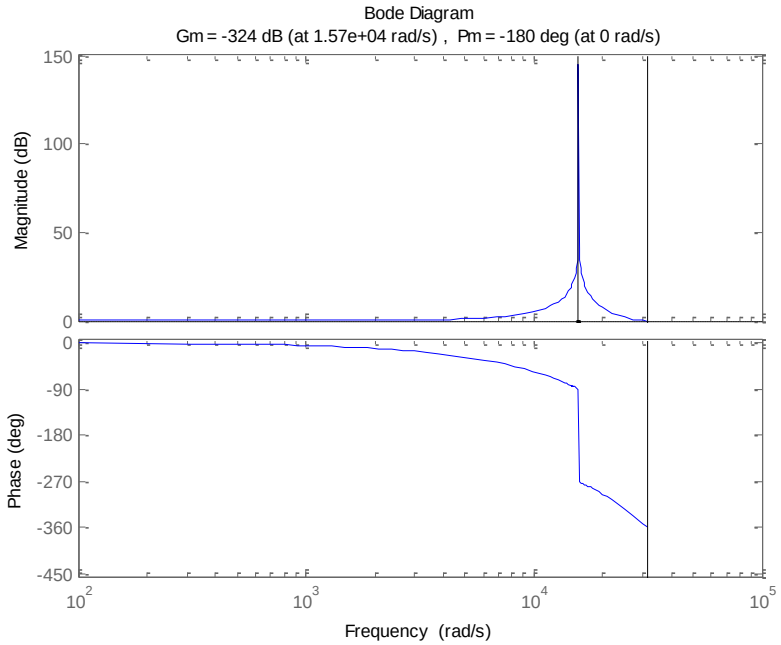


Figura 2.4.10 – Margem de ganho e de fase de um sistema com controlador PI para T_S=100 µs

Onde:

- k_g = -324 dB, como k_g < 1 ➔ Instável
- ɣ = -180º, como ɣ < 0º ➔ Instável

Esta demostração veio provar a existência dos limites de estabilidade em função do tempo de amostragem, quando no domínio discreto.

O código utilizado para obtenção destas figuras está apresentado no Anexo 4.

2.5– Diagrama de Fluxo e Equação às Diferenças

As equações às diferenças são utilizadas em sistemas discretos assim como as equações diferenciais em sistemas contínuos. Um sistema dinâmico linear discreto de ordem *n* com entrada *u*(*k*) e saída *y*(*k*) pode ser descrito por uma equação às diferenças [7]:

$$u(k\text{-}m)+a_{m\text{-}1}u(k+m\text{-}1)+\ldots+a_0u(k)=b_ny(k\text{-}n)+b_{n\text{-}1}y(k+n\text{-}1)+\ldots+y(k) \qquad 2.5.1$$

Que pode ser representado sob a forma de função de sistema, assumindo b_0=1:

$$U(z)\sum_{j=0}^{m}a_jz^{-j}=Y(z)\sum_{j=1}^{n}b_jz^{-j}+Y(z) \qquad 2.5.2$$

Para implementação de algoritmos de controlo em controladores com a forma de transformada de *z* polinomial são realizados no computador programas que contêm atrasos unitários, constantes multiplicativas e somadores. Matematicamente, todas as formas de implementar um filtro ou controlador digital são idênticas, diferindo apenas da forma como são implementadas. No entanto, diferentes formas de implementação têm diferentes comportamentos (eficiência, tempo de amostra, sensibilidade a erro de parâmetros, etc.) no controlador. O tipo de estrutura direta é uma das implementações mais comuns, e irá ser estudada neste capítulo, contendo as vertentes de estrutura canónica direta e estrutura não canónica direta.

Considerando os pontos anterior, a integração trapezoidal é a integração que possui menor erro associado em comparação com as respostas obtidas no domínio contínuo, pelo que neste capítulo apenas irão ser estudadas as equações para os filtros e controladores pela integração trapezoidal.

2.5.1 – Estrutura Canónica Direta

A partir da equação 2.5.2, uma função de sistema pode ser representada por:

$$\frac{Y(z)}{U(z)}=\frac{\sum_{j=0}^{m}a_jz^{-j}}{1+\sum_{j=1}^{n}b_jz^{-j}} \qquad 2.5.3$$

Com *m* a ordem do numerador e *n* a ordem do numerador, se for introduzida uma variável *W*(*z*):

$$\frac{Y(z)}{W(z)}\frac{W(z)}{U(z)}=\frac{\sum_{j=0}^{m}a_jz^{-j}}{1+\sum_{j=1}^{n}b_jz^{-j}} \qquad 2.5.4$$

Onde, isolando o termo *Y*(*z*)/*W*(*z*) e *U*(*z*)/*W*(*z*) :

$$\frac{Y(z)}{W(z)} = \sum_{j=0}^{m} a_j z^{-j} \qquad 2.5.5$$

$$\frac{U(z)}{W(z)} = 1 + \sum_{j=1}^{n} b_j z^{-j} \quad \rightarrow \quad U(z) = W(z)\left(1 + \sum_{j=1}^{n} b_j z^{-j}\right) \qquad 2.5.6$$

E assim sendo:

$$Y(z) = \sum_{j=0}^{m} a_j z^{-j} W(z) \qquad 2.5.7$$

$$W(z) = U(z) - \sum_{j=1}^{n} b_j z^{-j} W(z) \qquad 2.5.8$$

Passando para o domínio do tempo tem-se:

$$y_k = \sum_{j=0}^{m} a_j w_{k-j} \qquad 2.5.9$$

$$w_k = u_k - \sum_{j=1}^{n} b_j w_{k-j} \qquad 2.5.10$$

Consequentemente o diagrama de fluxo de sinais (ou grafo) tem a seguinte forma:

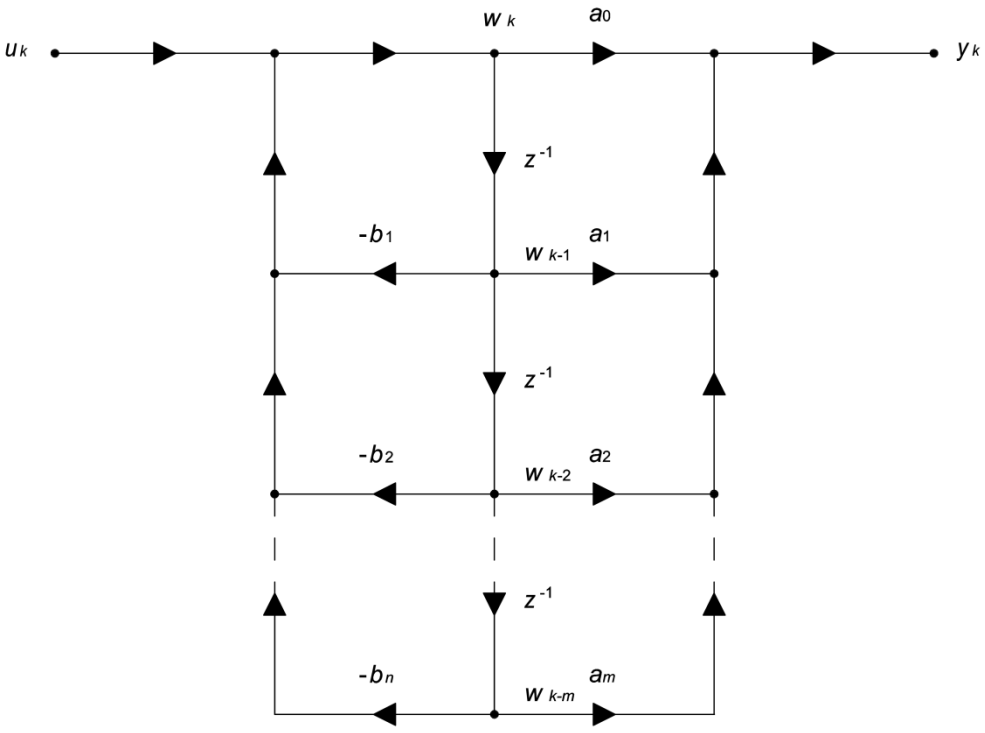


Figura 2.5.1 – Diagrama de blocos genérico de um sistema com estrutura canónica direta

Em que o par de equações às diferenças toma a forma:

$$y_k=w_k\left(a_0+a_1z^{-1}+a_2.z^{-2}+\ldots+a_mz^{-m}\right)$$
$$w_k=u_k-w_k\left(b_1z^{-1}+b_2z^{-2}+\ldots+b_nz^{-n}\right) \qquad 2.5.11$$

2.5.1.1– Representação Canónica do Filtro PB de 1.ª Ordem

Considerando a equação 2.3.13 obtida pelo método de integração trapezoidal, antes de aplicar uma entrada do tipo degrau unitário:

$$F_T(z)=G_0\frac{\omega_cT_sz+\omega_cT_s}{(2+\omega_cT_s)z-2+\omega_cT_s}=G_0\frac{\frac{\omega_cT_s}{2+\omega_cT_s}+\frac{\omega_cT_s}{2+\omega_cT_s}.z^{-1}}{1+\left(\frac{-2+\omega_cT_s}{2+\omega_cT_s}\right)z^{-1}} \qquad 2.5.12$$

Seja:

$$a_0=\frac{\omega_cT_s}{2+\omega_cT_s} \qquad a_1=\frac{\omega_cT_s}{2+\omega_cT_s} \qquad b_0=1 \qquad b_1=\frac{-2+\omega_cT_s}{2+\omega_cT_s}$$

$$F_T(z)=G_0\frac{a_0+a_1z^{-1}}{1+b_1z^{-1}} \qquad 2.5.13$$

Introduzindo a variável $W(z)$ e desenvolvendo, considerando G_0 unitário tem-se:

$$Y(z)=W(z)\left(a_0+a_1z^{-1}\right)$$
$$W(z)=U(z)-W(z)b_1z^{-1} \qquad 2.5.14$$

Que no domínio temporal corresponde a:

$$y_F=a_0w_F+a_1w_{F-1}$$
$$w_F=u_F-b_1w_{F-1} \qquad 2.5.15$$

E, assim sendo, é possível esboçar o diagrama de fluxo de sinais:

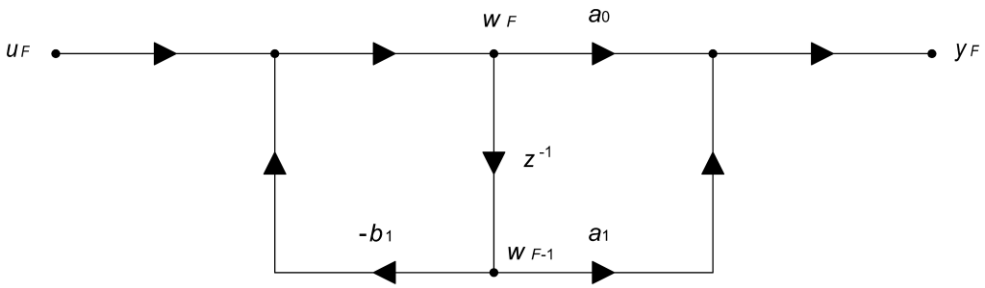


Figura 2.5.2 – Diagrama de blocos do Filtro PB de 1.ª Ordem com estrutura canónica direta

E as equações às diferenças são assim:

$$y_F=a_0 w_F+a_1 w_F z^{-1}$$
$$w_F=u_F-b_1 w_F z^{-1} \qquad 2.5.16$$

Na forma de funções de sistema:

$$y_F=\frac{\omega_c T_s}{2+\omega_c T_s} w_F+\frac{\omega_c T_s}{2+\omega_c T_s} w_F z^{-1}$$
$$w_F=u_F-\frac{-2+\omega_c T_s}{2+\omega_c T_s} w_F z^{-1} \qquad 2.5.17$$

2.5.1.2– Representação Canónica do Filtro PB de 2.ª Ordem

Partindo da equação 2.3.28 obtida pelo método de integração trapezoidal:

$$F_{2\,T}(z)=G_0\frac{\omega_c^2 T_s^2 z^2+2\omega_c^2 T_s^2 z+\omega_c^2 T_s^2}{\left(4+4\zeta\omega_c T_s+\omega_c^2 T_s^2\right)z^2+\left(-8+2\omega_c^2 T_s^2\right)z+4-4\zeta\omega_c T_s+\omega_c^2 T_s^2}$$

$$F_{2\,T}(z)=G_0\frac{\frac{\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2}+\frac{2\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2}z^{-1}+\frac{\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2}z^2}{1+\left(\frac{-8+2\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2}\right)z^{-1}+\left(\frac{4-4\zeta\omega_c T_s+\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2}\right)z^2} \qquad 2.5.18$$

Seja:

$$a_0=\frac{\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2} \qquad a_1=\frac{2\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2}$$
$$a_2=\frac{\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2} \qquad b_0=1$$
$$b_1=\frac{-8+2\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2} \qquad b_2=\frac{4-4\zeta\omega_c T_s+\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2}$$

$$F_{2\,T}(z)=G_0\frac{a_0+a_1 z^{-1}+a_2 z^{-2}}{1+b_1 z^{-1}+b_2 z^{-2}} \qquad 2.5.19$$

Introduzindo a variável $W(z)$ e desenvolvendo, para um ganho G_0 unitário, tem-se:

$$Y(z)=W(z)\left(a_0+a_1 z^{-1}+a_2 z^{-2}\right)$$
$$W(z)=U(z)-W(z)\left(b_1 z^{-1}+b_2 z^{-2}\right) \qquad 2.5.20$$

Que no domínio temporal corresponde a:

$$y_{F2}=a_0 w_{F2}+a_1 w_{F2\text{-}1}+a_2 w_{F2\text{-}2}$$
$$w_{F2}=u_{F2}-b_1 w_{F2\text{-}1}-b_2 w_{F2\text{-}2} \qquad 2.5.21$$

E, desta forma, é possível esboçar o diagrama de fluxo de sinais:

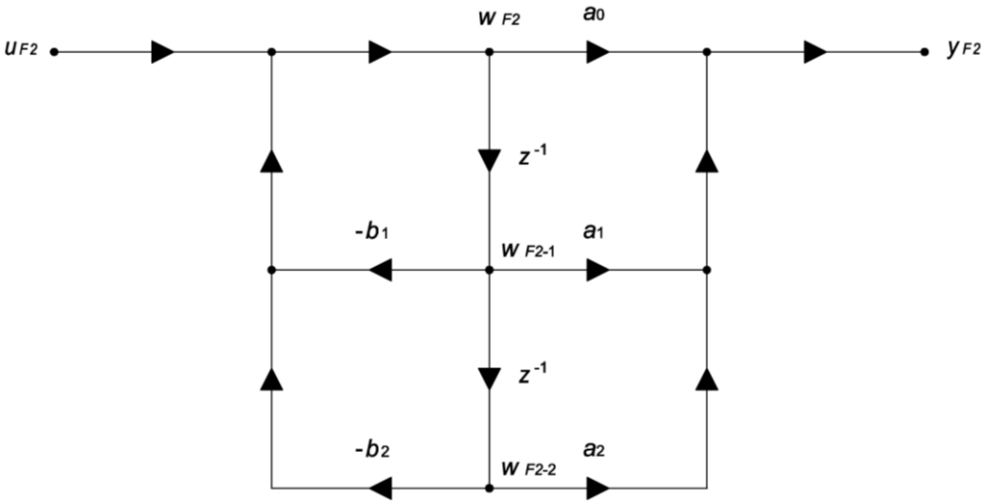


Figura 2.5.3 – Diagrama de blocos do Filtro PB de 2.ª Ordem com estrutura canónica direta

As equações às diferenças têm assim a forma:

$$y_{F2}=a_0 w_{F2}+a_1 w_{F2} z^{-1}+a_2 w_{F2} z^{-2}$$
$$w_{F2}=u_{F2}-b_1 w_{F2} z^{-1}-b_2 w_{F2} z^{-2} \qquad 2.5.22$$

Na forma de funções de sistema, e substituindo as constantes *a* e *b*:

$$y_{F2}=\frac{\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2} w_{F2}+\frac{2\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2} w_{F2} z^{-1}+\frac{\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2} w_{F2} z^{-2}$$
$$w_{F2}=u_{F2}-\frac{-8+2\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2} w_{F2} z^{-1}-\frac{4-4\zeta\omega_c T_s+\omega_c^2 T_s^2}{4+4\zeta\omega_c T_s+\omega_c^2 T_s^2} w_{F2} z^{-2} \qquad 2.5.23$$

2.5.1.3– Representação Canónica do Controlador PI

Com base na equação 2.3.40 obtida pelo método de integração trapezoidal:

$$C_{PI\,T}(z)=\frac{(2K_P+K_I T_s)z-2K_P+K_I T_s}{2z-2}$$
$$C_{PI\,T}(z)=\frac{K_P+\frac{1}{2}K_I T_s+\left(-K_P+\frac{1}{2}K_I T_s\right)z^{-1}}{1-z^{-1}} \qquad 2.5.24$$

Seja:

$$a_0=K_p+\frac{1}{2}K_I T_s \qquad a_1=-K_P+\frac{1}{2}K_I T_s \qquad b_0=1 \qquad b_1=-1$$

$$C_{PI\,T}(z)=\frac{a_0+a_1 z^{-1}}{1+b_1 z^{-1}} \tag{2.5.25}$$

Introduzindo a variável $W(z)$ e, desenvolvendo, tem-se:

$$\begin{aligned} Y(z)&=W(z)\left(a_0+a_1 z^{-1}\right) \\ W(z)&=U(z)-W(z)b_1 z^{-1} \end{aligned} \tag{2.5.26}$$

Que no domínio temporal corresponde a:

$$\begin{aligned} y_{PI}&=a_0 w_{PI}+a_1 w_{PI\text{-}1} \\ w_{PI}&=u_{PI}-b_1 w_{PI\text{-}1} \end{aligned} \tag{2.5.27}$$

E assim sendo é possível esboçar o diagrama de fluxo de sinais:

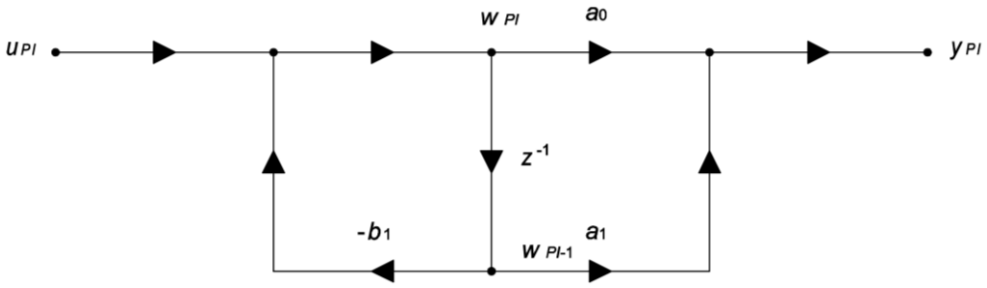


Figura 2.5.4 – Diagrama de blocos do Controlador PI com estrutura canónica direta

As equações às diferenças são então:

$$\begin{aligned} y_{PI}&=a_0 w_{PI}+a_1 w_{PI} z^{-1} \\ w_{PI}&=u_{PI}-b_1 w_{PI} z^{-1} \end{aligned} \tag{2.5.28}$$

E na forma de funções de transferência:

$$\begin{aligned} y_{PI}&=\left(K_p+\frac{1}{2}K_I T_s\right)w_{PI}+\left(-K_P+\frac{1}{2}K_I T_s\right)w_{PI} z^{-1} \\ w_{PI}&=u_{PI}+w_{PI} z^{-1} \end{aligned} \tag{2.5.29}$$

2.5.1.4– Representação Canónica do Controlador PID

Tendo em conta a equação 2.3.44 obtida pelo método de integração trapezoidal:

$$C_{PID\,T}(z)=\frac{\left(4K_D+2T_sK_P+T_s^2K_I\right)z^2+\left(-8K_D+2T_s^2K_I\right)z+4K_D-2T_sK_P+T_s^2K_I}{2T_sz^2-2T_s}$$

$$C_{PID\,T}(z)=\frac{\frac{4K_D+2T_sK_P+T_s^2K_I}{2T_s}+\frac{-8K_D+2T_s^2K_I}{2T_s}z^{-1}+\frac{4K_D-2T_sK_P+T_s^2K_I}{2T_s}z^{-2}}{1-1z^{-2}} \qquad 2.5.30$$

Sendo:

$$a_0=\frac{4K_D+2T_sK_P+T_s^2K_I}{2T_s} \qquad a_1=\frac{-8K_D+2T_s^2K_I}{2T_s}$$

$$a_2=\frac{4K_D-2T_sK_P+T_s^2K_I}{2T_s} \qquad b_0=1$$

$$b_1=0 \qquad b_2=-1$$

$$C_{PID\,T}(z)=\frac{a_0+a_1z^{-1}+a_2z^{-2}}{1-b_2z^{-2}} \qquad 2.5.31$$

Introduzindo a variável $W(z)$ e desenvolvendo tem-se:

$$\begin{aligned} Y(z)&=W(z)\left(a_0+a_1z^{-1}+a_2z^{-2}\right)\\ W(z)&=U(z)-W(z)b_2z^{-2}\end{aligned} \qquad 2.5.32$$

Que no domínio temporal corresponde a:

$$\begin{aligned} y_{PID}&=a_0w_{PID}+a_1w_{PID-1}+a_2w_{PID-2}\\ w_{PID}&=u_{PID}-b_2w_{PID-2}\end{aligned} \qquad 2.5.33$$

E assim sendo é possível esboçar o diagrama de fluxo de sinais:

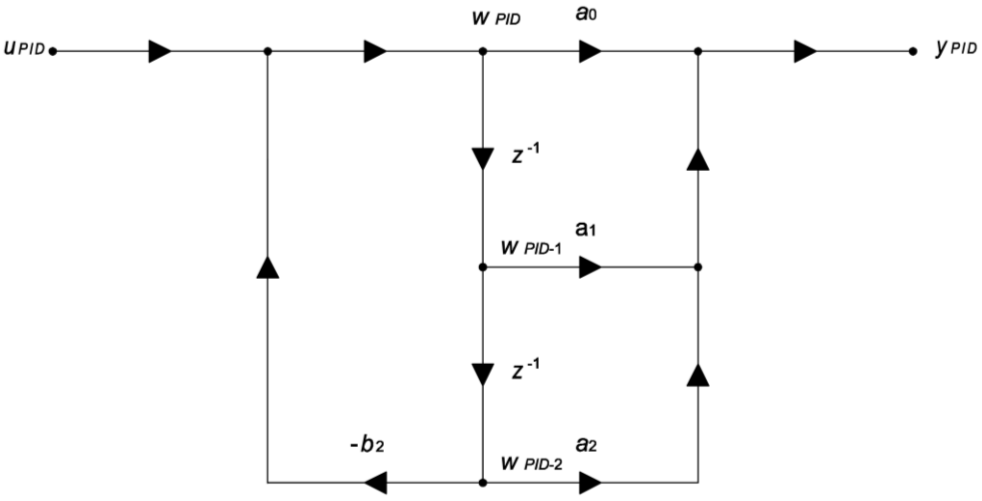


Figura 2.5.5 – Diagrama de blocos do Controlador PID com estrutura canónica direta

Equação às diferenças:

$$y_{PID}=a_0 w_{PID}+a_1 w_{PID} z^{-1}+a_2 w_{PID} z^{-2}$$

$$w_{PID}=u_{PID}-b_2 w_{PID} z^{-2} \qquad 2.5.34$$

$$y_{PID}=\frac{4K_D+2T_s K_P+T_s^2 K_I}{2T_s} w_{PID}+\frac{-8K_D+2T_s^2 K_I}{2T_s} w_{PID} z^{-1}+\frac{4K_D-2T_s K_P+T_s^2 K_I}{2T_s} w_{PID} z^{-2}$$

$$w_{PID}=u_{PID}+w_{PID} z^{-2} \qquad 2.5.35$$

2.5.2– Estrutura Não Canónica Direta

A partir da equação 2.5.3 pode-se escrever:

$$Y(z)+Y(z)\left(1+\sum_{j=1}^{n} b_j z^{-j}\right)=U(z)\sum_{j=1}^{n} a_j z^{-j}$$

Considerando b_0=1, pode dizer-se que:

$$1+\sum_{j=1}^{n} b_j z^{-j}=\sum_{j=0}^{n} b_j z^{-j}$$

E assim:

$$Y(z)=U(z)\sum_{j=1}^{n} a_j z^{-j}-Y(z)\sum_{j=0}^{n} b_j z^{-j} \qquad 2.5.36$$

Passando para o domínio do tempo tem-se:

$$y_k=\sum_{j=1}^{n} a_j u_{k-j}-\sum_{j=0}^{n} b_j y_{k-j} \qquad 2.5.37$$

Que pode ser representado em diagrama de fluxo de sinal da seguinte forma:

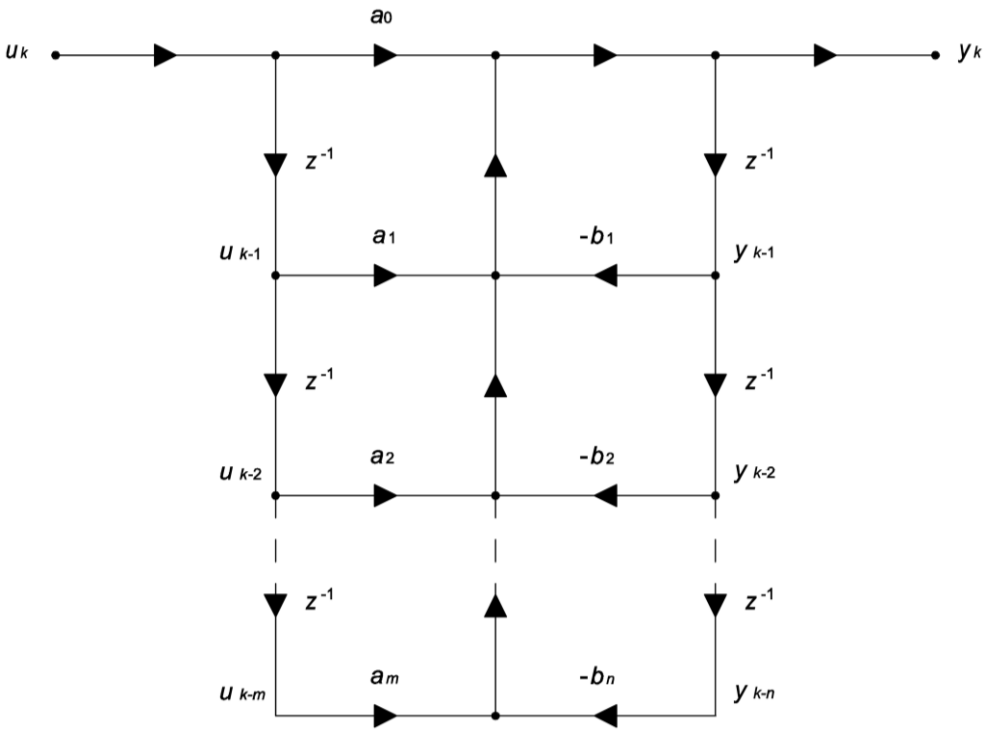


Figura 2.5.6 – Diagrama de blocos genérico de um sistema com estrutura não canónica direta

A equação às diferenças toma a forma:

$$y_k=u_k\left(a_0+a_1z^{-1}+a_2z^{-2}+\ldots+a_nz^{-n}\right)-y_k\left(b_1z^{-1}+b_2z^{-2}+\ldots+b_nz^{-n}\right) \quad 2.5.38$$

2.5.2.1– Representação Não Canónica do Filtro PB de 1.ª Ordem

Partindo da equação 2.3.13, obtida pelo método de integração trapezoidal:

$$F_T(z)=G_0\frac{\frac{\omega_cT_s}{2+\omega_cT_s}+\frac{\omega_cT_s}{2+\omega_cT_s}z^{-1}}{1+\left(\frac{-2+\omega_cT_s}{2+\omega_cT_s}\right)z^{-1}}$$

$$F_T(z)=\frac{Y(z)}{U(z)}=G_0\frac{a_0+a_1z^{-1}}{1+b_1z^{-1}} \quad 2.5.39$$

Desenvolvendo tem-se, para G_0 unitário:

$$Y(z)\left(1+b_1z^{-1}\right)=U(z)\left(a_0+a_1z^{-1}\right)$$

$$Y(z)=U(z)\left(a_0+a_1z^{-1}\right)-Y(z)b_1z^{-1} \quad 2.5.40$$

E passando para o domínio do tempo:

$$y_F=a_0u_F+a_1u_{F-1}-b_1y_{F-1} \quad 2.5.41$$

Que pode ser representado em diagrama de fluxo de sinal da seguinte forma:

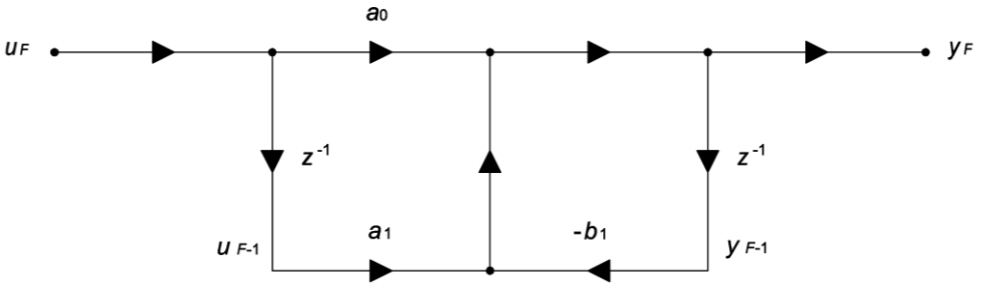


Figura 2.5.7 – Diagrama de blocos do Filtro PB de 1.ª ordem com estrutura não canónica direta

Em que a equação às diferenças toma a forma:

$$y_F=u_F\left(a_0+a_1z^{-1}\right)-y_Fb_1z^{-1} \quad 2.5.42$$

Onde os coeficientes são idênticos aos da representação canónica direta.

2.5.2.2 – Representação Não Canónica do Filtro PB de 2.ª Ordem

Considerando a equação 2.5.18, obtida pelo método de integração trapezoidal:

$$F_{2\,T}(z)=G_0\frac{\frac{\omega_c^2T_s^2}{4+4\zeta\omega_cT_s+\omega_c^2T_s^2}+\frac{2\omega_c^2T_s^2}{4+4\zeta\omega_cT_s+\omega_c^2T_s^2}z^{-1}+\frac{\omega_c^2T_s^2}{4+4\zeta\omega_cT_s+\omega_c^2T_s^2}z^2}{1+\left(\frac{-8+2\omega_c^2T_s^2}{4+4\zeta\omega_cT_s+\omega_c^2T_s^2}\right)z^{-1}+\left(\frac{4-4\zeta\omega_cT_s+\omega_c^2T_s^2}{4+4\zeta\omega_cT_s+\omega_c^2T_s^2}\right)z^2}$$

$$F_{2\,T}(z)=\frac{Y(z)}{U(z)}=G_0\frac{a_0+a_1z^{-1}+a_2z^{-2}}{1+b_1z^{-1}+b_2z^{-2}} \qquad 2.5.43$$

Desenvolvendo tem-se, para G_0 unitário:

$$Y(z)\left(1+b_1z^{-1}+b_2z^{-2}\right)=U(z)\left(a_0+a_1z^{-1}+a_2z^{-2}\right)$$

$$Y(z)=U(z)\left(a_0+a_1z^{-1}+a_2z^{-2}\right)-Y(z)\left(b_1z^{-1}+b_2z^{-2}\right) \qquad 2.5.44$$

E passando para o domínio do tempo:

$$y_{F2}=a_0u_{F2}+a_1u_{F2\text{-}1}+a_2u_{F2\text{-}2}-b_1y_{F2\text{-}1}-b_2y_{F2\text{-}2} \qquad 2.5.45$$

Que pode ser representado em diagrama de fluxo de sinal da seguinte forma:

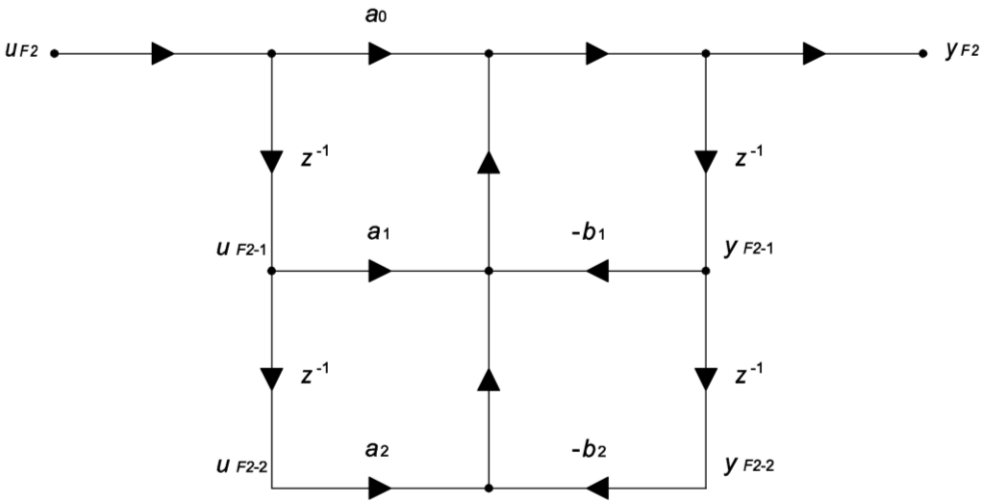


Figura 2.5.8 – Diagrama de blocos do Filtro PB de 2.ª ordem com estrutura não canónica direta

Em que a equação às diferenças toma a forma:

$$y_{F2}=u_{F2}\left(a_0+a_1z^{-1}+a_2z^{-2}\right)-y_{F2}\left(b_1z^{-1}+b_2z^{-2}\right) \qquad 2.5.46$$

2.5.2.3– Representação Não Canónica do Controlador PI

Tendo em conta a equação 2.5.24, obtida pelo método de integração trapezoidal:

$$C_{PI\,T}(z)=\frac{K_P+\frac{1}{2}K_I T_s+\left(-K_P+\frac{1}{2}K_I T_s\right)z^{-1}}{1-z^{-1}}$$

$$C_{PI\,T}(z)=\frac{Y(z)}{U(z)}=\frac{a_0+a_1 z^{-1}}{1+b_1 z^{-1}} \qquad 2.5.47$$

Desenvolvendo tem-se:

$$Y(z)\left(1+b_1 z^{-1}\right)=U(z)\left(a_0+a_1 z^{-1}\right)$$

$$Y(z)=U(z)\left(a_0+a_1 z^{-1}\right)-Y(z)b_1 z^{-1} \qquad 2.5.48$$

E passando para o domínio do tempo:

$$y_{PI}=a_0 u_{PI}+a_1 u_{PI\text{-}1}-b_1 y_{PI\text{-}1} \qquad 2.5.49$$

Que pode ser representado em diagrama de fluxo de sinal da seguinte forma:

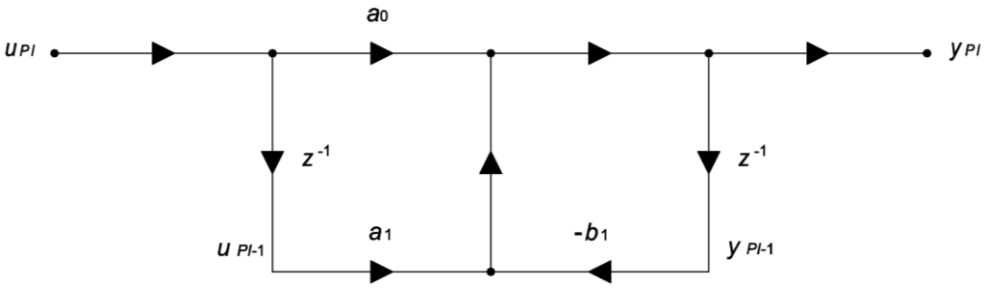


Figura 2.5.9 – Diagrama de blocos do Controlador PI com estrutura não canónica direta

Em que a equação às diferenças toma a forma:

$$y_{PI}=u_{PI}\left(a_0+a_1 z^{-1}\right)-y_{PI}b_1 z^{-1} \qquad 2.5.50$$

Onde os coeficientes são idênticos aos da representação canónica direta.

2.5.2.4 – Representação Não Canónica do Controlador PID

Com base na equação 2.5.30, obtida pelo método de integração trapezoidal:

$$C_{PID\,T}(z)=\frac{\frac{4K_D+2T_s K_P+T_s^2 K_I}{2T_s}+\frac{-8K_D+2T_s^2 K_I}{2T_s}z^{-1}+\frac{4K_D-2T_s K_P+T_s^2 K_I}{2T_s}z^{-2}}{1-1z^{-2}}$$

$$C_{PID\,T}(z)=\frac{Y(z)}{U(z)}=\frac{a_0+a_1 z^{-1}+a_2 z^{-2}}{1-b_2 z^{-2}} \qquad 2.5.51$$

Desenvolvendo tem-se:

$$Y(z)\left(1+b_2z^{-2}\right)=U(z)\left(a_0+a_1z^{-1}+a_2z^{-2}\right)$$

$$Y(z)=U(z)\left(a_0+a_1z^{-1}+a_2z^{-2}\right)-Y(z)b_2z^{-2} \qquad 2.5.52$$

E passando para o domínio do tempo:

$$y_{PID}=a_0u_{PID}+a_1u_{PID-1}+a_2u_{PID-2}-b_2y_{PID-2} \qquad 2.5.53$$

Que pode ser representado em diagrama de fluxo de sinal da seguinte forma:

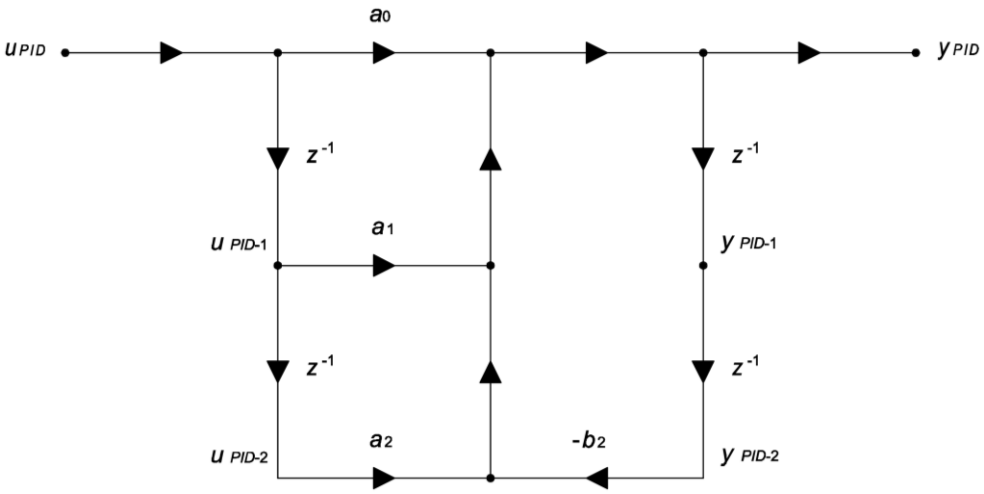


Figura 2.5.10 – Diagrama de blocos do Controlador PID com estrutura não canónica direta

Em que a equação às diferenças toma a forma:

$$y_{PID}=u_{PID}\left(a_0+a_1z^{-1}+a_2z^{-2}\right)-y_{PID}b_2z^{-2} \qquad 2.5.54$$

Onde os coeficientes são idênticos aos da representação canónica direta.

Capítulo III: Simulação em *Simulink*

Os sistemas estudados neste trabalho podem também ser representados no *software Matlab-Simulink* na forma de diagramas de blocos. O objetivo deste capítulo é a comparação das respostas obtidas pelos diferentes métodos de apresentação dos sistemas tanto no domínio contínuo como discreto. Por consequente, irão ser comparadas as seguintes formas de representação:

- Função de transferência *F*(*s*);
- Função de sistema *F*(*z*);
- Estrutura canónica direta;

O código utilizado em *Matlab* para a obtenção das respostas apresentadas neste ponto encontra-se no Anexo 5.

3.1 – Respostas do Filtro PB de 1.ª Ordem

A simulação efetuada tem a seguinte estrutura:

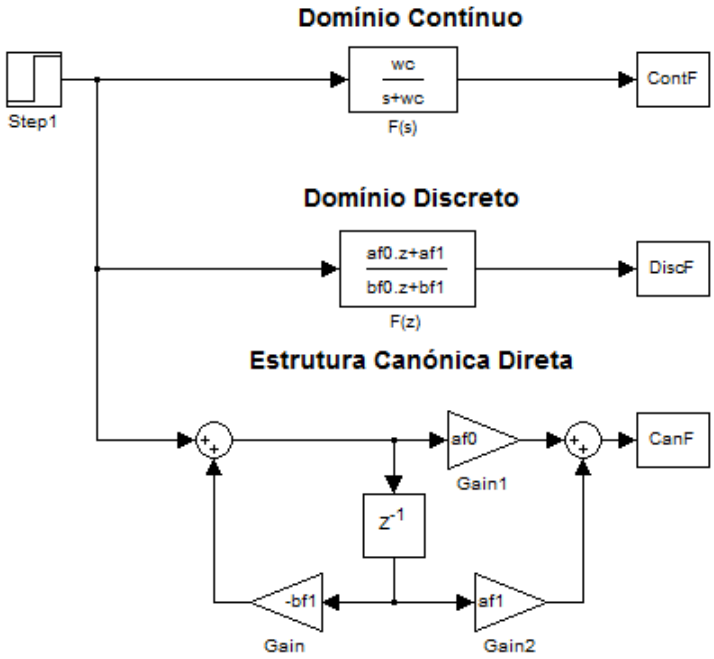


Figura 3.1.1 – Diagrama de blocos *Simulink* para simulação do filtro PB 1.ª nos diferentes domínios

A estrutura utilizada para a simulação foi a estrutura canónica direta, pois permite pois permite a utilização de um menor número de blocos, garantindo o desempenho.

As respostas tomam a seguinte forma, para T_s=10 µs, e G_0=1 V/V:

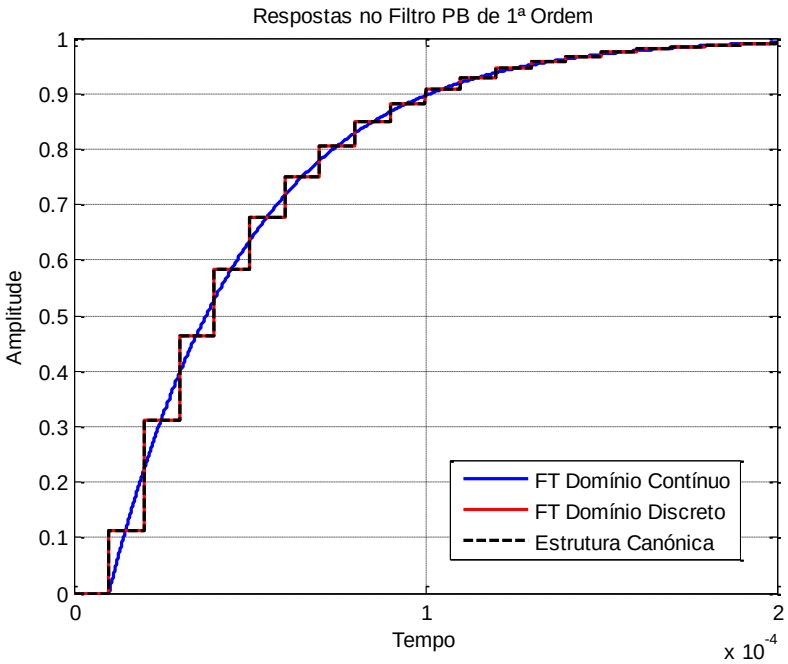


Figura 3.1.2 – Resposta do Filtro PB 1.ª a uma entrada do tipo degrau pela simulação em *Simulink*

Através de várias simulações em simultâneo é possível observar o comportamento do filtro quando sujeito a uma entrada sinusoidal com frequências de 1 kHz, 2 kHz e 4 kHz, para T_s=10 µs, f_c=4 kHz e G_0=1 V/V:

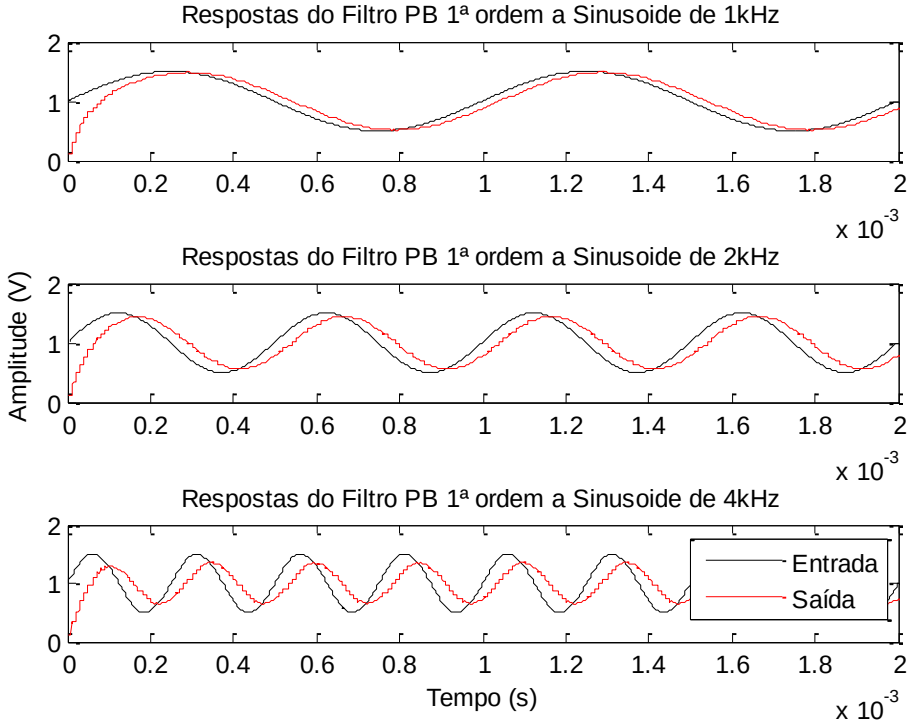


Figura 3.1.3 – Respostas do Filtro PB 1.ª simulado em *Simulink* a entradas do tipo sinusoidal com variação de frequência para f_c=4 kHz

3.2– Respostas do Filtro PB de 2.ª Ordem

A simulação efetuada tem a seguinte estrutura:

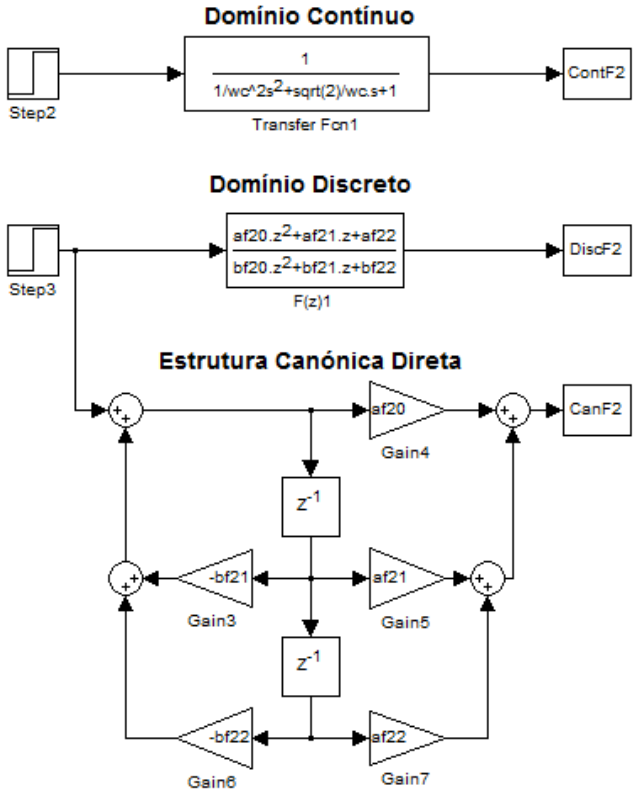


Figura 3.2.1 – Diagrama de blocos *Simulink* para simulação do filtro PB 2.ª nos diferentes domínios

A resposta temporal do filtro de 2.ª ordem com característica *Butterworth* para uma f_c=4.000 Hz, G_0=1 V/V e T_s=10 µs está apresentada na Figura 3.2.2.

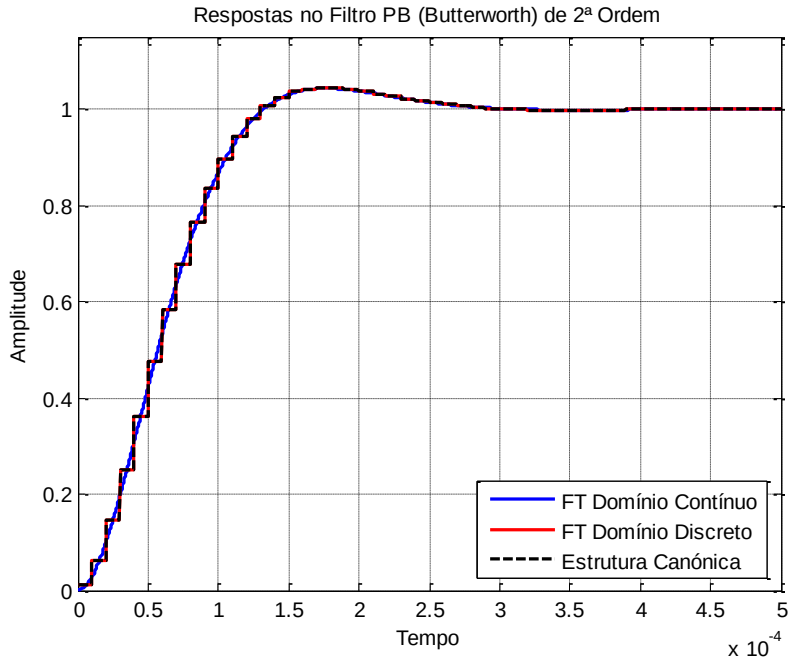


Figura 3.2.2 – Resposta do Filtro PB 2.ª a uma entrada do tipo degrau pela simulação em *Simulink*

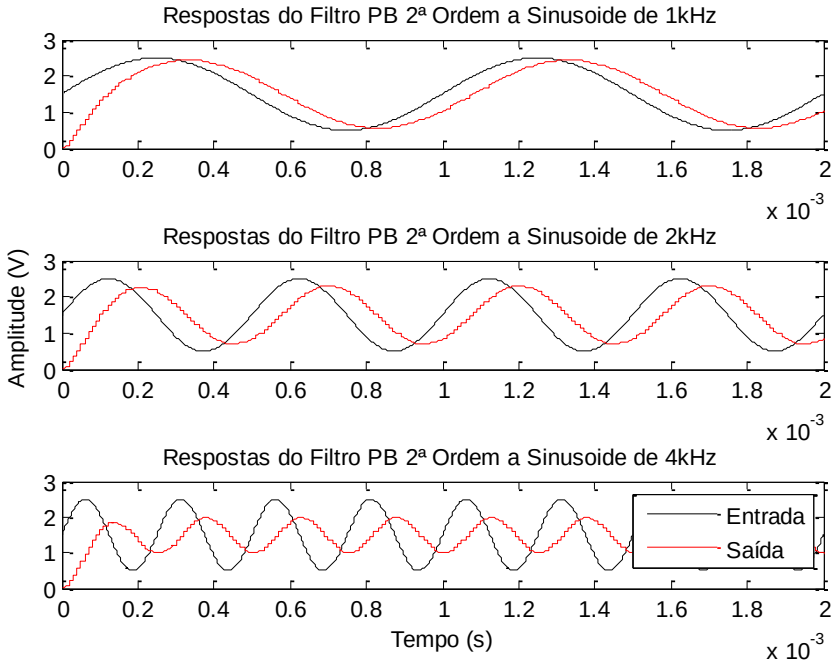


Figura 3.2.3 – Respostas do Filtro PB 2.ª simulado em *Simulink* a entradas do tipo sinusoidal com variação de frequência para f_c=4 kHz

Nesta resposta é possível de observar que, em comparação com o filtro de 1.ª ordem, existe um atraso um pouco maior no filtro de 2.ª ordem. No entanto a amplitude a 4 kHz é menor, ou seja, um filtro de 2.ª ordem face a um de 1.ª permite uma maior atenuação do sinal à frequência de corte.

3.3 – Respostas do Controlador PI

Para a simulação, e posterior ensaio prático, simula-se uma entrada com um valor constante para que seja observada a resposta do controlador PI. Tal como apresentado na Figura 2.2.6, é de esperar que a resposta do PI seja uma reta com declive positivo até que seja atingido o valor máximo à saída do mesmo.

Para o controlador PI não é feita a simulação com o bloco de equações no domínio contínuo e discreto, pois, tendo um número de polos e zeros igual, estes blocos devolvem um erro de cálculo. Na Figura 3.3.1 está apresentado o diagrama de blocos da simulação do controlador PID com uma entrada com uma amplitude de 0,25 V e T_S=10 µs. As constantes têm os valores:

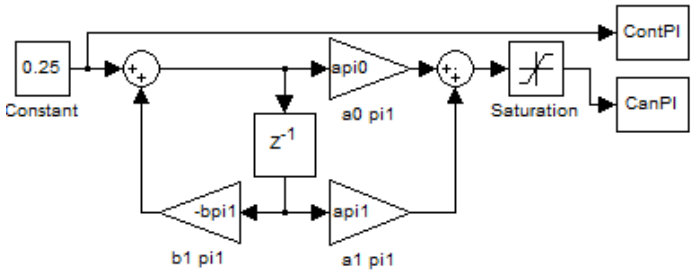


Figura 3.3.1 – Diagrama de blocos do controlador PI no *Simulink*

É utilizado um bloco denominado *saturation* que tem a função de limitador do sinal à saída, limitando entre o intervalo [0; 3] V.

A resposta do sistema, para as constantes K_P=30x10^{-6} e K_I=1,5, está apresentada na Figura 3.3.2.

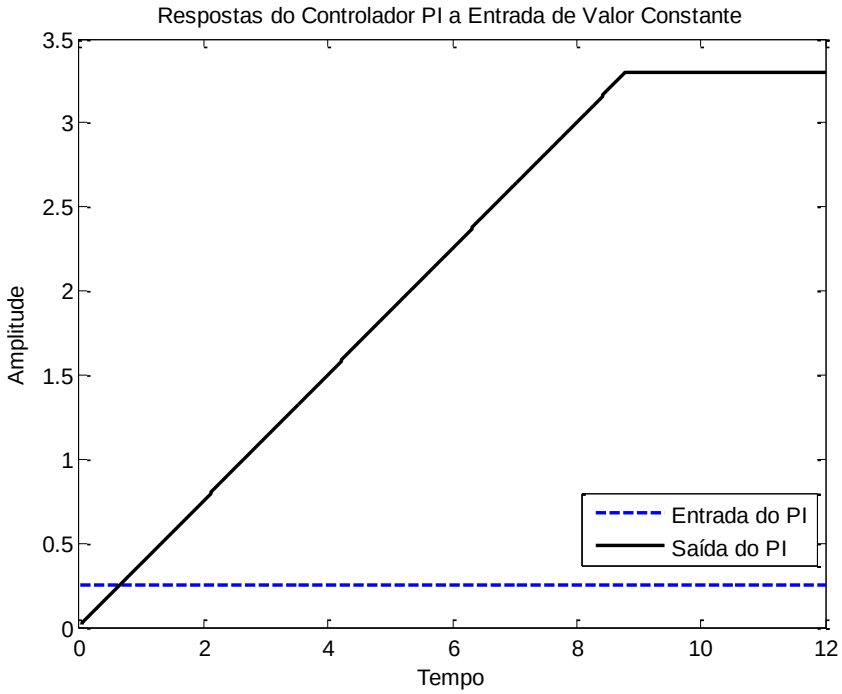


Figura 3.3.2 – Resposta do controlador PI a uma entrada constante de amplitude 0,25

Pela figura da reposta do controlador PI, observa-se que ao fim de um determinado tempo a saída do controlador atinge o seu valor máximo que, nesta simulação, está definido como 3,3.

A implementação prática do controlador PID é, em termos de programação, feita da mesma forma que os filtros e controlador PI. No entanto o seu ensaio implica um sistema a controlar para que exista anulamento de polos, como dito no ponto 2.3.4. Seria possível por exemplo definir a constante K_D como um valor baixo o suficiente para que a resposta fosse idêntica à do controlador PI. No entanto, como neste trabalho já é apresentada a resposta do mesmo, não é interessante do ponto de vista pedagógico fazê-lo. Assim sendo, o controlador PID não é simulado nem ensaiado na prática.

A Figura 3.3.3 apresenta o diagrama de blocos que seria utilizado em *Simulink* para a simulação do controlador PID, com entrada – IN e saída – OUT), com estrutura não canónica direta.

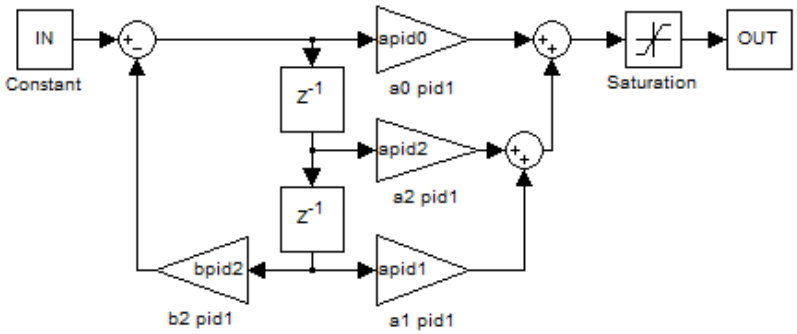


Figura 3.3.3 – Diagrama de blocos do controlador PI no *Simulink*

Capítulo IV: Implementação no *dsPIC*

4.1 – Microcontrolador *dsPIC*

4.1.1– *Hardware e Software*

Para a aplicação prática de um filtro ou controlador é possível escolher entre uma grande variedade de MCUs. Neste trabalho, optou-se por utilizar um MCU da *Microchip*. Para a implementação dos sistemas em causa, os requisitos de um MCU são:

- 1 entrada analógica com conversor A/D;
- 1 saída analógica com conversão D/A;

A família de MCUs da *Microchip* é denominada PIC, e dentro desta existe a possibilidade de escolher o MCU que melhor se adapta à aplicação. Dentro dos requisitos impostos, o que mais limita as possibilidades de escolha é o módulo conversor D/A, que existe apenas em alguns MCUs e muitas vezes com aplicação apenas em áudio. Desta forma, o PIC escolhido para a implementação dos filtros e controladores em estudo é o dsPIC33FJ16GS502 que tem como principais características:

- Até 50 MIPS (*Millions Instructions per Second*)
- Controlador de Sinal Digital a 16 bits;
- 16 kbyte Flash com conversor A/D de alta velocidade;
- 2 kbytes de memória RAM
- 8 entradas para conversão A/D;
- 1 saída da conversão D/A;
- Módulo ADC com 10 bits de resolução, dois conversores de aproximações sucessivas e seis circuitos de *Sample-and-Hold* (S&H);
- Comparador de alta velocidade com módulo de conversão D/A integrado;

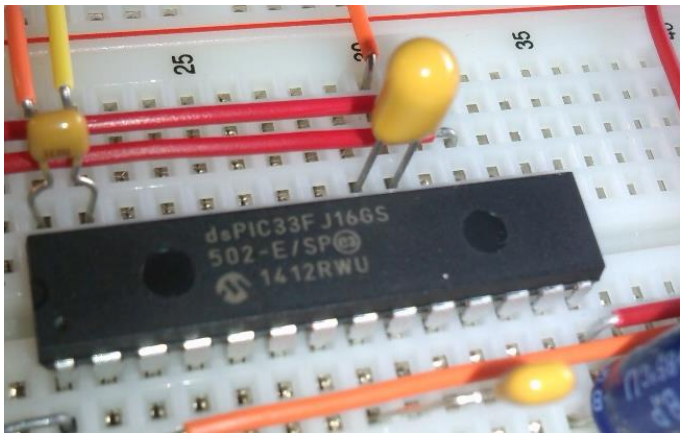


Figura 4.1.1 – Microcontrolador dsPIC33FJ16GP502 da Microchip, utilizado na aplicação prática

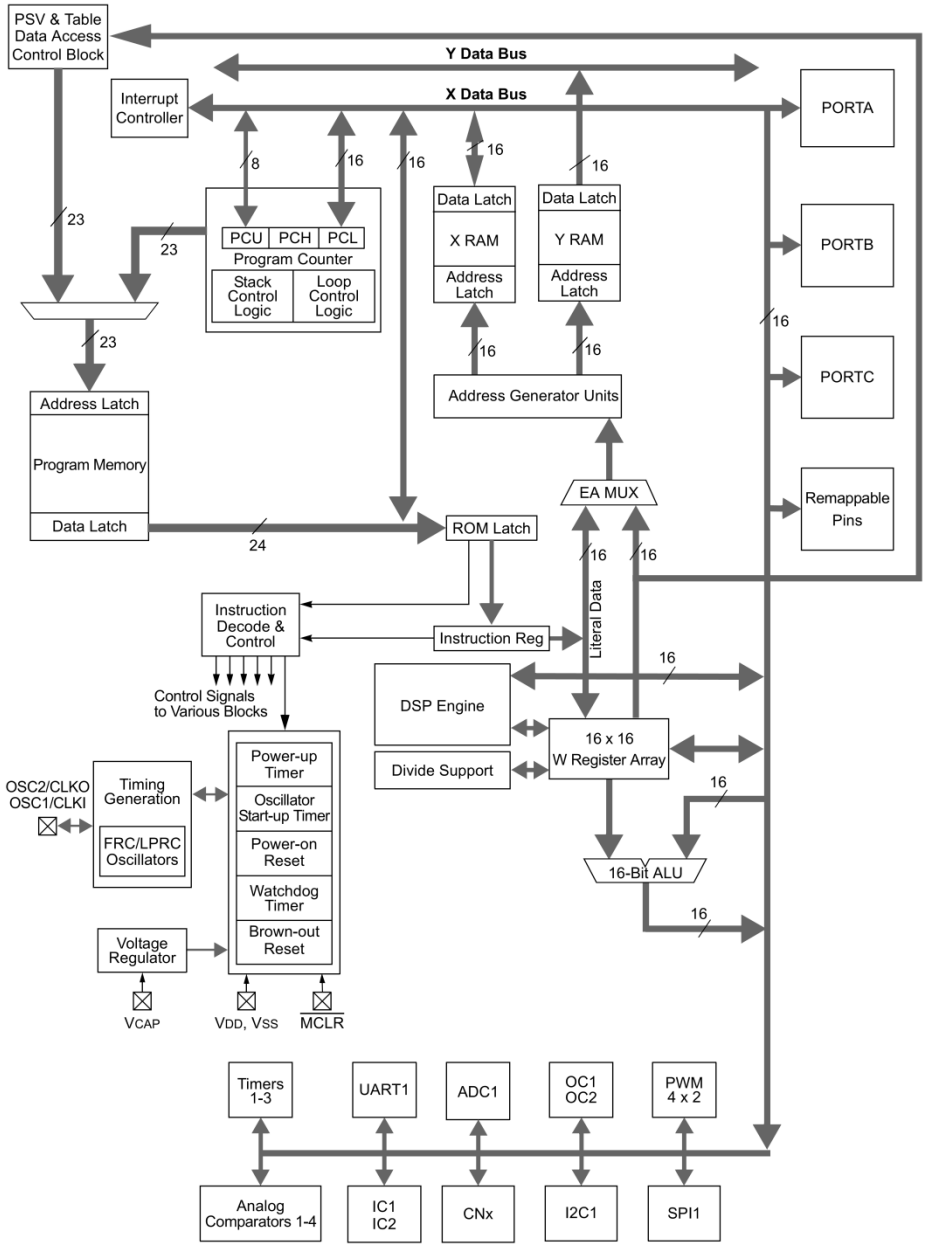


Figura 4.1.2 – Estrutura interna do dsPIC33FJ16GS502 [10]

A Figura 4.1.2 apresenta o esquema da estrutura interna do núcleo e módulos periféricos no MCU utilizado neste trabalho.

Para que o MCU funcione corretamente é necessário um circuito básico constituído por componentes como resistências e condensadores que pode ser montado numa *breadboard.* O circuito básico de funcionamento pode ser encontrado no *datasheet* do fabricante. A Figura 4.1.3 apresenta o esquema do circuito utilizado na implementação prática.

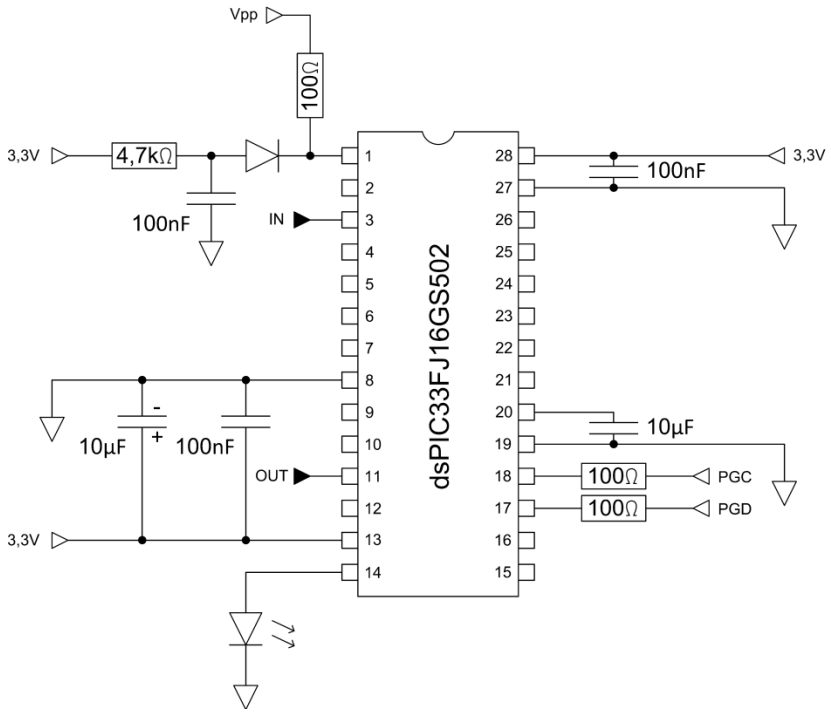


Figura 4.1.3 – Esquema do Circuito implementado para o uso do MCU como filtro ou controlador

Do *datasheet* do MCU são retirados os valores máximos permitidos na entrada e saída analógica do conversor D/A. Uma entrada analógica deste *dsPIC* consegue processar normalmente um valor em tensão entre 0 e 3,3 V. Este valor, depois de convertido para digital irá corresponder a um valor entre 0 e 1023. Após processado este valor, por métodos explicados em pontos anteriores, é posto um novo valor de 0 a 1023 no conversor D/A que, por sua vez, tem uma gama de valores entre 0 e $\frac{3,3\,V}{2}$, ou seja, entre 0 e 1,65 V. Deste modo diz-se que, à saída, a amplitude máxima do sinal é de metade do valor à entrada. A Figura 4.1.4 representa esquematicamente a estrutura da aplicação prática.

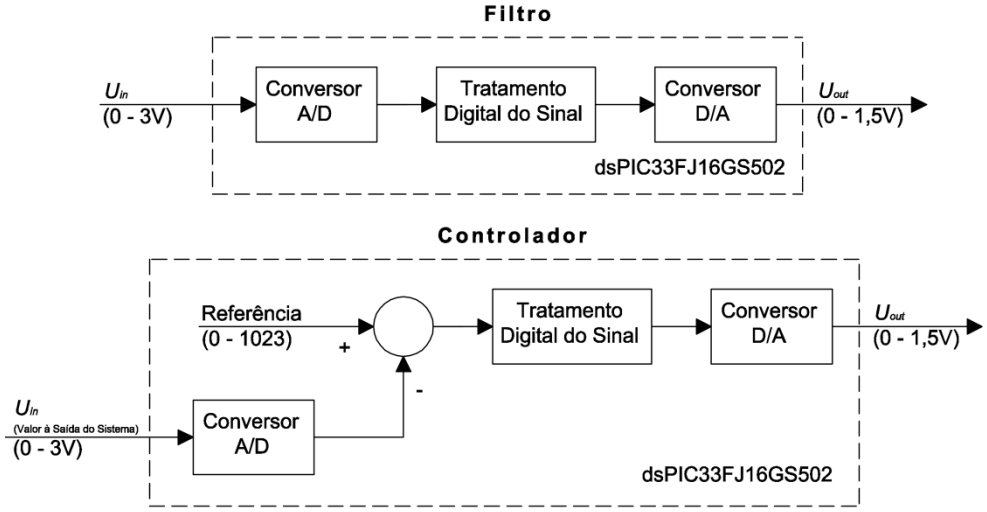


Figura 4.1.4 – Representação esquemática da aplicação prática do filtro ou controlador

Por forma a preservar o espectro original do sinal amostrado, é necessário dimensionar o filtro de *anti-aliasing*. Este filtro é utilizado quando o MCU é utilizado como filtro, pois importa garantir que não existe distorção do sinal amostrado. Nos controladores PI e PID o filtro de *anti-aliasing* não é necessário, pois não há a importância de manter o espetro original do sinal

Optou-se por utilizar para a filtragem *anti-aliasing* um filtro de 4ª ordem, pois possui uma relação entre a frequência de corte e a complexidade da implementação adequada à aplicação. Assim, pela Tabela 1.3.2, o filtro terá uma frequência de corte de 8.394 Hz.

Para o dimensionamento dos componentes do filtro utilizou-se o software de dimensionamento de filtros da *Microchip*, *FilterLab*. As Figura 4.1.5 e Figura 4.1.6 apresentam o resultado da utilização do programa.

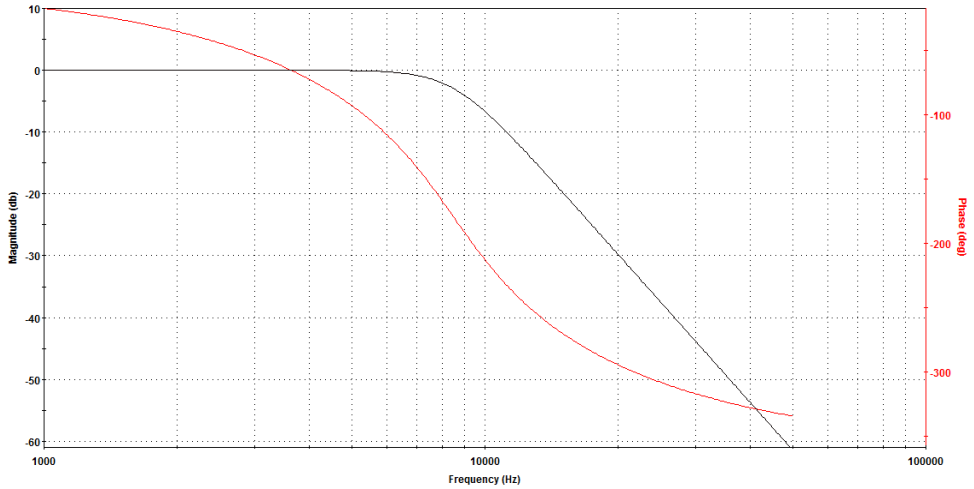


Figura 4.1.5 – Resposta em frequência do filtro *anti-aliasing* dimensionado (obtido no *FilterLab*)

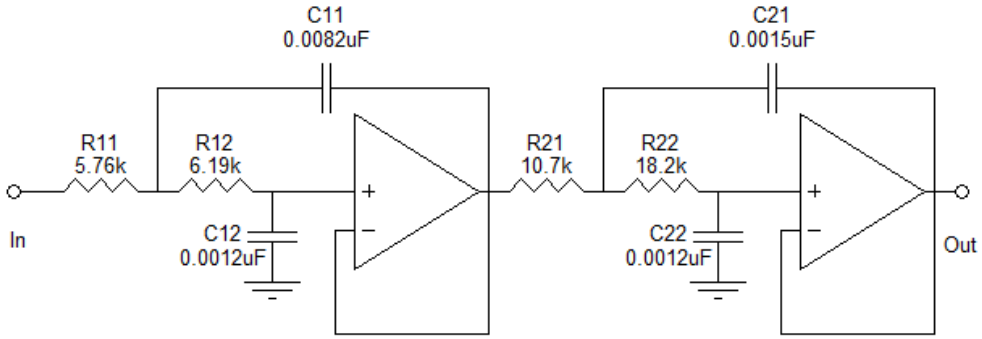


Figura 4.1.6 – Circuito do filtro *anti-aliasing* dimensionado (obtido no *FilterLab*)

A ação de retenção de amostra realiza-se com um retentor de ordem zero (ZOH – *Zero Order Hold*) presente nos conversores D/A. À saída destes torna-se necessário efetuar uma filtragem do tipo passa-baixo para limitar o espectro de saída. Em consequência desta necessidade, é utilizado um filtro de reconstrução à saída do conversor que permite desta forma eliminar as altas frequências provenientes da conversão D/A criando um efeito de alisamento no sinal reduzindo a consequência da função de retenção.

O filtro de reconstrução possui a mesma ordem e atenuação do filtro *anti-aliasing* pelo que o dimensionamento é idêntico assim como filtro em si.

4.1.2– Estrutura do Programa

Para efetuar a programação do MCU é necessário um *hardware* que permita inserir no dispositivo o código traduzido para uma linguagem hexadecimal. Este dispositivo tem o nome de *PicKit 3*. A técnica utilizada pela *Microchip* para a programação deste tipo de equipamentos é denominada ICSP (*In-Circuit Serial Programming*). Esta comunicação utiliza apenas dois módulos de entradas/saídas (à exceção da alimentação, massa e V_{PP}) tornando-se menos intrusiva para o dispositivo e aplicação, permitindo programar com os dispositivos já nas placas ou já instalados, de forma rápida e fácil, recorrendo à utilização de 5 ligações no total. O ICSP trás vantagens na redução de tempo, na realização de *updates* aos sistemas já instalados, na redução de custos de fabrico (consequência da velocidade de programação), acertos finais nas aplicações (como criação de ID's, ajustes de variáveis, etc.), personalização das aplicações após instaladas, entre outras.

Figura 4.1.7 – Programador PicKIt 3, utilizado para programar o MCU

Para a programação e teste do *dsPIC* foi utilizado o *software* da *Microchip MPLAB X*, que permite programar em linguagem C, configurando o *dsPIC* corretamente por forma a obter o desempenho desejado. A implementação dos filtros e controladores em si é feita a partir das equações apresentadas nos capítulos anteriores utilizando a estrutura canónica direta, pois é a estrutura que permite obter um menor número de cálculos, simplificando o código e tornando-o mais rápido.

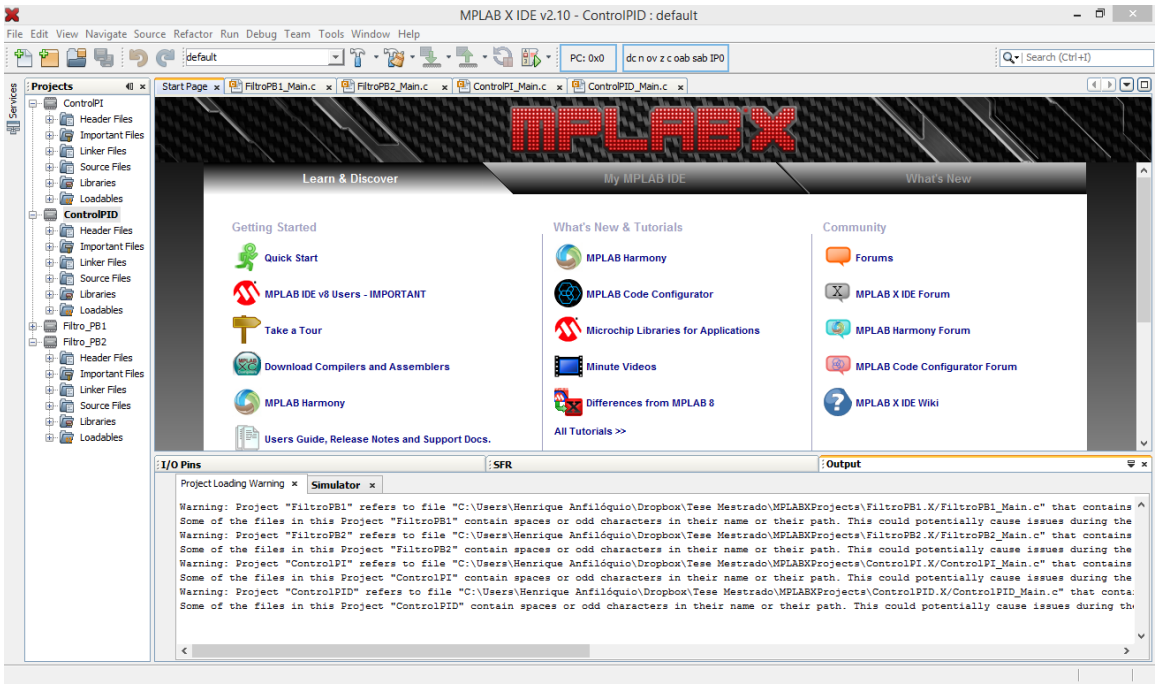


Figura 4.1.8 – Ambiente do software MPLAB X, utilizado para a construção do programa em C

Todo o programa é feito em linguagem C que é posteriormente convertida para hexadecimal por um compilador (neste caso o X16) para programar o *dsPIC*. A estrutura do programa é apresentada na Tabela 4.1.2.

Tabela 4.1.1 – Estrutura do programa em linguagem C

Configuração	Adição das bibliotecas a utilizar, configuração dos bits de configuração, configuração do oscilador;
Inicialização	Configuração de entradas e saídas a utilizar, inicialização e parametrização dos módulos ADC, DAC, temporizador e rotinas de interrupção;
Rotina de cálculo	Definição de variáveis, cálculo das constantes, aquisição de um valor no ADC à custa de uma interrupção, cálculos correspondentes ao filtro ou controlador, colocação de um valor no DAC;
Rotinas de Interrupção	Rotina de leitura do valor no módulo ADC, rotina de interrupção do temporizador e rotinas correspondentes a falhas ou avarias no MCU;

O objetivo deste programa é funcionar num *loop* infinito em que se extrai um valor do módulo ADC, é efetuado o cálculo do valor da saída em função dos valores das entradas e saídas obtidas anteriormente, é colocado o resultado no módulo DAC, guarda-se o valor da entrada e saída numa variável e repete-se infinitamente o mesmo processo.

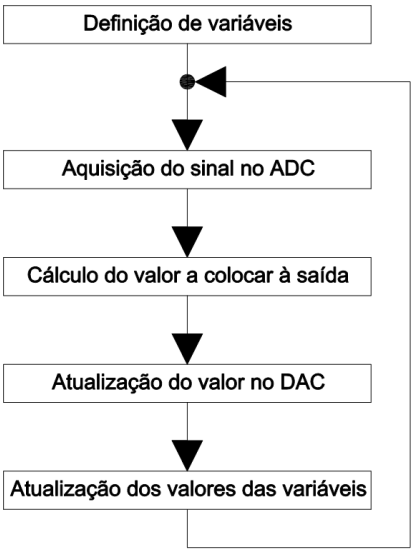


Figura 4.1.9 – Estrutura da rotina de cálculo

Para que o cálculo seja feito de forma mais rápida é necessário trabalhar com as variáveis no formato inteiro (*int*) e não no formato de vírgula flutuante (*float*). No entanto ao passar para o formato inteiro é perdida a informação das casas decimais devido aos arredondamentos feitos. Assim, para que não seja perdida informação, multiplicam-se as variáveis por um valor tal que as casas decimais passam para a parte inteira (multiplicar por 1.000.000 por exemplo). Consequentemente, é necessário dividir o valor final dos cálculos por esta constante de multiplicação.

Há, no entanto, que considerar o tamanho das variáveis. Se ao longo dos cálculos o resultado for um valor tal que uma palavra de 16 bits não é suficiente, no formato *int*, há a necessidade de utilizar o formado de inteiro longo, *long int*, que possui 32 bits. No caso de ainda existirem problemas com o tamanho da palavra, divide-se ao longo dos cálculos a variável por um valor suficiente (1.000 por exemplo) para diminuir o tamanho por forma a ter valores mais pequenos e multiplica-se, posteriormente, pelo mesmo valor para que os resultados não sejam alterados. No entanto, estes passos resultam na perca de informação existente em casas decimais, o que resulta posteriormente num erro associado ao valor final obtido. A Figura 4.1.10 apresenta o código utilizado para a definição das variáveis e cálculo das constantes.

```
float Ts=0.000010;                  // Tempo de Amostragem
float fc=4000;                      // Frequência de Corte
float wc=2*3.14*fc;                 // Frequência Angular de Corte
float a0_aux=((wc*Ts)/(2+wc*Ts))*10000;     // Cálculo de a0
float b1_aux=((wc*Ts-2)/(2+wc*Ts))*10000;   // Cálculo de a1
```

Figura 4.1.10 – Código C da definição das variáveis e cálculo das constantes para o Filtro PB 1

Para que o tempo de amostragem seja o menor possível e, consequentemente, a frequência máxima admitida no MCU seja a maior possível, é necessário ter uma rotina de cálculo bem elaborada. A Figura 4.1.9 apresenta a estrutura da rotina de cálculo.

A rotina principal do programa é feita com o seguinte código, para o filtro passa-baixo de primeira ordem:

```
while (1)                          // Loop Infinito
{
    PORTBbits.RB11 = 1;            // Sinal Auxiliar de Tempo de Amostragem

    while(ADCPC0bits.PEND0);       // Espera pela conversão ADC
    Uk = ADCBUF1;                  // Leitura da segunda conversão
    ADCPC0bits.SWTRG0 = 1;         // Iniciar nova conversão em AN0 e AN1

    Yk = (a0*Uk + a1*Uk1 - b1*Yk1)/10000;   // Atualizar Saída

    if (Yk>1023)                   // Limitador Superior
        Yk=1023;                   // Valor Máximo
    if (Yk<0)                      // Limitador Inferior
        Yk=0;                      // Valor Mínimo

    CMPDAC1bits.CMREF = Yk;        // Colocar Valor na Saída
                                   // (CMREF*(AVDD/2)/1024)volts
    Uk1=Uk;                        // Update Uk-1
    Yk1=Yk;                        // Update Yk1

    while (PORTBbits.RB11 == 1)    // Espera pela interrupção do Timer 1
    {}                             // Realizar Tempo de Amostragem Completo
}
```

Figura 4.1.11 – Código C da rotina principal do Filtro PB1 para implementação no PIC

Na atualização dos valores das variáveis, os valores de entrada no ADC ou de saída no DAC são guardados em variáveis correspondentes aos valores dos ciclos anteriores. Isto é, segundo a estrutura utilizada, canónica direta, é necessário para os cálculos que sejam guardados em variáveis os valores de U_k, U_{k-1}, U_{k-2}, Y_k, Y_{k-1}, Y_{k-2}. Estes valores correspondem aos valores de entrada e saída do sistema nos ciclos anteriores.

- U_k – valor da entrada no ciclo atual;
- U_{k-1} – valor da entrada no ciclo anterior;
- U_{k-2} – valor da entrada dois ciclos antes;
- Y_k – valor da saída atual;
- Y_{k-1} – valor da saída no ciclo anterior;
- Y_{k-2} – valor da saída dois ciclos antes;

Após o código em C++ ser convertido para hexadecimal, cada instrução demora um determinado tempo a ser executada. A definição de uma variável, a soma ou multiplicação de duas, etc. demoram tempos diferentes dependendo do número de instruções pertencentes a cada uma destas "tarefas". O tempo entre cada instrução pode ser maior ou menor conforme a frequência de processamento do oscilador (F_{osc}). Como tal, é importante para se ter um bom tempo de amostragem, que esta frequência seja a maior possível. Para isso parametriza-se o *dsPIC*, segundo o *datasheet* do mesmo, para que a frequência do oscilador seja a maior possível diminuído assim o tempo entre cada instrução.

O MCU permite a utilização de um cristal externo para que o relógio interno (*clock*) possua maior exatidão. Como nesta aplicação não é necessária tal exatidão, o uso desse cristal externo é escusado, recorrendo-se à utilização do oscilador (RC) interno. A frequência do oscilador primário (F_{in}) máxima conseguida com este oscilador é de 7,37 MHz e é a partir desta que se altera a F_{osc} através da multiplicação da frequência obtida com uma malha de captura de fase (*PLL – Phase Locked Loop*). Como a frequência de funcionamento do oscilador depende desta frequência F_{in} e das variáveis *Prescaler* e *Postscaler*, é possível parametrizar o *dsPIC* para um valor mais elevado desde que a velocidade de operação do dispositivo (F_{CY}) não exceda muito os 40 MHz, definidos pelo fabricante.

Com base nesta F_{osc} é calculado o valor a carregar na variável do temporizador (*PR1* do *Timer1*). Estes cálculos são apresentados na Tabela 4.1.2.

Tabela 4.1.2 – Parametrização do oscilador e Timer

Cálculo da Frequência do Oscilador		
F_{in}	7,37 Mhz	
Variável *M*	106	
Poscaler (*N*1)	2	
Prescaler (*N*2)	4	
F_{osc}	$F_{osc}=F_{in}.\frac{M}{N1.N2}$	97.625.500 Hz
F_{CY}	$F_{CY}=\frac{F_{osc}}{2}$	48.826.250 Hz
Cálculo da Valor a carregar no PR1		
$F_{Timer\ 1}$	$=F_{osc}$	97.625.500 Hz
Período entre instrução	$T_{inst}=\frac{1}{F_{osc}}$	10,24 ns
Prescaler *PS*	1	
Tempo desejado $T_{Timer\ 1}$	10µs	
***PR*1**	$PR1=\frac{T_{Timer\ 1}}{2.T_{inst}.PS}$	488

Neste MCU, quando o resultado de um dado cálculo é um valor fora do intervalo [0; 1023] é admitido 0 para o resultado. Por forma a evitar que o valor seja 0, quando o resultado do cálculo for superior a 1023, o valor de Y_k é imediatamente comparado duas vezes após o cálculo do mesmo, por forma a garantir que não vai ser colocado à saída um valor que não esteja compreendido no intervalo [0; 1023]. Estas duas comparações, por 0 e por 1023, fazem o papel de limitador e, no caso dos controladores PI e PID, de anti *Windup*, pois garantem que não é acumulado um valor crescente na variável Y_k, devido à ação integral do mesmo.

As rotinas de interrupção presentes no programa, conforme apresentado na Tabela 4.1.2, servem para definir as tarefas a realizar em caso das interrupções do temporizador auxiliar da amostragem (*Timer* 1) e do módulo ADC. Este temporizador tem como objetivo garantir um tempo de amostragem certo (neste caso de 10 µs), para que as diferentes implementações dos filtros e controladores possuam todas elas a mesma frequência de amostragem (100 kHz).

Existem ainda as rotinas correspondentes a erros no programa ou no próprio *dsPIC*. Estas rotinas de interrupção fazem com que, no caso da ocorrência de um erro, é feito um *Reset* ao *dsPIC*. As interrupções associadas a erros podem ser:

- Falha no oscilador (*oscillator fail*);
- Erro de endereço (*address error*);
- Erro no apontador (*stack error*);
- Erro matemático ou de cálculos (*math error*);

No Pin 14 do MCU é ligado um LED que permite perceber se o sistema está em funcionamento. Este LED acende e apaga uma vez por cada duas amostras, permitindo observar metade do valor da frequência de amostragem.

O Anexo 6 apresenta o código em C utilizado para a implementação do Filtro Passa-Baixo de 1.ª ordem no *dsPIC* utilizado.

Na utilização como filtro, o sistema de aquisição possui uma entrada e uma saída apenas. Quando utilizado como controlador, é possível utilizar uma outra entrada do MCU para definir a referência do controlador. Outra forma passa por utilizar uma referência interna necessitando apenas de uma entrada e uma saída. No entanto, na eventualidade de se querer alterar a referência do sistema, é necessário reprogramar o MCU. A Figura 1.1.1 apresenta esquematicamente o uso da referência interna num sistema em cadeia fechada, tal como feito na implementação prática.

4.2 – Respostas dos Sistemas Implementados no *dsPIC*

Utilizando uma fonte geradora de sinais é possível realizar a simulação dos filtros de 1.ª e 2.ª ordem. Para isso basta criar uma onda sinusoidal com uma tensão entre o intervalo [0; 3,3V] e variar a sua frequência observando a diminuição da amplitude à medida que se aproxima da frequência de corte, inserida na programação. Na simulação do controlador PI é utilizada uma fonte de alimentação, com um sinal constante, e avalia-se a resposta do controlador conforme simulado no ponto 3.3 deste trabalho. Como explicado no ponto anterior, a tensão à saída do MCU será no máximo de metade do valor à entrada, pelo que o ganho utilizado no osciloscópio no canal que mede a saída (500 mV) será metade do ganho do canal que mede a entrada (1 V).

4.2.1– Resposta do Filtro PB de 1.ª Ordem (*dsPIC*)

Programando o MCU para a função de Filtro Passa-Baixo de 2.ª ordem, com um tempo de amostragem de 10 µs e f_c=4 kHz, valores idênticos aos utilizados na simulação, obtêm-se as respostas apresentadas nas figuras seguintes, para as frequências de 1 kHz, 2 kHz e 4 kHz. O sinal à entrada do MCU, em tensão, tem amplitude de 1 V com *offset* DC em 1,5 V.

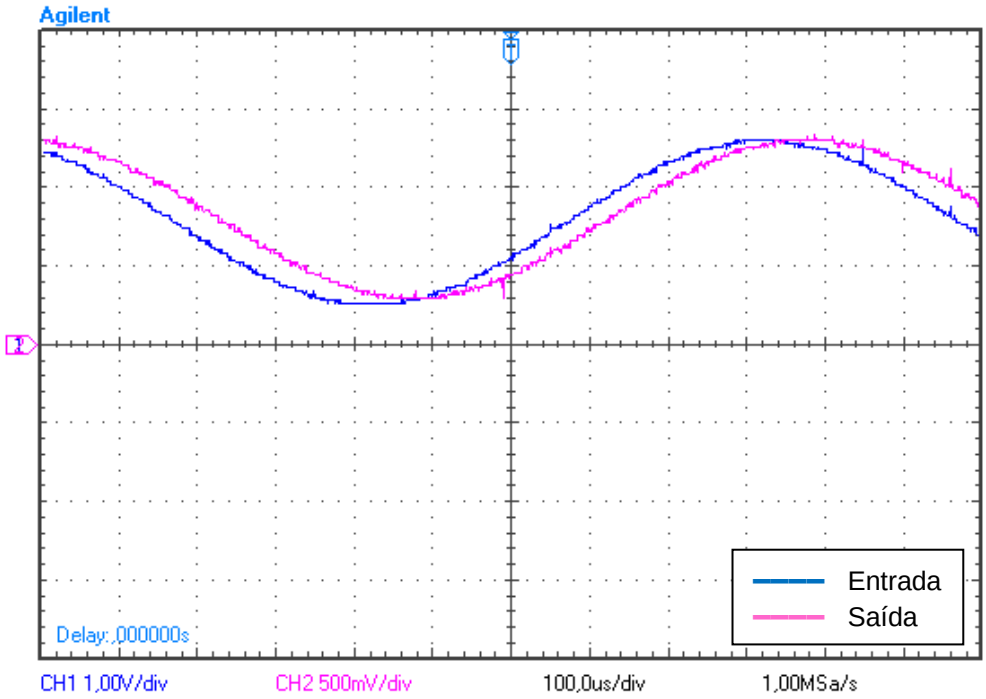


Figura 4.2.1 – Resposta de um Filtro PB de 1.ª Ordem implementado em MCU com f_c=4 kHz, sinal com 1,5 V de *offset*, amplitude 1 V e frequência 1 kHz

É possível observar, na Figura 4.2.1, que o sinal à saída possui um valor com amplitude de aproximadamente metade da entrada. Este facto é explicado no ponto 4.1.1 deste trabalho. Observa-se também um atraso na resposta, e que, não estando próximo da frequência de corte o sinal não possui atenuação.

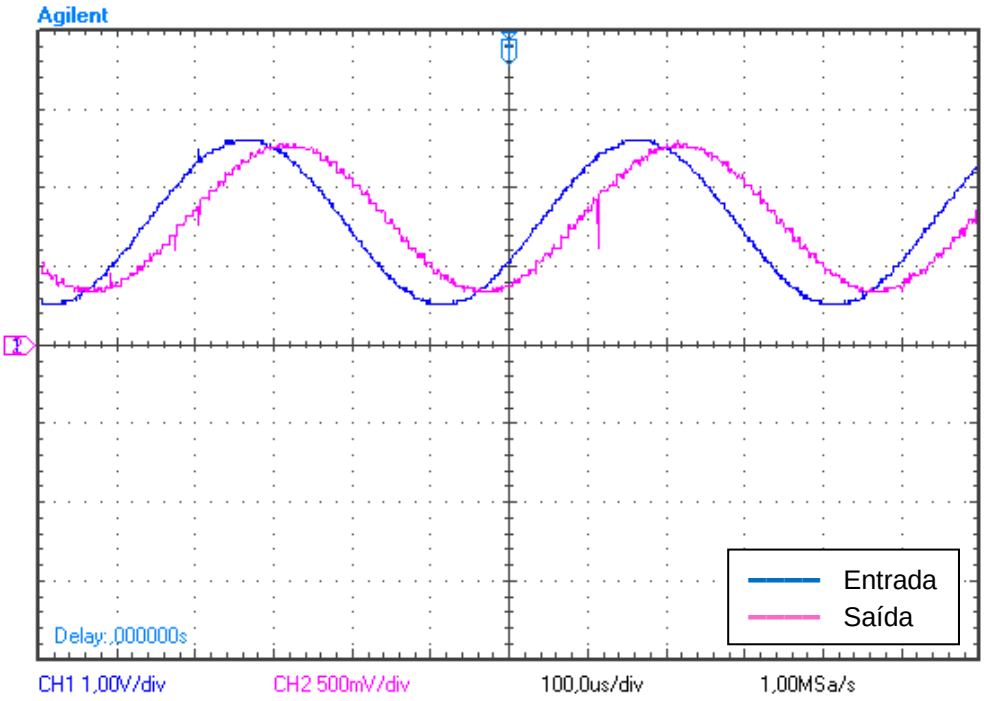


Figura 4.2.2 – Resposta de um Filtro PB de 1.ª Ordem implementado em MCU com f_c=4 kHz, sinal com 1,5 V de *offset*, amplitude 1 V e frequência 2 kHz

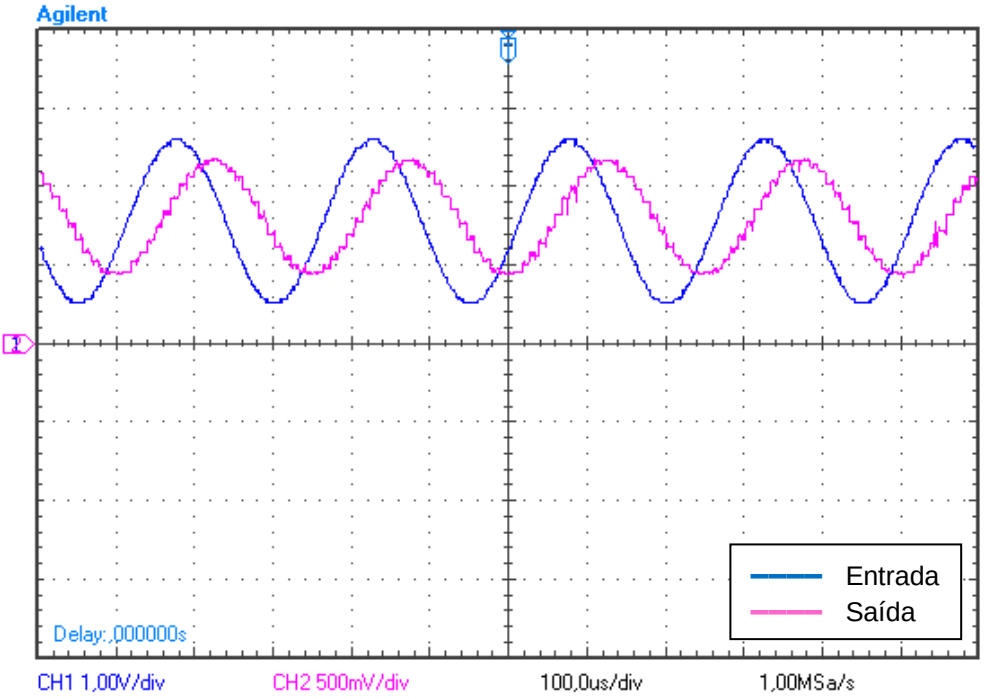
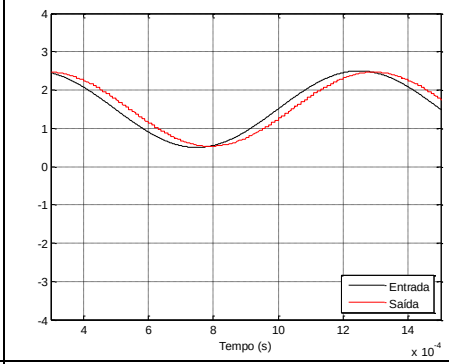
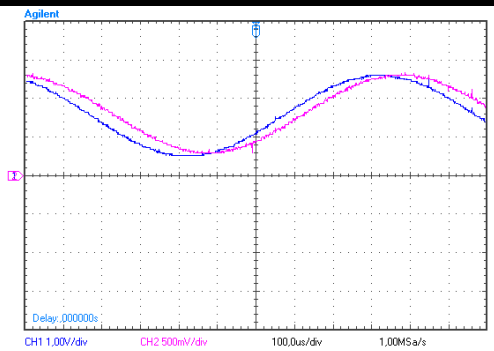
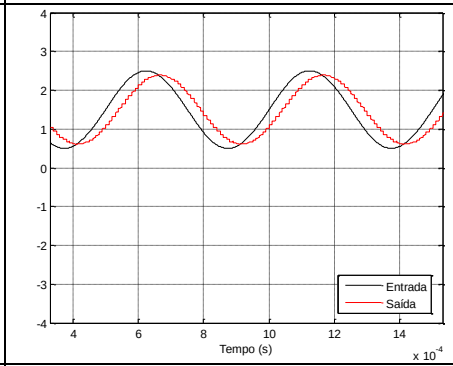
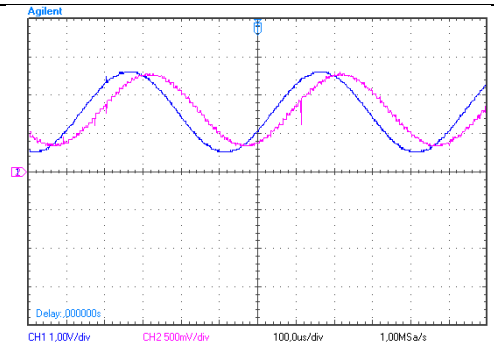
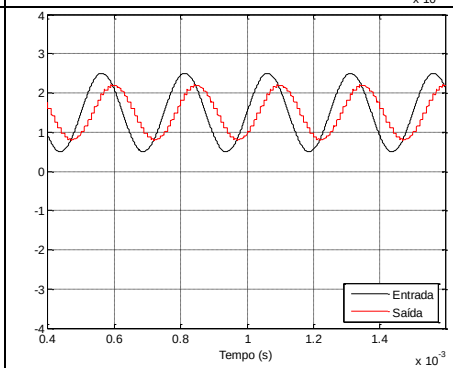
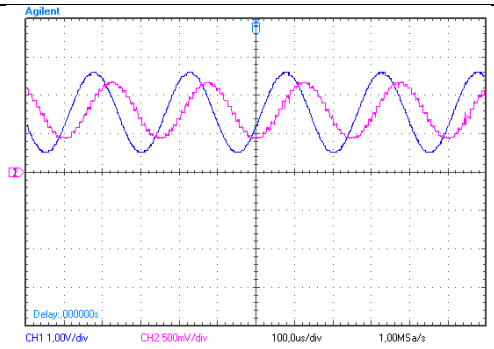


Figura 4.2.3 – Resposta de um Filtro PB de 1.ª Ordem implementado em MCU com f_c=4 kHz, sinal com 1,5 V de *offset*, amplitude 1 V e frequência 4 kHz

Na Figura 4.2.3 observa-se que, à frequência de corte, existe já uma atenuação do sinal, tendo assim o sinal à saída do filtro uma amplitude menor do que à entrada (na proporção 3,3 V - 1,65 V). Para uma melhor comparação dos resultados foi criada a Tabela 4.2.1, onde se encontram lado a lado os resultados da simulação e do ensaio experimental.

Tabela 4.2.1 – Comparação das respostas obtidas na simulação e no ensaio de um Filtro PB de 1.ª Ordem implementado em MCU com f_c=4 kHz, sinal com 1,5 V de *offset*, amplitude 1 V

Freq.	Simulação	Experimental
1 kHz		
2 kHz		
4 kHz		

Observa-se assim que os resultados obtidos em simulação, pelo *software Simulink*, são idênticos aos resultados obtidos no ensaio prático, à exceção da amplitude do sinal à saída do MCU, em que o valor é sempre metade do valor obtido em simulação, consequência da gama de valores de tensão à saída do MCU ter metade da amplitude da gama dos valores à entrada.

4.2.2– Resposta do Filtro PB de 2.ª Ordem (*Butterworth*) (*dsPIC*)

Quando programando para a função de Filtro Passa-Baixo de 2.ª ordem, com um tempo de amostragem de 10 µs e f_c=4 kHz, valores idênticos aos utilizados na simulação, obtêm-se à saída do MCU as respostas apresentadas nas figuras seguintes, para as frequências de 1 kHz, 2 kHz e 4 kHz. O sinal à entrada do MCU, em tensão, tem amplitude de 1 V com *offset* DC em 1,5 V.

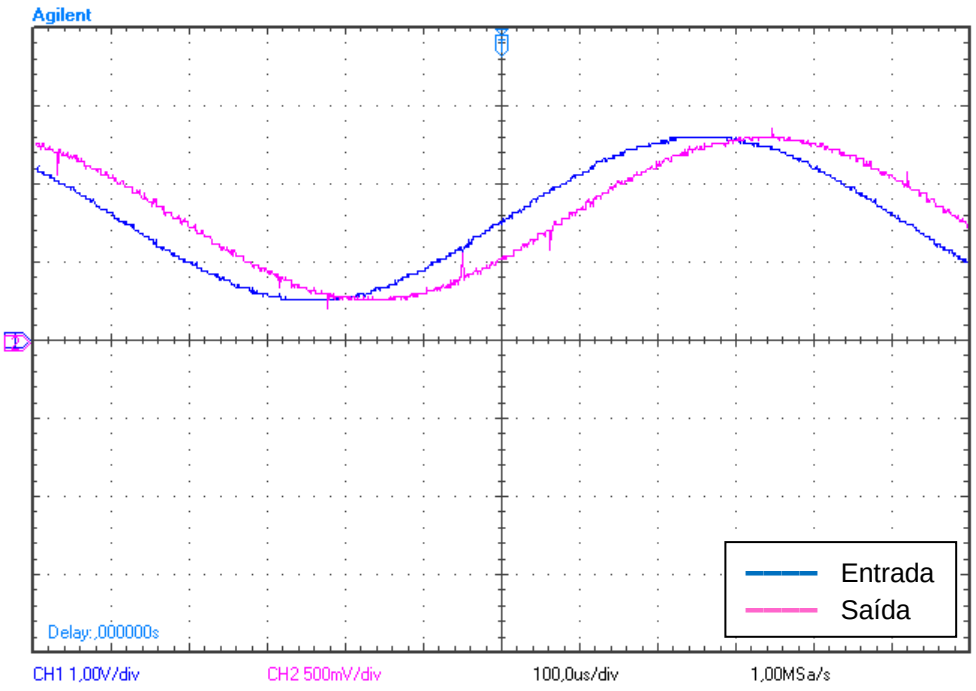


Figura 4.2.4 – Resposta de um Filtro PB de 2.ª Ordem implementado em MCU com f_c=4 kHz, sinal com 1,5 V de *offset*, amplitude 1 V e frequência 1 kHz

Comparando a Figura 4.2.1 com a Figura 4.2.4, observa-se no filtro de 2.ª ordem um maior atraso na resposta.

Na Figura 4.2.5, é possível observar uma ligeira atenuação do sinal, a uma frequência de metade da frequência de corte, atenuação esta que, no filtro de 1.ª ordem, a esta frequência, era menor.

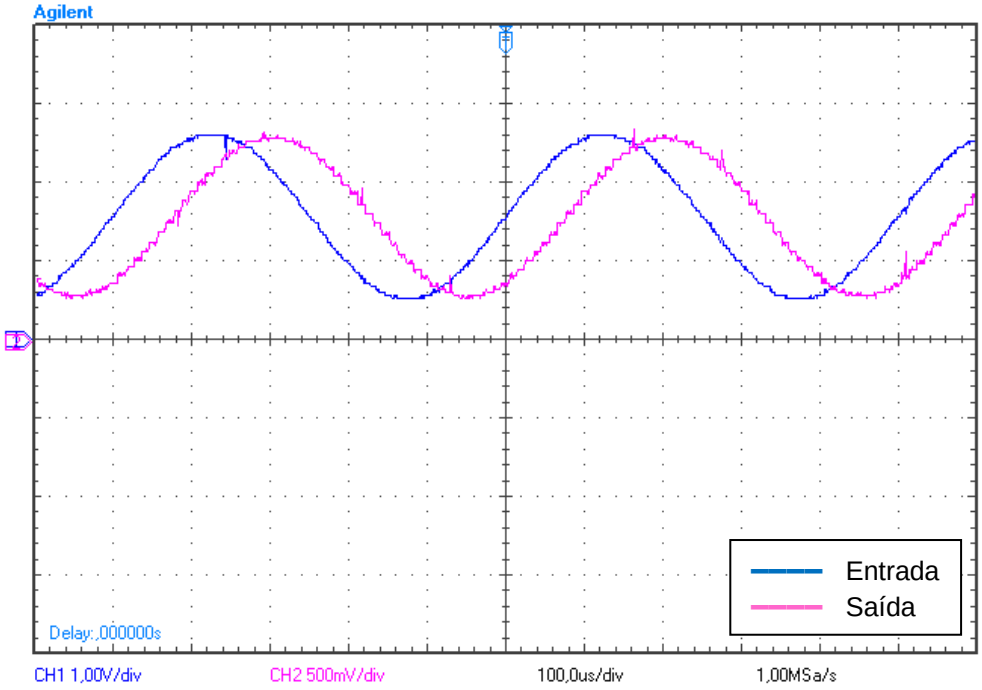


Figura 4.2.5 – Resposta de um Filtro PB de 2.ª Ordem implementado em MCU com f_c=2 kHz, sinal com 1,5 V de *offset*, amplitude 1 V e frequência 2 kHz

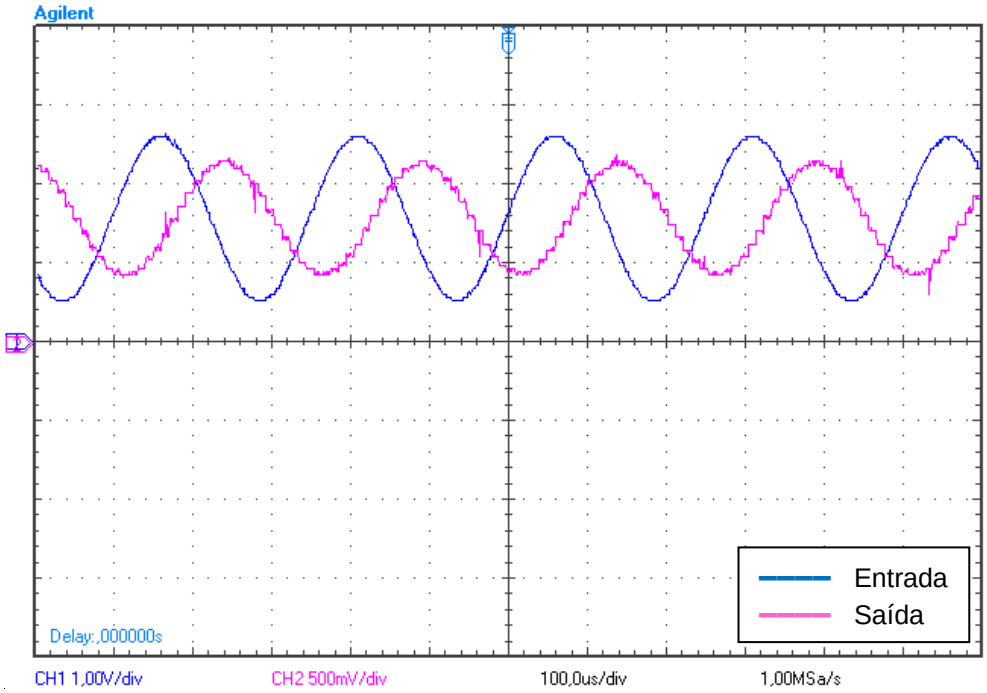
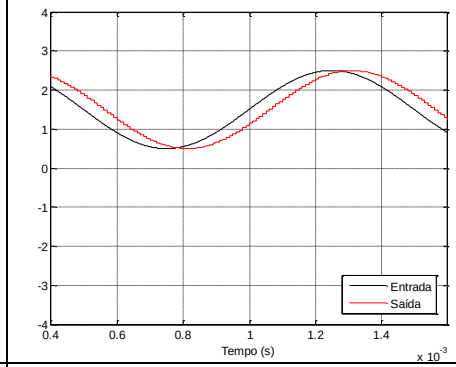
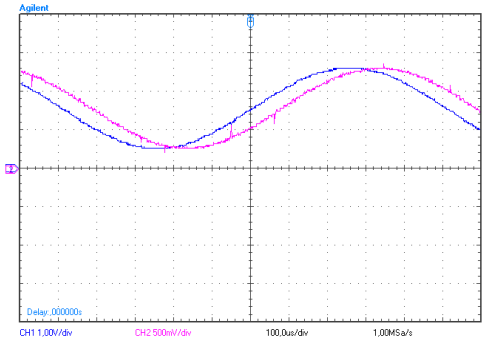
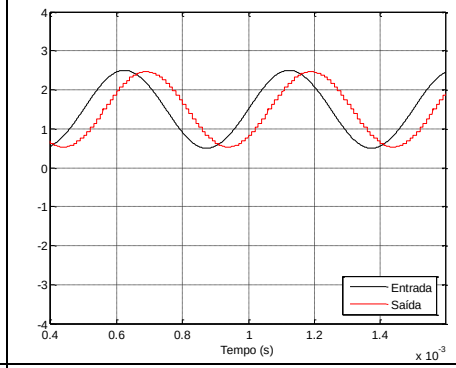
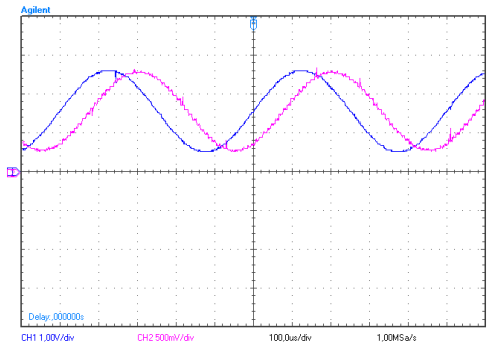
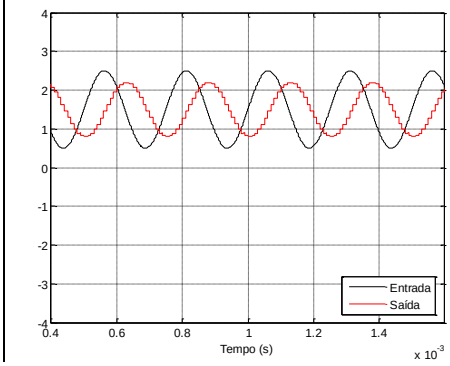
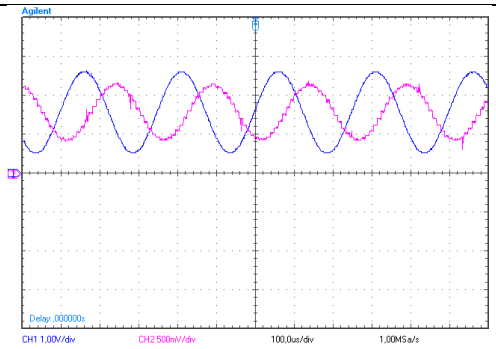


Figura 4.2.6 – Resposta de um Filtro PB de 2.ª Ordem implementado em MCU com f_c=4 kHz, sinal com 1,5 V de *offset*, amplitude 1 V e frequência 4 kHz

Para uma melhor comparação dos resultados foi criada a Tabela 4.2.2, onde se encontram lado a lado os resultados da simulação e do ensaio experimental.

Tabela 4.2.2 – Comparação das respostas obtidas na simulação e no ensaio de um Filtro PB de 2.ª Ordem implementado em MCU com f_c=4 kHz, sinal com 1,5 V de *offset*, amplitude 1 V

Freq.	Simulação	Experimental
1 kHz		
2 kHz		
4 kHz		

Os resultados obtidos em simulação, pelo *software Simulink*, são idênticos aos resultados obtidos no ensaio prático. Novamente, a amplitude do sinal à saída do MCU é sempre metade do valor obtido em simulação, consequência da gama de valores de tensão à saída do MCU ser de metade da gama dos valores à entrada.

4.2.3 – Resposta do Controlador PI (*dsPIC*)

A resposta do controlador PI é obtida à custa de uma entrada com valor constante de 0,25 V, conforme simulado no ponto 3.3 deste trabalho (simulação do controlador PI).

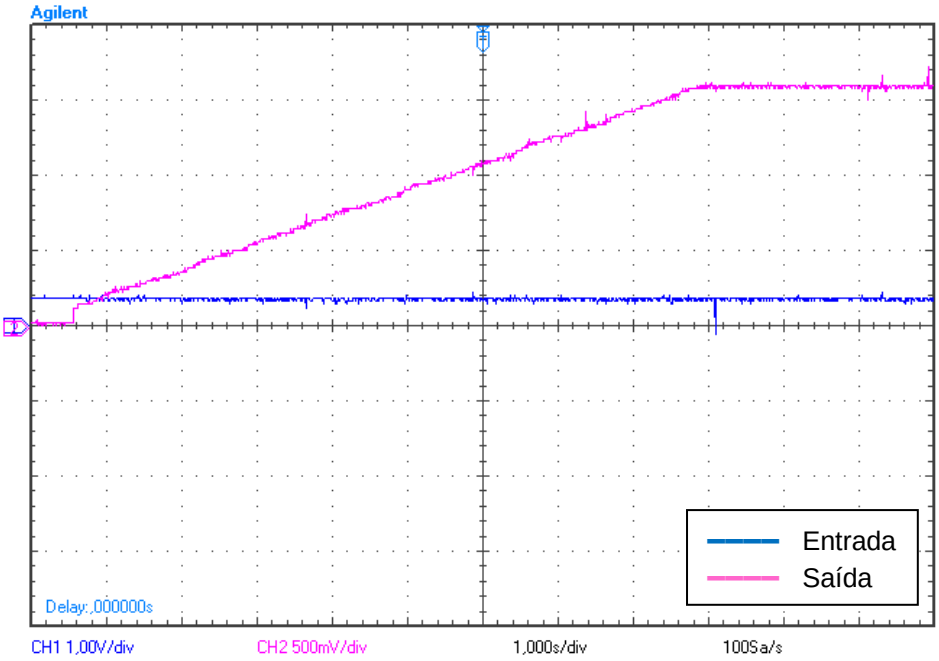
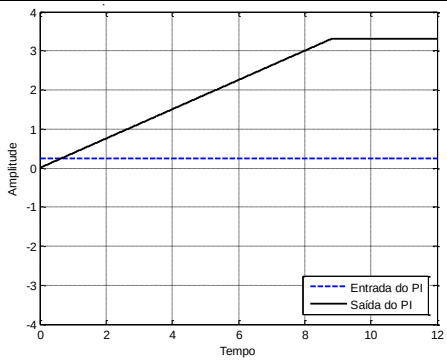
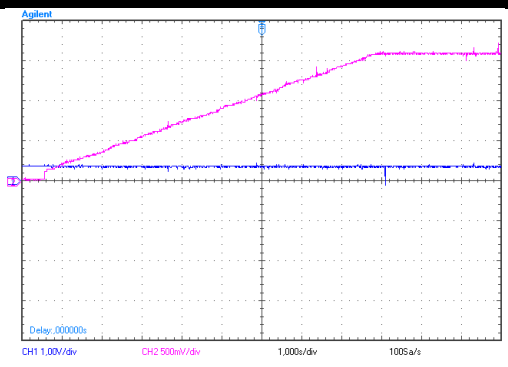


Figura 4.2.7 – Resposta de um controlador PI implementado em MCU, sinal de entrada de 0,25 V contantes, K_P=30x10^{-6} e K_I=1,5

Observa-se na prática que a amplitude do sinal à saída aumenta até que seja atingido o valor máximo possível que, neste caso, é de 1,65 V.

Tabela 4.2.3 – Comparação das respostas obtidas na simulação e no ensaio de um Controlador PI implementado em MCU com K_P=30x10^{-6} e K_I=1,5, entrada de 0,25 V constantes

Simulação	Implementado
	

A reta com declive positivo tem uma duração temporal diferente nas respostas do controlador PI simulado e implementado. Esta diferença deve-se ao facto de existir algum ruido na *breadboard* resultando num sinal que, na prática, não tem exatamente 0,25 V, existindo uma diferença suficiente para alterar o tempo que a saída demora a saturar.

O controlador PID não foi ensaiado pelas razões apresentadas no ponto 3.3 deste trabalho.

Capítulo V: Conclusões e Propostas de Trabalhos Futuros

5.1 – Conclusão

Os microcontroladores aparecem, hoje em dia, em aplicações que utilizamos diariamente na eletrónica de consumo, como áudio e vídeo digital (sistemas de comunicação, reprodução e reconhecimento de voz e imagem, etc.), assim como em aplicações médicas (aparelhos de electrocardiogramas, etc.) e ainda em aplicações militares análogas às civis.

Uma grande vantagem da utilização do processamento digital é a diminuição do *hardware* utilizado. Muitos dos componentes ativos e passivos utilizados no processamento analógico são substituídos por programação. Além da diminuição da complexidade dos circuitos, é possível alterar o *software* em qualquer altura, sem alterar o *hardware*, aumentado a flexibilidade no tipo de resposta desejada bem como no número de variáveis a controlar.

É também possível, através da utilização de MCU's, comunicar com outros sistemas através das portas de comunicação, permitindo controlo à distância ou com variáveis que possam depender de outros sistemas. Além disso, um sistema digital possui uma maior imunidade a interferências electromagnéticas pois requer apenas uma parte analógica de *interface* dado que todo o processamento se realiza na forma digital.

Quanto às implementações estudadas neste trabalho, os controladores PI e PID podem ser encontrados facilmente na indústria em aplicações como controlo de temperatura, nível de líquidos, pressão, velocidade, etc. O surgimento destes controladores, desenvolvidos de forma digital, apresenta contudo algumas desvantagens, nomeadamente:

- Atraso no processamento;
- Erro proveniente da conversão A/D;
- Maior custo, *a priori*, que controladores simples;
- Eventual necessidade de dimensionamento de filtros de *anti-aliasing* e de reconstrução;

Estas desvantagens são cada vez menos relevantes devido ao aumento da resolução dos conversores A/D e à maior capacidade de processamento que se verifica atualmente.

É importante ter noção das limitações que a aplicação de um MCU em controlo ou filtragem de sinais pode ter, nomeadamente quanto à frequência de amostragem. É possível, por

exemplo, implementar um filtro de ordem elevada à custa do aumento do número de cálculos, com a consequente diminuição da velocidade de processamento. Desta forma, ao ser reduzida a frequência de amostragem limita-se assim o desempenho do filtro ou a velocidade de resposta de um controlador.

O aumento da frequência de amostragem tem como consequência direta o aumento da frequência de corte máxima do filtro, aumentando a largura de banda do mesmo, enquanto nos controladores a frequência de amostragem não influencia de forma direta o desempenho. Em ambas as aplicações, existe uma relação entre a frequência de amostragem e a estabilidade dos sistemas e, com a diminuição desta frequência, aumenta a probabilidade de existência de *aliasing*.

Uma outra vantagem na utilização de MCU's em processamento e controlo é a capacidade de armazenamento de dados, que existe no controlo analógico mas de forma muito limitada. Torna-se então possível a gestão e supervisão de todas as variáveis de um processo. Também na utilização como controlador é possível utilizar várias técnicas de controlo recorrendo apenas a programação, sem ser necessário a alteração de componentes ativos e passivos externos.

Conclui-se assim, com este trabalho, que através de métodos de integração conhecidos, é possível transformar um sistema contínuo num discreto equivalente e utilizá-lo numa cadeia de aquisição e processamento de sinal obtendo resultados idênticos em ambos os domínios.

5.2 – Propostas e Trabalhos Futuros

Como proposta de desenvolvimento futuro refira-se que, embora a programação do MCU tenha sido realizada tendo em atenção a minimização do tempo de execução, poderá eventualmente ser desenvolvido código em ASM ao invés de linguagem C.

O estudo da exatidão e estabilidade sobre o ponto de vista de dimensão das palavras digitais utilizadas na discretização de constantes e de sinal de entrada é igualmente um dos aspetos que merece maior aprofundamento. Ainda, as diferenças entre o processamento em aritmética de vírgula flutuante e de inteiros, bem como as suas implicações do ponto de vista de truncagem, mereceria uma atenção especial.

A utilização de um MCU com uma frequência de funcionamento mais elevada permitirá a diminuição do tempo de execução do código, podendo também utilizar-se um conversor A/D com resolução mais elevada o que se traduziria num menor erro de quantificação.

5.1– Bibliografia

[[1] *Dogan Ibrahim – Microcontroller Based Applied Digital Control*; 1ª Edição, *Wiley,* 2006

[2] *Katsuhiko Ogata – Modern Control Engineering*; 5ª Edição, *Prentince* Hall, 2010

[3] *Gene F. Franklin, J. David Powel and Michael Workman – Digital Control of Dynamic*; 3ª Edição, *Addison Wesley Longman*, 1998

[4] *Lawrence R. Rabiner – Theory and Application of Digital Processing;* 1ª Edição*, Prentince Hall*, 1975

[5] *Vasco Soares* – APS – Aquisição e Processamento Digital de Sinais; ISEL – ADEEEA, 2012

[6] *J. V. Sopa Soares* – Controlo de Sistemas; ISEL – ADEEEA, 2005

[7] *Alan V. Oppenheim – Discrete Time Signal Processing Systems;* 2ª Edição*, Prentice Hall*, 1998

[8] *Gustavo da Silva* – Processamento Digital de Sinais; EST, 2000

[9] *João C.P. Palma* – Accionamentos Eletromecânicos de Velocidade Variável; 2.ª Edição, *Gulbenkian*, 1999

[10] *Microchip* dsPIC33FJ16GS502 *Datasheet*; *Microchip*, 2014

Anexos

Anexo 1 – Código em *Matlab* para Obtenção das Respostas no Domínio do Tempo

```
%################        Filtro PB de 1.ª Ordem       ############

fc=4000;
wc=2*pi*fc;

Filtro = tf([1], [1/wc 1])
figure (1)
step(Filtro)
xlabel('Tempo');
ylabel('Saída');
title('Resp. Temporal - Filtro PB de 1ª Ordem')

Hz = bodeoptions;
Hz.FreqUnits = 'Hz';
figure (2)
bode(Filtro,Hz,{100,10000000});
xlabel('Frequência')
ylabel('Fase')
title('Resp. em Freq. - Filtro Passa Baixo de 1ª Ordem');

%################        Filtro PB de 2.ª Ordem       ############

fc=4000;
wc=2*pi*fc;

figure (2)
Filtro2 = tf([1], [1/wc^2 sqrt(2)/wc 1])
step (Filtro2)
xlabel('Tempo');
ylabel('Saída');
title('Resp. Temporal - Filtro PB de 2ª Ordem');

Hz = bodeoptions;
Hz.FreqUnits = 'Hz';
figure (2)
bode(Filtro2,Hz,{100,10000000});
xlabel('Frequência')
ylabel('Fase')
title('Resp. em Freq. - Filtro Passa Baixo de 2ª Ordem');

%#################          Controlador PI        ###############

KP=1;
KI=1;

syms t;
Uout_PI = KP+KI*t      %Para e(t)=1
figure (3);
ezplot(Uout_PI, [0, 100]);
xlabel('Tempo (seconds)');
ylabel('Saída');
title('Resp. Temporal - Controlador PI');
axis([0 4 0 5]);
```

```
%##################         Controlador PID        ###############

KP=1;
KI=1;
KD=1;

syms t
Uout_PID = KP*t+KI*t^2+KD    %Para e(t)=t
figure (4);
ezplot(Uout_PID, [0, 100]);
xlabel('Tempo (seconds)');
ylabel('Saída');
title('Resp. Temporal - Controlador PID');
axis([0 2 0 5]);
```

Anexo 2 – Código em *Matlab* para Obtenção das Respostas Pelas Diferentes Integrações

```
%################        Filtro PB de 1.ª Ordem       ############

clear all
clc
fc=10;
wc=2*pi*fc;
Ts=0.01
n=0:Ts:1

%% _____________Integração Progressiva_____________

%Continue Domain
    F = tf([wc], [1 wc])
%Forward Rule
    F_f=tf([wc*Ts], [1 wc*Ts-1], Ts)
% Equação às Diferenças
    y_P=            1-2.6903.*0.3717.^(n/Ts)
    y_P(1)=1.6903 + 1-2.6903.*0.3717.^(0/Ts)

figure (1)
hold on
plot(n,y_P,'ro')
axis([0 0.1 0 1.05])
step(F, '--b', F_f, 'g')
axis([0 0.1 0 1.05])
legend('Representação da Equação às Diferenças', 'Domínio Contínuo', 'Domínio
Discreto')
xlabel('Tempo')
ylabel('Resposta')
title('Resposta Temporal do Filtro PB de 1ª Ordem - Int. Prog.');

%% _____________Integração Regressiva_____________

%Continue Domain
    F = tf([wc], [1 wc]);
%Backward Rule
    F_b=tf([wc*Ts 0], [1+wc*Ts  -1], Ts)
% Equação às Diferenças
    y_B=1-0.6143.^(n/Ts)

figure (2)
hold on
plot(n,y_B,'ro')
axis([0 0.1 0 1.05])
step(F, '--b', F_b, 'g')
axis([0 0.1 0 1.05])
legend('Representação da Equação às Diferenças', 'Domínio Contínuo', 'Domínio
Discreto')
xlabel('Tempo')
ylabel('Resposta')
title('Resposta Temporal do Filtro PB de 1ª Ordem - Int. Regres.');

%% _____________Integração Trapezoidal_____________

%Continue Domain
    F = tf([wc], [1 wc]);
%Trapezoidal Rule
    F_t=tf([wc*Ts wc*Ts], [2+wc*Ts  -2+wc*Ts], Ts)
%Equação às Diferenças
```

```
    y_T=            1-1.4590*0.5221.^(n/Ts)
    y_T(1)=0.4580 + 1-1.4590*0.5221.^(0/Ts)

figure (3)
hold on
plot(n,y_T,'ro')
axis([0 0.1 0 1.05])
step(F, '--b', F_t, 'g')
axis([0 0.1 0 1.05])
legend('Representação da Equação às Diferenças', 'Domínio Contínuo', 'Domínio
Discreto')
xlabel('Tempo')
ylabel('Resposta')
title('Resposta Temporal do Filtro PB de 1ª Ordem - Int. Trap.');

%%
%################        Filtro PB de 2.ª Ordem       ############

clear all
clc
fc=10;
wc=2*pi*fc;
Ts=0.01
n=0:Ts:1

%% _____________Integração Progressiva_____________

%Continue Domain
    F2 = tf([wc^2], [1 sqrt(2)*wc wc^2])
%Forward Rule
    F2_f=tf([(wc*Ts)^2], [1 sqrt(2)*wc*Ts-2 1-sqrt(2)*wc*Ts+(wc*Ts)^2], Ts)
%Equação às Diferenças
%    y_P=            1.0003+109.3310.*(0.3639.^(n/Ts))-
107.4730.*(0.3795.^(n/Ts))
%    y_P(1)=-2.8588 +1.0003+109.3310.*(0.3639.^(0/Ts))-
107.4730.*(0.3795.^(0/Ts))
    y_P=            0.9991+(-0.1095-0.9869*j).*((0.5555-0.4445*j).^(n/Ts))+(-
0.1095+0.9869*j).*((0.5555+0.4445*j).^(n/Ts))
    y_P(1)=-0.7800 +0.9991+(-0.1095-0.9869*j).*((0.5555-0.4445*j).^(0/Ts))+(-
0.1095+0.9869*j).*((0.5555+0.4445*j).^(0/Ts))

figure (1)
hold on
plot(n,y_P,'ro')
axis([0 0.2 0 1.25])
step(F2, '--b', F2_f, 'g')
axis([0 0.2 0 1.25])
legend('Representação da Equação às Diferenças', 'Domínio Contínuo', 'Domínio
Discreto')
xlabel('Tempo')
ylabel('Resposta')
title('Resposta Temporal do Filtro PB de 2ª Ordem - Int. Prog.');

%% _____________Integração Regressiva_____________

%Continue Domain
    F2 = tf([wc^2], [1 sqrt(2)*wc wc^2])
%Backward Rule
    F2_b=tf([(wc*Ts)^2 0 0], [1+sqrt(2)*wc*Ts+(wc*Ts)^2 -2-sqrt(2)*wc*Ts 1], Ts)
%Equação às Diferenças
    y_B=1.0019+(-0.4145+0.2206*j).*((0.6327+0.1941*j).^(n/Ts))+(-0.4145-
0.2206*j).*((0.6327-0.1941*j).^(n/Ts))

figure (2)
hold on
```

```
plot(n,y_B,'ro')
axis([0 0.2 0 1.15])
step(F2, '--b', F2_b, 'g')
axis([0 0.2 0 1.15])
legend('Representação da Equação às Diferenças', 'Domínio Contínuo', 'Domínio
Discreto')
xlabel('Tempo')
ylabel('Resposta')
title('Resposta Temporal do Filtro PB de 2ª Ordem - Int. Regres.');

%% _____________Integração Trapezoidal_____________

%Continue Domain
    F2 = tf([wc^2], [1 sqrt(2)*wc wc^2])
%Trapezoidal Rule
    F2_t=tf([(wc*Ts)^2 2*(wc*Ts)^2 (wc*Ts)^2], [4+2*sqrt(2)*wc*Ts+(wc*Ts)^2 -
8+2*(wc*Ts)^2  4-2*sqrt(2)*wc*Ts+(wc*Ts)^2], Ts)
%Equação às Diferenças
    y_T=            0.9996+(-0.4244+0.7634*j).*((0.5841+0.2881*j).^(n/Ts))+(-
0.4244-0.7634*j).*((0.5841-0.2881*j).^(n/Ts))
    y_T(1)=-0.1509 +0.9996+(-0.4244+0.7634*j).*((0.5841+0.2881*j).^(0/Ts))+(-
0.4244-0.7634*j).*((0.5841-0.2881*j).^(0/Ts))

figure (3)
hold on
plot(n,y_T,'ro')
axis([0 0.15 0 1.15])
step(F2, '--b', F2_t, 'g')
axis([0 0.15 0 1.15])
legend('Representação da Equação às Diferenças', 'Domínio Contínuo', 'Domínio
Discreto')
xlabel('Tempo')
ylabel('Resposta')
title('Resposta Temporal do Filtro PB de 2ª Ordem - Int. Trap.');


%#################          Controlador PI        ###############

clear all
clc
KP=0.1
KI=1
Ts=0.01
n=0:Ts:1

%% _____________Integração Progressiva_____________

%Contínuo
    Fc = tf([KP KI], [1 0])
%Discreto
    Fc_f=tf([KP -KP+Ts*KP*KI], [1 -1], Ts)
%Equação às Diferenças
    y_P=          0.1+0.001*(n/Ts)
    y_P(1)=0.1 +  0.1+0.001*(0/Ts)

figure (1)
hold on
plot(n,y_P,'ro')
axis([0 0.2 0 0.15])
step(Fc, '--b',Fc_f, 'g')
axis([0 0.2 0 0.15])
legend('Representação da Equação às Diferenças', 'Domínio Contínuo', 'Domínio
Discreto')
xlabel('Tempo')
ylabel('Resposta')
```

```
title('Resposta Temporal do Controlador PI - Int. Prog.');
%% _____________Integração Regressiva_____________

%Contínuo
    Fc = tf([KP KI], [1 0])
%Discreto
    Fc_b=tf([KP+KI*Ts -KP], [1 -1], Ts)
%Equação às Diferenças
    y_B=0.101+0.001*(n/Ts)

figure (2)
hold on
plot(n,y_B,'ro')
axis([0 0.2 0 0.15])
step(Fc, '--b',Fc_b, 'g')
axis([0 0.2 0 0.15])
legend('Representação da Equação às Diferenças', 'Domínio Contínuo', 'Domínio
Discreto')
xlabel('Tempo')
ylabel('Resposta')
title('Resposta Temporal do Controlador PI - Int. Regres.');

%% _____________Integração Trapezoidal_____________

%Contínuo
    Fc = tf([KP KI], [1 0])
%Discreto
    Fc_t=tf([2*KP+KI*Ts -2*KP+KI*Ts], [2 -2], Ts)
%Equação às Diferenças
    y_T=0.0995+0.001*(n/Ts)
    y_T(1)=0.0995

figure (3)
hold on
plot(n,y_T,'ro')
axis([0 0.2 0 0.15])
step(Fc, '--b',Fc_t, 'g')
axis([0 0.2 0 0.15])
legend('Representação da Equação às Diferenças', 'Domínio Contínuo', 'Domínio
Discreto')
xlabel('Tempo')
ylabel('Resposta')
title('Resposta Temporal do Controlador PI - Int. Trap.');
```

Anexo 3 – Código em *Matlab* para Obtenção das Respostas Pelas Diferentes Integrações na mesma Figura

```
%################        Filtro PB de 1ª Ordem       ############

Ts=0.01
fc=10;
wc=2*pi*fc;

%Continue Domain
    F = tf([wc], [1 wc])

%Forward Rule
    F_f=tf([wc*Ts], [1 wc*Ts-1], Ts)
    Ffs=d2c(F_f)

%Backward Rule
    F_b=tf([wc*Ts 0], [1+wc*Ts  -1], Ts)
    Fbs=d2c(F_b)

%Trapezoidal Rule
    F_t=tf([wc*Ts wc*Ts], [2+wc*Ts  -2+wc*Ts], Ts)
    Fts=d2c(F_t)

%Resposta Temporal
    figure (1)
    step(F_f, F_b, F_t, F, '--c', .1);
    legend('Resposta pela Integração Progressiva', 'Resposta pela Integração
Regressiva', 'Resposta pela Integração Trapezoidal', 'Resposta no Domínio
Contínuo');
    xlabel('Tempo')
    ylabel('Resposta')
    title('Resp. Temporal do Filtro PB1 pelos Diversos Tipos de Integração');

%Resposta em Frequência
    opt = bodeoptions;
    opt.FreqUnits = 'Hz';
    opt.MagLowerLimMode = 'manual';
    opt.MagLowerLim = -40;

    figure (2)
    bode(F_f, F_b, F_t, F, opt, {1,1000});
    legend('Resposta pela Integração Progressiva', 'Resposta pela Integração
Regressiva', 'Resposta pela Integração Trapezoidal', 'Resposta no Domínio
Contínuo')
    title('Resp. em Freq. do Filtro PB1 pelos Diversos Tipos de Integração');
    ylabel('Fase')
    xlabel('Frequência')

%%
%################        Filtro PB de 2ª Ordem       ############

clear all
clc

Ts=0.01;
fc=10;
wc=2*pi*fc;

%Continue Domain
    F2 = tf([wc^2], [1 sqrt(2)*wc wc^2])
```

```
%Forward Rule
    F2_f=tf([(wc*Ts)^2], [1 sqrt(2)*wc*Ts-2 1-sqrt(2)*wc*Ts+(wc*Ts)^2], Ts)
    Ffs=d2c(F2_f)

%Backward Rule
    F2_b=tf([(wc*Ts)^2 0 0], [1+sqrt(2)*wc*Ts+(wc*Ts)^2 -2-sqrt(2)*wc*Ts 1], Ts)
    Fbs=d2c(F2_b)

%Trapezoidal Rule
    F2_t=tf([(wc*Ts)^2 2*(wc*Ts)^2 (wc*Ts)^2], [4+2*sqrt(2)*wc*Ts+(wc*Ts)^2 -
8+2*(wc*Ts)^2  4-2*sqrt(2)*wc*Ts+(wc*Ts)^2], Ts)
    Fts=d2c(F2_t)


%Resposta Temporal
    figure (1)
    step(F2_f, F2_b, F2_t, F2, '--c');
    axis([0 0.15 0 1.25])
    legend('Resposta pela Integração Progressiva', 'Resposta pela Integração
Regressiva', 'Resposta pela Integração Trapezoidal', 'Resposta no Domínio
Contínuo');
    xlabel('Tempo')
    ylabel('Resposta')
    title('Resp. Temporal do Filtro PB de 2º Ordem pelos Diversos Tipos de
Integração');

%Resposta em Frequência
    opt = bodeoptions;
    opt.FreqUnits = 'Hz';
    opt.MagLowerLimMode = 'manual';
    opt.MagLowerLim = -40;

    figure (2)
    bode(F2_f, F2_b, F2_t, F2, opt, {1,1000});
    legend('Resposta pela Integração Progressiva', 'Resposta pela Integração
Regressiva', 'Resposta pela Integração Trapezoidal', 'Resposta no Domínio
Contínuo')
    xlabel('Frequência')
    ylabel('Fase')
    title('Resp. em Freq. do Filtro PB de 2º Ordem pelos Diversos Tipos de
Integração');

% Resposta do Filtro PB 2 para diferentes tempos de amostragem

% Trapezoidal Rule
    fc=10
    wc=2*pi*fc

    ws1=10*wc
    Ts1=1/(ws1/(2*pi))
    ws2=20*wc
    Ts2=1/(ws2/(2*pi))
    ws3=40*wc
    Ts3=1/(ws3/(2*pi))

    F2_t1=tf([(wc*Ts1)^2 2*(wc*Ts1)^2 (wc*Ts1)^2],
[4+2*sqrt(2)*wc*Ts1+(wc*Ts1)^2 -8+2*(wc*Ts1)^2  4-2*sqrt(2)*wc*Ts1+(wc*Ts1)^2],
Ts1)
    F2_t2=tf([(wc*Ts2)^2 2*(wc*Ts2)^2 (wc*Ts2)^2],
[4+2*sqrt(2)*wc*Ts2+(wc*Ts2)^2 -8+2*(wc*Ts2)^2  4-2*sqrt(2)*wc*Ts2+(wc*Ts2)^2],
Ts2)
```

```
    F2_t3=tf([(wc*Ts3)^2 2*(wc*Ts3)^2 (wc*Ts3)^2],
[4+2*sqrt(2)*wc*Ts3+(wc*Ts3)^2 -8+2*(wc*Ts3)^2  4-2*sqrt(2)*wc*Ts3+(wc*Ts3)^2],
Ts3)

    opt = bodeoptions;
    opt.FreqUnits = 'Hz';
    opt.MagLowerLimMode = 'manual';
    opt.MagLowerLim = -40;

    figure (10)
    bode(F2_t1, F2_t2, F2_t3);
    legend('\omegas=10\omegac', '\omegas=20\omegac', '\omegas=40\omegac')
    xlabel('Frequência')
    ylabel('Fase')
    title('Resp. em Freq. do Filtro PB de 2º Ordem com variação do tempo de
amostragem');

%%
%#################          Controlador PI        ###############

clear all
clc

KP=0.1
KI=1
Ts=0.01

%Continue Domain
    Fc = tf([KP KI], [1 0])

%Forward Rule
    Fc_f=tf([KP -KP+Ts*KI], [1 -1], Ts)
    Fcfs=d2c(Fc_f)

%Backward Rule
    Fc_b=tf([KP+KI*Ts -KP], [1 -1], Ts)
    Fcbs=d2c(Fc_b)

%Trapezoidal Rule
    Fc_t=tf([2*KP+KI*Ts -2*KP+KI*Ts], [2 -2], Ts)
    Fcts=d2c(Fc_t)

%Resposta Temporal
    figure (4)
    step(Fc_f, Fc_b, Fc_t, Fc);
    axis([0 0.2 0 0.3])
    legend('Resposta pela Integração Progressiva', 'Resposta pela Integração
Regressiva', 'Resposta pela Integração Trapezoidal', 'Resposta no Domínio
Contínuo');
    xlabel('Tempo')
    ylabel('Resposta')
    title('Resposta Temporal do Controlador PI pelos Diversos Tipos de
Integração');

%Resposta em Frequência
    opt = bodeoptions;
    opt.FreqUnits = 'Hz';

    figure (5)
    bode(Fc_f, Fc_b, Fc_t, Fc, opt, {1,1000});
    legend('Resposta pela Integração Progressiva', 'Resposta pela Integração
Regressiva', 'Resposta pela Integração Trapezoidal', 'Resposta no Domínio
Contínuo')
    xlabel('Frequência')
```

```
    ylabel('Fase')
    title('Resp. em Freq. do Controlador PI pelos Diversos Tipos de
Integração');

%##################        Controlador PID        ###############
clear all
clc

KD=0.1
KP=1
KI=1
Ts=0.01

%Continue Domain
     Fc = tf([KD KP KI], [0.0001 1 0.0001])

%Forward Rule
     Fc_f=tf([KP -KP+Ts*KI], [1 -1], Ts)
     Fcfs=d2c(Fc_f)

%Backward Rule
    Fc_b=tf([KD+Ts*KP+(Ts^2)*KI -2*KD-Ts*KP KD], [Ts -Ts 0], Ts)
    Fcfs=d2c(Fc_b)

%Trapezoidal Rule
    Fc_t=tf([4*KD+2*Ts*KP+(Ts^2)*KI -8*KD+2*(Ts^2)*KI 4*KD-2*Ts*KP+(Ts^2)*KI],
[2*Ts 0 -2*Ts], Ts)
    Fcfs=d2c(Fc_t)
```

Anexo 4 – Código em *Matlab* para Obtenção da Estabilidade Relativa

```
% Estabilidade na Máquina DC com controlador PI

clear all
clc

% Definição das Constantes
    U=60;
    A=10;
    ucmax=A;
    ke=U/ucmax;
    Ra=3;
    La=20e-3;
    Ta=La/Ra;
    Tc=Ta;
    f=10000;
    T=1/f;
    Te=T/2;
    KI=5000;

    Ts=10e-6

FT_d = tf([(KI*ke*Ts^2)/Ra], [Te -2*Te+Ts Te-Ts+(KI*ke*Ts^2)/Ra], Ts)

figure (2)
margin (FT_d)    % Margem de estabilidade em 'z'
```

Anexo 5 – Código em *Matlab* para Obtenção das Respostas do Simulink

```
fc=4000;
wc=2*pi*fc
Ts=0.00001

%################        Filtro PB de 1.ª Ordem       ############
af0=(wc*Ts)/(2+wc*Ts)
af1=af0
bf0=1
bf1=(-2+wc*Ts)/(2+wc*Ts)

%################        Filtro PB de 2.ª Ordem       ############
k=sqrt(2)/2
af20=((wc^2*Ts^2))/(4+4*k*wc*Ts+(wc^2*Ts^2))
af21=2*af20
af22=af20
bf20=1
bf21=(-8+2*(wc^2*Ts^2))/(4+4*k*wc*Ts+(wc^2*Ts^2))
bf22=(4-4*k*wc*Ts+(wc^2*Ts^2))/(4+4*k*wc*Ts+(wc^2*Ts^2))

%##################          Controlador PI        ###############
KP=0.00001
KI=1.5

api0=KP+0.5*KI*Ts
api1=-KP+0.5*KI*Ts
bpi0=1
bpi1=-1

%##################         Controlador PID        ###############
KP=0.00003
KI=1.5
KD=0.00000

apid0= (4*KD+2*Ts*KP+Ts^2*KI)/(2*Ts)
apid1= (-8*KD+2*Ts^2*KI)/(2*Ts)
apid2= (4*KD-2*Ts*KP+Ts^2*KI)/(2*Ts)
bpid0= 1
bpid1= 0
bpid2= -1

%%  Respostas do Filtro PB 1ª Ordem
figure (1)
plot(ContF.time, ContF.signals.values, 'b', DiscF.time,
DiscF.signals.values,'r', CanF.time, CanF.signals.values, '--k',
'linewidth',2);
axis([0 0.0002 0 1])
grid on;
title('Respostas no Filtro PB de 1ª Ordem');
legend('FT Domínio Contínuo','FT Domínio Discreto', 'Estrutura Canónica');
xlabel('Tempo');
ylabel('Amplitude');

figure (2)
subplot (3, 1, 1)
plot(Sin_1000Hz.time, Sin_1000Hz.signals.values, 'k', Sind_1000Hz.time,
Sind_1000Hz.signals.values, 'r',  'linewidth',1);
axis([0 0.002 0 2])
grid off;
title('Respostas do Filtro PB 1ª ordem a Sinusoide de 1kHz');
```

```
subplot (3, 1, 2)
plot(Sin_2000Hz.time, Sin_2000Hz.signals.values, 'k', Sind_2000Hz.time,
Sind_2000Hz.signals.values, 'r', 'linewidth',1);
axis([0 0.002 0 2])
grid off;
title('Respostas do Filtro PB 1ª ordem a Sinusoide de 2kHz');
ylabel('Amplitude (V)');
subplot (3, 1, 3)
plot(Sin_4000Hz.time, Sin_4000Hz.signals.values, 'k', Sind_4000Hz.time,
Sind_4000Hz.signals.values, 'r',  'linewidth',1);
axis([0 0.002 0 2])
grid off;
title('Respostas do Filtro PB 1ª ordem a Sinusoide de 4kHz');
legend('Entrada','Saída');
xlabel('Tempo (s)');

figure (3)
plot(Sin_1000Hz.time, Sin_1000Hz.signals.values, 'k', Sind_1000Hz.time,
Sind_1000Hz.signals.values, 'r',  'linewidth',1);
axis([0+0.3e-3 0.0012+0.3e-3 -4 4])
grid on;
legend('Entrada','Saída');
xlabel('Tempo (s)');

figure (4)
plot(Sin_2000Hz.time, Sin_2000Hz.signals.values, 'k', Sind_2000Hz.time,
Sind_2000Hz.signals.values, 'r',  'linewidth',1);
axis([0+0.33e-3 0.0012+0.33e-3 -4 4])
grid on;
legend('Entrada','Saída');
xlabel('Tempo (s)');

figure (5)
plot(Sin_4000Hz.time, Sin_4000Hz.signals.values, 'k', Sind_4000Hz.time,
Sind_4000Hz.signals.values, 'r',  'linewidth',1);
axis([0+0.4e-3 0.0012+0.4e-3 -4 4])
grid on;
legend('Entrada','Saída');
xlabel('Tempo (s)');

%%  Respostas do Filtro PB 2ª Ordem

figure (6)
plot(ContF2.time, ContF2.signals.values, 'b', DiscF2.time,
DiscF2.signals.values,'r', CanF2.time, CanF2.signals.values, '--k',
'linewidth',2);
axis([0 0.0005 0 1.15])
grid on;
title('Respostas no Filtro PB (Butterworth) de 2ª Ordem');
legend('FT Domínio Contínuo','FT Domínio Discreto', 'Estrutura Canónica');
xlabel('Tempo');
ylabel('Amplitude');

figure (7)
subplot (3, 1, 1)
plot(Sin2_1000Hz.time, Sin2_1000Hz.signals.values, 'k', Sin2d_1000Hz.time,
Sin2d_1000Hz.signals.values, 'r',  'linewidth',1);
axis([0 0.002 0 3])
grid off;
title('Respostas do Filtro PB 2ª Ordem a Sinusoide de 1kHz');
subplot (3, 1, 2)
plot(Sin2_2000Hz.time, Sin2_2000Hz.signals.values, 'k', Sin2d_2000Hz.time,
Sin2d_2000Hz.signals.values, 'r', 'linewidth',1);
axis([0 0.002 0 3])
grid off;
title('Respostas do Filtro PB 2ª Ordem a Sinusoide de 2kHz');
```

```
ylabel('Amplitude (V)');
subplot (3, 1, 3)
plot(Sin2_4000Hz.time, Sin2_4000Hz.signals.values, 'k', Sin2d_4000Hz.time,
Sin2d_4000Hz.signals.values, 'r',  'linewidth',1);
axis([0 0.002 0 3])
grid off;
title('Respostas do Filtro PB 2ª Ordem a Sinusoide de 4kHz');
legend('Entrada','Saída');
xlabel('Tempo (s)');

figure (8)
plot(Sin2_1000Hz.time, Sin2_1000Hz.signals.values, 'k', Sin2d_1000Hz.time,
Sin2d_1000Hz.signals.values, 'r',  'linewidth',1);
axis([0+0.4e-3 0.0012+0.4e-3 -4 4])
grid on;
legend('Entrada','Saída');
xlabel('Tempo (s)');

figure (9)
plot(Sin2_2000Hz.time, Sin2_2000Hz.signals.values, 'k', Sin2d_2000Hz.time,
Sin2d_2000Hz.signals.values, 'r',  'linewidth',1);
axis([0+0.4e-3 0.0012+0.4e-3 -4 4])
grid on;
legend('Entrada','Saída');
xlabel('Tempo (s)');

figure (10)
plot(Sin2_4000Hz.time, Sin2_4000Hz.signals.values, 'k', Sin2d_4000Hz.time,
Sin2d_4000Hz.signals.values, 'r',  'linewidth',1);
axis([0+0.4e-3 0.0012+0.4e-3 -4 4])
grid on;
legend('Entrada','Saída');
xlabel('Tempo (s)');

%%  Respostas do Controlador PI

figure (11)
plot(ContPI.time, ContPI.signals.values, '--b', CanPI.time,
CanPI.signals.values, 'k',  'linewidth',2);
axis([0 12 -4 4])
grid on;
title('Respostas do Controlador PI a Entrada de Valor Constante');
legend('Entrada do PI','Saída do PI');
ylabel('Amplitude');
xlabel('Tempo');

%%  Respostas do Controlador PID

figure (12)
plot(StepPID.time, StepPID.signals.values, 'b', ContPID.time,
ContPID.signals.values, 'k', CanPID.time, CanPID.signals.values, '--r',
'linewidth',2);
axis([0 0.1 0 1.5])
grid off;
title('Respostas do Controlador PID a Entrada do Tipo Degrau');
legend('Entrada Degrau','Resposta no Domínio Contínuo', 'Resposta no Domínio
Discreto','Resposta no Domínio Discreto');
ylabel('Amplitude');
xlabel('Tempo');

    figure (13)
    subplot (3, 1, 1)
    plot(StepPID.time, StepPID.signals.values, 'b', ContPID.time,
ContPID.signals.values, 'k', DiscPID.time, DiscPID.signals.values, '--r',
'linewidth',2);
```

```
    axis([0 0.1 0 1.5])
    grid off;
    title('Respostas do Controlador PI (Equações no Domínio Contínuo e Equação
Discreta)');
    subplot (3, 1, 2)
    plot(StepPID.time, StepPID.signals.values, 'b', ContPID.time,
ContPID.signals.values, 'k', CanPID.time, CanPID.signals.values, '--r',
'linewidth',2);
    axis([0 0.1 0 1.5])
    grid off;
    title('Respostas do Controlador PI (Eq. no Domínio Contínuo e Estrutura
Canónica)');
    ylabel('Amplitude');
    subplot (3, 1, 3)
    plot(StepPID.time, StepPID.signals.values, 'b', ContPID.time,
ContPID.signals.values, 'k', NCanPID.time, NCanPID.signals.values, '--r',
'linewidth',2);
    axis([0 0.1 0 1.5])
    grid off;
    title('Respostas do Controlador PI (Eq. no Domínio Contínuo e Estrutura Não
Canónica)');
    legend('Step','Contínuo','Discreto');
    xlabel('Tempo');
```

Anexo 6 – Código em C para Implementação do Filtro PB 1 no MCU

```
#include<xc.h>

// FBS
#pragma config BWRP = WRPROTECT_OFF      // Proteção de Escrita na Inicialização
#pragma config BSS = NO_FLASH            // Código de Programa Flash na Inicialização

// FGS
#pragma config GWRP = OFF          // Proteção de Escrita – Pode ser Escrito
#pragma config GSS = OFF           // Proteção de Código – Memória do Utilizador não Protegida

// FOSCSEL
#pragma config FNOSC = FRC         // Escolha da Fonte do Oscilador  - Internal Fast RC (FRC) oscillator
#pragma config IESO = ON           // Iniciar MCU com FRC e passar só depois para fonte do oscilador escolhida

// FOSC
#pragma config POSCMD = NONE       // Fonte do Oscilador Primário
#pragma config OSCIOFNC = ON       // Função do Pin OSC2 – Porta I/O Digital
#pragma config IOL1WAY = ON        // Configuração da Escolha de Pin Periférica
#pragma config FCKSM = CSECMD      // Clock Switching ativo e Fail-Safe Clock Monitor inativo

// FWDT
#pragma config WDTPOST = PS32768  // Postscaler do Temporizador do Watchdog (1:32,768)
#pragma config WDTPRE = PR128      // WDT Prescaler (1:128)
#pragma config WINDIS = OFF        // Watchdog Timer Window
#pragma config FWDTEN = OFF        // Ativar Temporizador de Watchdog

// FPOR
#pragma config FPWRT = PWR128      // Valor do Timer POR (128ms)

// FICD
#pragma config ICS = PGD1          // Seleção de Canal Com (Comunicar em PGC1/EMUC1 e PGD1/EMUD1)
#pragma config JTAGEN = OFF        // Inativar Porta JTAG

void configure_pins ();

void __attribute__((__interrupt__)) _OscillatorFail(void);
void __attribute__((__interrupt__)) _AddressError(void);
void __attribute__((__interrupt__)) _StackError(void);
void __attribute__((__interrupt__)) _MathError(void);

void  main (void)
{
   unsigned long int Uk  = 0;                          // Sinal de Entrada Atual
   unsigned long int Uk1 = 0;                          // Sinal de Entrada na Amostra Anterior
   unsigned long int Yk  = 0;                          // Sinal de Saída
   unsigned long int Yk1 = 0;                          // Sinal de Saída a amostra anterior

   float Ts=0.000010;                                  // Tempo de amostragem
   float fc=4000;                                      // Frequência de Corte
   float wc=2*3.14*fc;                                 // Constante Omega  – Frequência Angular de Corte
   float a0_aux=((wc*Ts)/(2+wc*Ts))*10000;             // Cálculo de a0 e a1
   float b1_aux=((wc*Ts-2)/(2+wc*Ts))*10000;           // Cálculo de b1

   int a0 = a0_aux;                                    // Inicializar a0
   int a1 = a0;                                        // Inicializar a1
   int b1 = b1_aux;                                    // Inicializar b1

   configure_pins ();                                  // Configuração dos Pins

   while (1)                                           // Loop Infinito
   {
      PORTBbits.RB11 = 1;                              // Saída Auxiliar para Visualizar Tempo de Amostragem = 1

      while(ADCPC0bits.PEND0);                         // Esperar pela conversão A/D
      Uk = ADCBUF1;                                    // Ler valor da conversão
      ADCPC0bits.SWTRG0 = 1;                           // Iniciar nova conversão

      Yk = (a0*Uk + a1*Uk1 - b1*Yk1)/10000;            // Atualizar Saída

      if (Yk>1023)                                     // Limitador Superior
```

```
        Yk=1023;                                    // Valor Máximo
      if (Yk<0)                                     // Limitador Inferior
        Yk=0;                                       // Valor Minímo

      CMPDAC1bits.CMREF = Yk;              // Colocar Valor Calculado na Saída REF*(AVDD/2)/1024)volts

      Uk1=Uk;                                       // Atualizar Uk-1
      Yk1=Yk;                                       // Atualizar Yk1

      while (PORTBbits.RB11 == 1)                   // Esperar pelo final da Contagem do Timer de Ts
      {}
   }
}

void configure_pins ()
{
   INTCON1bits.NSTDIS = 0;                          // Ativar Interrupção por Nesting

   CLKDIVbits.FRCDIV = 0;                           // Divisão da Frequência por 1          (Default: 0)
   CLKDIVbits.PLLPOST = 0;                          // (N1) Divisão da Frequência por 2     (Default: 1)
   CLKDIVbits.PLLPRE = 2;                           // (N2) Divisão da Frequência por 2     (Default: 0)
   PLLFBD = 106;                                    // M)= PVVFBD+2  Multip. da Frequência  (Default: 50)

   __builtin_write_OSCCONH(0x01);                   // Novo Oscillator FRC com PLL
   __builtin_write_OSCCONL(0x01);                   // Ativar Interruptor

   while(OSCCONbits.COSC != 0b001);                 // Esperar que o Novo Oscilador se Torne FRC com  PLL
   {};
   while(OSCCONbits.LOCK != 1);                     // Esperar que o PLL bloqueie
   {};

                                                    // Configuração de Entradas e Saídas
   TRISB = 0x00F0;                                  // <RB0:RB7> como Entradas e <RB8:RB15> como Saídas
   PORTB = 0;                                        // Colocar PORTB a zero
                                                    // Configuração da entrada Analógica
   ADPCFG = 0x0F00;                                 // <AN0:AN7> Como entradas Analógicas
                                                    // Configurar Módulo DAC
   CMPCON1bits.CMPON = 0;                           // Inativar Comparador de Alta-Velocidade
   CMPCON1bits.CMPSIDL = 0;                         // Continuar Módulo em Modo Ocioso (Idle Mode)
   CMPCON1bits.DACOE = 1;                           // Permitir comunicação entre Módulo DAC e pin DACOUT
   CMPCON1bits.INSEL = 0;                           // Pin de Alimentação a Entrada Analógica não Influencia
   CMPCON1bits.EXTREF = 0;                          // Utilizar referência Interna de tensão
   CMPCON1bits.CMPSTAT = 0;                         // Comparador de Saída Inativo – Não Utilizado
   CMPCON1bits.CMPPOL = 0;                          // Valor do Comparador de Saída não utilizado
   CMPCON1bits.RANGE = 1;          // Gama da Saída do DAC Output definida para  AVDD/2 (1.65V @ 3.3V AVDD)
   CMPCON1bits.CMPON = 1;                           // Ativar Comparador de Alta Velocidade

                                                    // Configuração do Módulo ADC
   ADCONbits.ADSIDL   = 0;                          // Funcionar em Modo Ocioso (Idle Mode)
   ADCONbits.FORM     = 0;                          // Saída no formato inteiro
   ADCONbits.EIE      = 1;                          // Ativar Interrupção Prévia
   ADCONbits.ORDER    = 1;                          // Canal Odd com Prioridade
   ADCONbits.SEQSAMP  = 0;                          // Amostragem Sequencial Ativa
   ADCONbits.ADCS     = 0;                          // Divisão do Clock Definida para Fadc/14
   ADCONbits.SLOWCLK  = 0;
   ADSTAT             = 0;                          // Limpar Registo ADSTAT
   ADCPC0bits.TRGSRC0 = 1;                          // Utilizar SW trigger
   ADCPC0bits.IRQEN0  = 1;                          // Ativar Interrupção
   ADCONbits.ADON     = 1;                          // Iniciar Módulo ADC
   IFS0bits.ADIF      = 0;                          // Limpar Flag de Interrupção do Módulo
   IEC0bits.ADIE      = 1;                          // Ativar Interrupção do ADC
   ADCPC0bits.SWTRG0  = 1;                          // Alteração de Trigger ao Par de Conversão

   T1CONbits.TON = 0;                               // Inativar Timer
   T1CONbits.TCS = 0;                               // Selecionar ciclo de clock por instrução interna
   T1CONbits.TGATE = 0;                             // Modo Gated Timer Inativo
   T1CONbits.TCKPS = 0b00;                          // Selecionar 1:1 para Prescaler
   TMR1 = 0x00;                                     // Limpar registo do Timer
   PR1 = 505;                                       // Carregar Valor  (1/Fosc)*Prescaler*PR1/2 = T1 (s)
   IPC0bits.T1IP = 0x01;                            // Escolher Prioridade da Interrupção do Timer
   IFS0bits.T1IF = 0;                               // Limpar Flag de Interrupção do Timer 1
   IEC0bits.T1IE = 1;                               // Ativar Interrupção do Timer 1
   T1CONbits.TON = 1;                               // Iniciar Timer 1
}
```

```
void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void)
{
IFS0bits.T1IF = 0;                                  // Limpar Interrupção do Timer 1

PORTBbits.RB11 = 0;                                 // Saída Auxiliar para Visualizar Tempo de Amostragem = 0

if(PORTBbits.RB8 == 0)                              // Comparador de LED de Estado
   PORTBbits.RB8 = 1;                               // LED de Estado ON
else
   PORTBbits.RB8 = 0;                               // LED de Estado OFF
}
void __attribute__ ((interrupt, no_auto_psv)) _ADCInterrupt(void)
{
                                                    // Interrupção Prévia – Conversão AD Completa
   int Uk;
   IFS0bits.ADIF = 0;                               // Limpar Flag de Interrupção
   Uk = ADCBUF0;                                    // Extrair Valor da Convers
   ADSTATbits.P0RDY = 0;                            // Limpar os bits do registo ADSTAT
}
void __attribute__((interrupt, no_auto_psv)) _OscillatorFail(void)
{
      INTCON1bits.OSCFAIL = 0;                      //Limpar Flag de Interrupção por Falha do Oscilador
      asm("RESET");
}
void __attribute__((interrupt, no_auto_psv)) _AddressError(void)
{
      INTCON1bits.ADDRERR = 0;                      // Limpar Flag de Interrupção por Erro de Endereço

      asm("RESET");
}
void __attribute__((interrupt, no_auto_psv)) _StackError(void)
{
      INTCON1bits.STKERR = 0;                       // Limpar Flag de Interrupção por Erro no Apontador
      asm("RESET");
}
void __attribute__((interrupt, no_auto_psv)) _MathError(void)
{
      INTCON1bits.MATHERR = 0;                      // Limpar Flag de Interrupção por Erro Matemático
      asm("RESET");
}
```